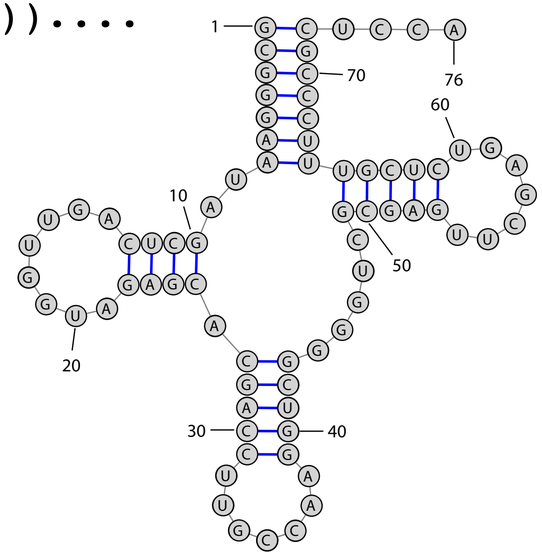
LinearFold: Linear-Time Parsing Meets RNA Folding

x GCGGGAAUAGCUCAGUUGGUAGAGCACGACCUUGCCAAGGUCGGGGUCGCGAGUUCGAGUCUCGUUUCCCGCUCCA

y ((((((((..((((........))))).((((.......)))).....(((((.......)))))))))))....



Liang Huang

Oregon State University & Baidu Research USA

Joint work with He Zhang, Dezhong Deng, Kai Zhao,
Kaibo Liu, David Hendrix and David Mathews

ISMB ECCB 2019 BASEL JULY 21–25 ISCB International Society for Computational Biology

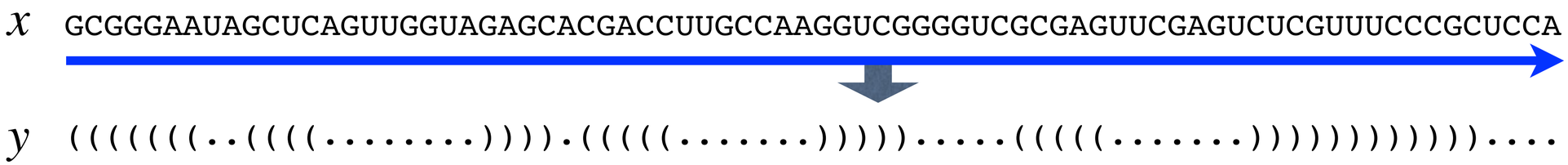
Oregon State University

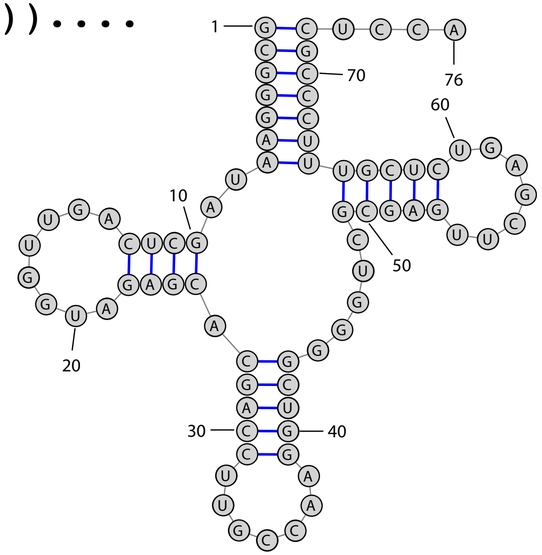
Baidu Research

University of Rochester

Southern California NLP Symposium, September 2019

LinearFold: Linear-Time Parsing Meets RNA Folding





Liang Huang

Oregon State University & Baidu Research USA

Joint work with He Zhang, Dezhong Deng, Kai Zhao,
Kaibo Liu, David Hendrix and David Mathews

BASEL JULY 21–25 ISMB ECCB 2019 ISCB INTERNATIONAL SOCIETY FOR COMPUTATIONAL BIOLOGY

Oregon State University

Baidu Research

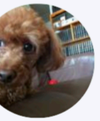
UNIVERSITY of ROCHESTER

Southern California NLP Symposium, September 2019

Appetizer: Simultaneous Translation

ACL 2019 Invited Talk

Simultaneous Translation: Recent Advances and Remaining Challenges

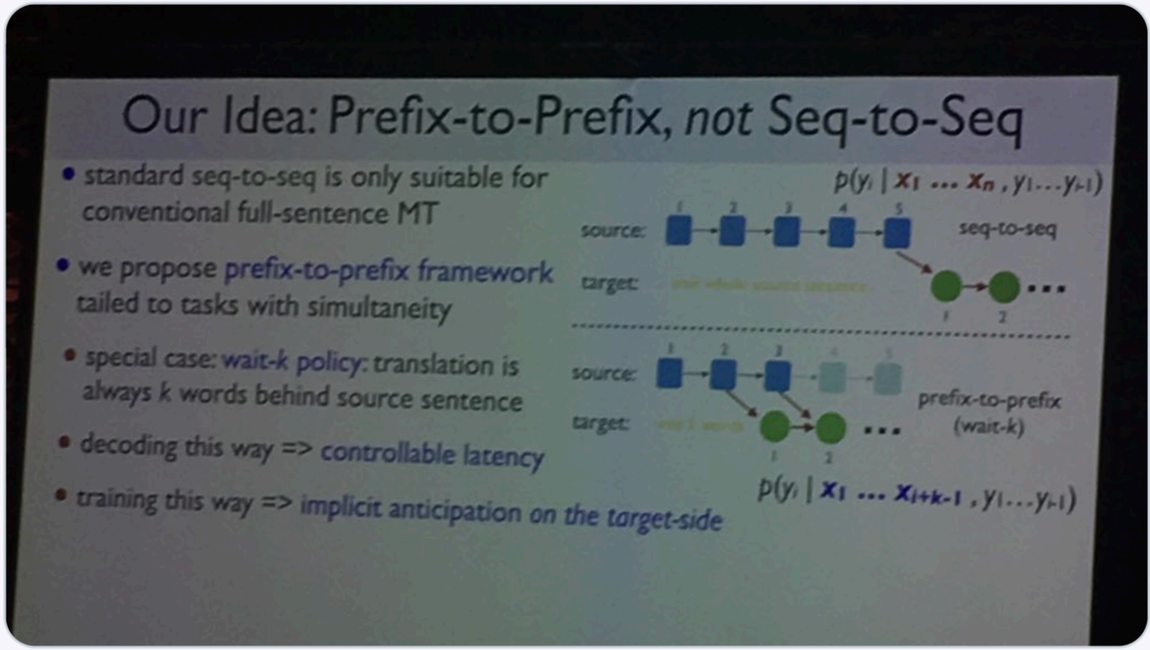


Kyunghyun Cho @kchonyc · Jul 30

very happy to see the first invited talk of the conference is on simultaneous translation. @thoma_gu I think we should've continued this direction ;)

Rada Mihalcea @radamihalcea · Jul 30

Great insight from Liang Huang's #ACL2019nlp keynote: When it comes to simultaneous tasks such as speech-to-speech MT, "forget about sequence to sequence", do instead prefix-to-prefix. Even during training when entire seqs are available, it's beneficial to only consider prefixes.



Appetizer: Simultaneous Translation

Baidu World Conference, Nov. 2017
full-sentence translation (latency: 10+ secs)

our work

Baidu World Conference, Nov. 2018
low-latency simultaneous translation (latency: ~3 secs)

Media coverage:

IEEE SPECTRUM · MIT Technology Review · CNBC · VentureBeat · SiliconANGLE · Synced AI TECHNOLOGY & INDUSTRY REVIEW · South China Morning Post · engadget · FORTUNE · I PROGRAMMER · The Register Biting the hand that feeds IT · lowyat.net malaysia's largest online community · Packt · RED PULSE · FLIPBOARD · China Knowledge

Appetizer: Simultaneous Translation

Baidu World Conference, Nov. 2017
full-sentence translation (latency: 10+ secs)

our work

Baidu World Conference, Nov. 2018
low-latency simultaneous translation (latency: ~3 secs)

Media coverage:

IEEE SPECTRUM · MIT Technology Review · CNBC · VentureBeat · SiliconANGLE · Synced AI TECHNOLOGY & INDUSTRY REVIEW · South China Morning Post · engadget · FORTUNE · I PROGRAMMER · The Register Biting the hand that feeds IT · lowyat.net malaysia's largest online community · Packt · RED PULSE · FLIPBOARD · China Knowledge

Appetizer: Simultaneous Translation

Baidu World Conference, Nov. 2017
full-sentence translation (latency: 10+ secs)

our work

Baidu World Conference, Nov. 2018
low-latency simultaneous translation (latency: ~3 secs)

Media coverage:

IEEE SPECTRUM · MIT Technology Review · CNBC · VentureBeat · SiliconANGLE · Synced AI TECHNOLOGY & INDUSTRY REVIEW · South China Morning Post · engadget · FORTUNE · I PROGRAMMER · The Register Biting the hand that feeds IT · lowyat.net malaysia's largest online community · Packt · RED PULSE · FLIPBOARD · China Knowledge

Appetizer: Simultaneous Translation

Baidu World Conference, Nov. 2017
full-sentence translation (latency: 10+ secs)

our work

Baidu World Conference, Nov. 2018
low-latency simultaneous translation (latency: ~3 secs)

Media coverage:

IEEE SPECTRUM · MIT Technology Review · CNBC · VentureBeat · SiliconANGLE · Synced AI TECHNOLOGY & INDUSTRY REVIEW · South China Morning Post · engadget · FORTUNE · I PROGRAMMER · The Register Biting the hand that feeds IT · lowyat.net malaysia's largest online community · Packt · RED PULSE · FLIPBOARD · China Knowledge

Appetizer: Simultaneous Translation

Baidu World Conference, Nov. 2017
full-sentence translation (latency: 10+ secs)

our work

Baidu World Conference, Nov. 2018
low-latency simultaneous translation (latency: ~3 secs)

Media coverage:

IEEE SPECTRUM · MIT Technology Review · CNBC · VentureBeat · SiliconANGLE · Synced AI TECHNOLOGY & INDUSTRY REVIEW · South China Morning Post · engadget · FORTUNE · I PROGRAMMER · The Register Biting the hand that feeds IT · lowyat.net malaysia's largest online community · Packt · RED PULSE · FLIPBOARD · China Knowledge

Appetizer: Simultaneous Translation

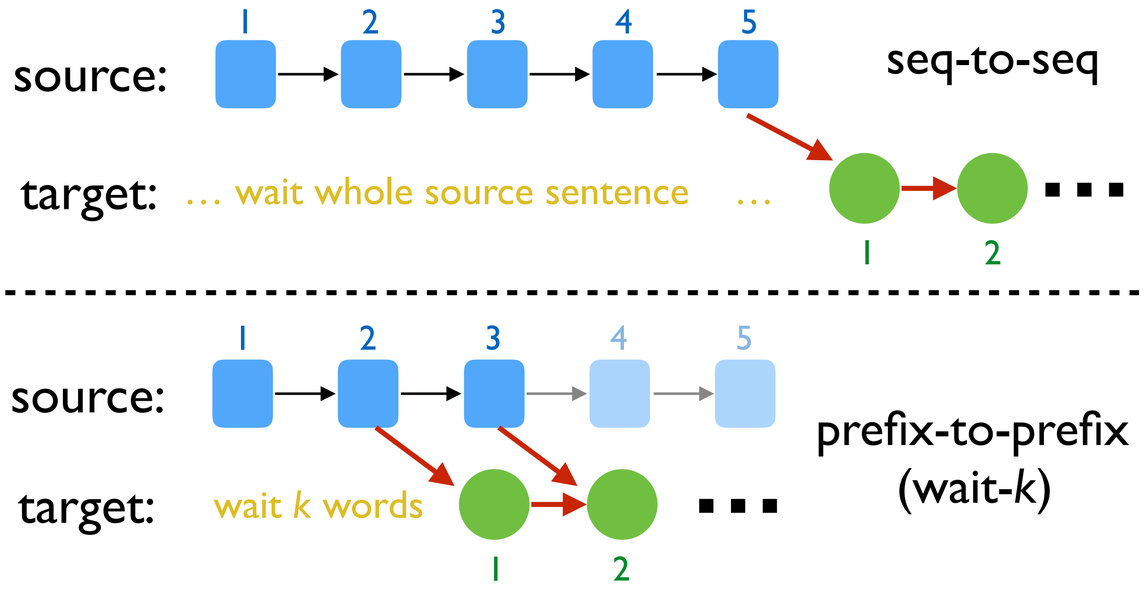
Baidu World Conference, Nov. 2017
full-sentence translation (latency: 10+ secs)

our work

Baidu World Conference, Nov. 2018
low-latency simultaneous translation (latency: ~3 secs)

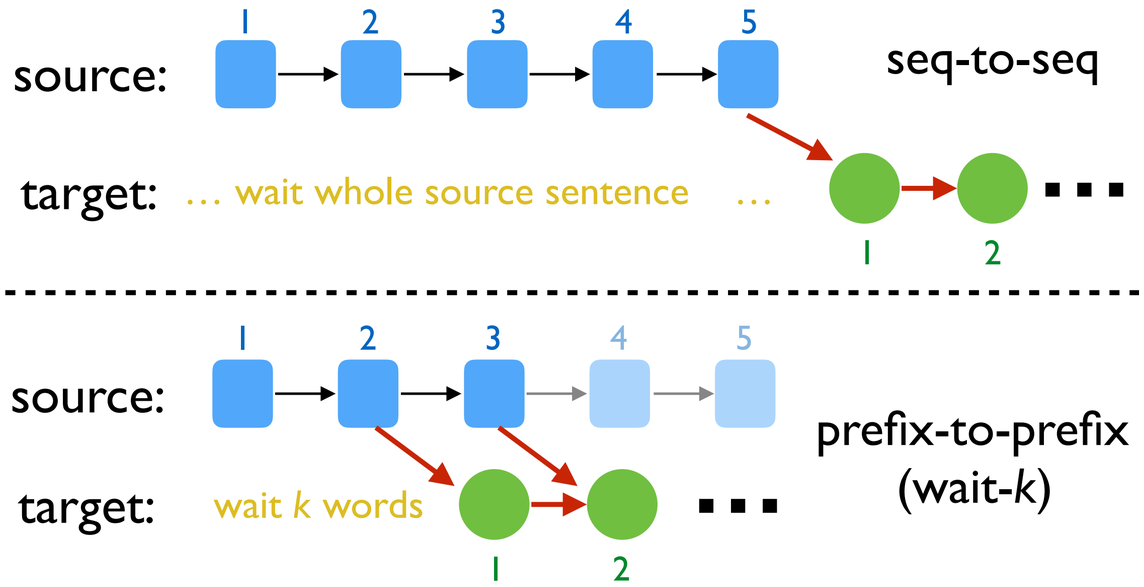
Key Idea: Prefix-to-Prefix, not Seq-to-Seq

- seq-to-seq is only suitable for conventional full-sentence MT
- we propose prefix-to-prefix, tailed to simultaneous MT
  - special case: wait-*k* policy: translation is always *k* words behind source sentence
  - training in this way enables anticipation



Key Idea: Prefix-to-Prefix, not Seq-to-Seq

- seq-to-seq is only suitable for conventional full-sentence MT
- we propose prefix-to-prefix, tailed to simultaneous MT
  - special case: wait-*k* policy: translation is always *k* words behind source sentence
  - training in this way enables anticipation



Bùshí *zǒngtǒng*

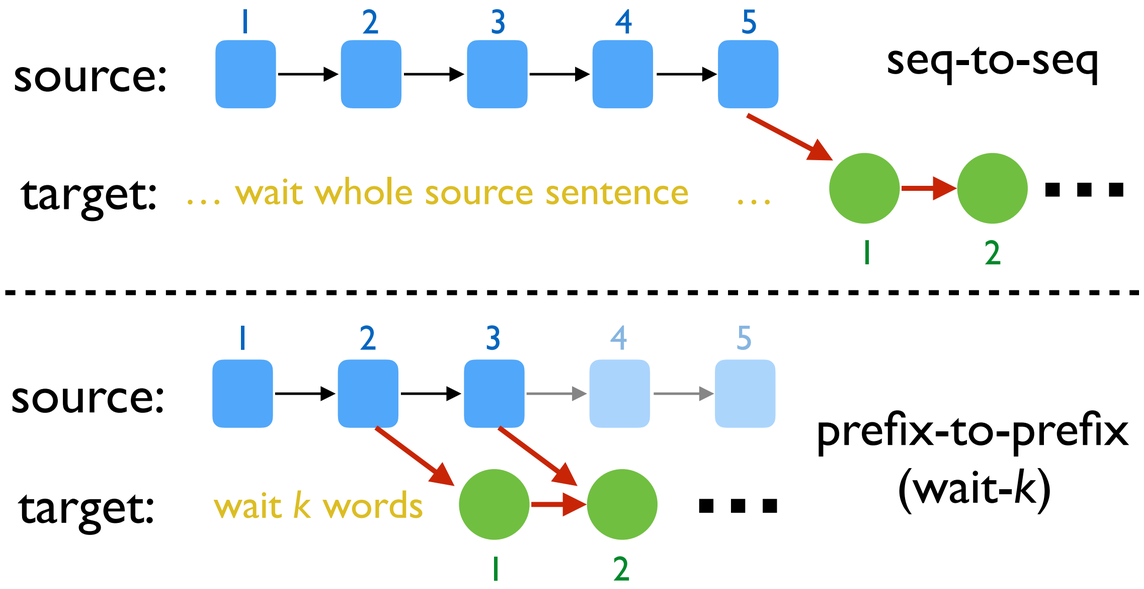
布什 总统

Bush *President*

President

Key Idea: Prefix-to-Prefix, not Seq-to-Seq

- seq-to-seq is only suitable for conventional full-sentence MT
- we propose prefix-to-prefix, tailed to simultaneous MT
  - special case: wait-*k* policy: translation is always *k* words behind source sentence
  - training in this way enables anticipation

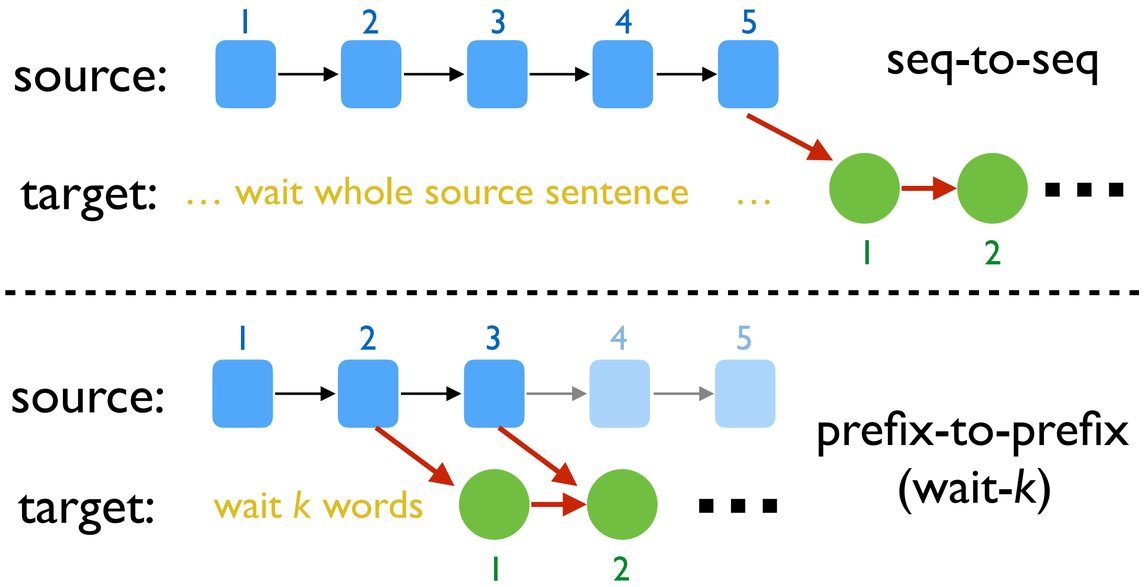


Bùshí	*zǒngtǒng*	*zài*
布什	总统	在
Bush	*President*	*in*

President Bush

Key Idea: Prefix-to-Prefix, not Seq-to-Seq

- seq-to-seq is only suitable for conventional full-sentence MT
- we propose prefix-to-prefix, tailed to simultaneous MT
  - special case: wait-*k* policy: translation is always *k* words behind source sentence
  - training in this way enables anticipation

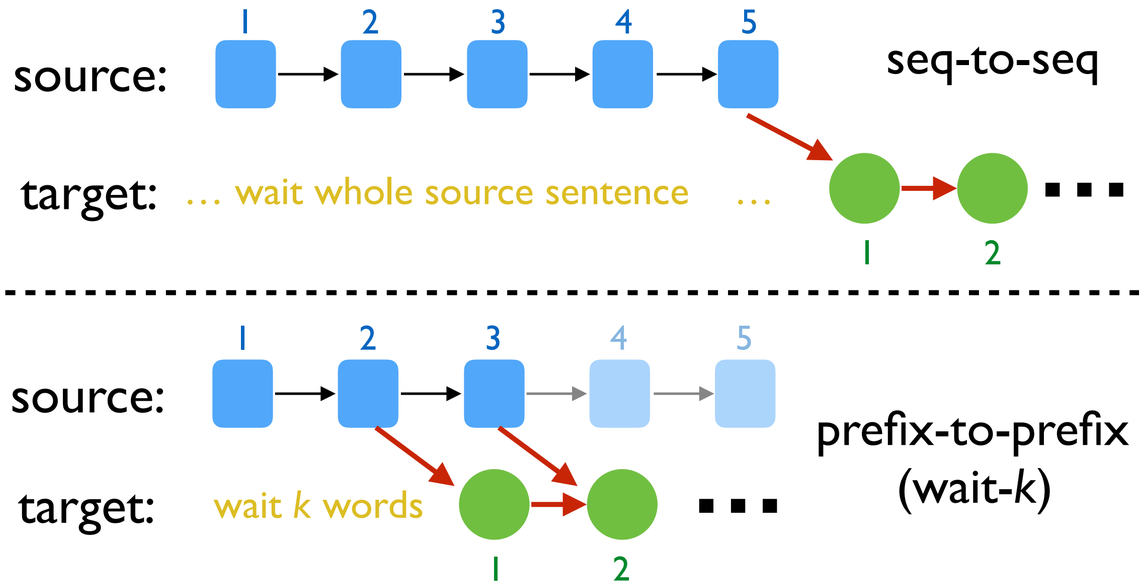


Bùshí	*zǒngtǒng*	*zài*	*Mòsīkē*
布什	总统	在	莫斯科
Bush	*President*	*in*	*Moscow*

President Bush meets

Key Idea: Prefix-to-Prefix, not Seq-to-Seq

- seq-to-seq is only suitable for conventional full-sentence MT
- we propose prefix-to-prefix, tailed to simultaneous MT
  - special case: wait-*k* policy: translation is always *k* words behind source sentence
  - training in this way enables anticipation

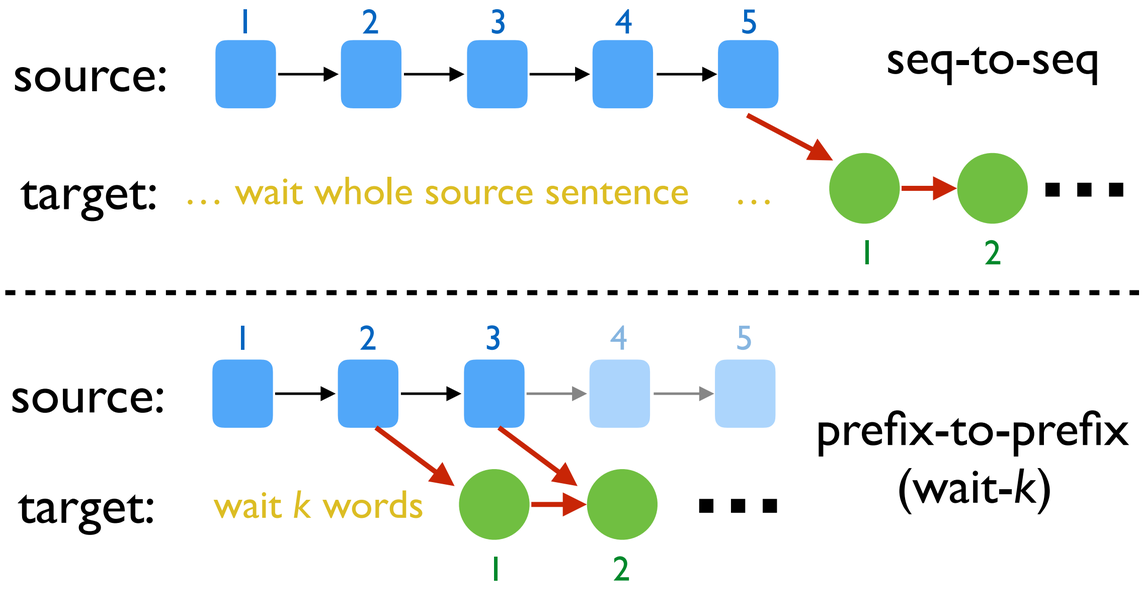


Bùshí	*zǒngtǒng*	*zài*	*Mòsīkē*	*yǔ*
布什	总统	在	莫斯科	与
Bush	*President*	*in*	*Moscow*	*with*

President Bush meets with

Key Idea: Prefix-to-Prefix, not Seq-to-Seq

- seq-to-seq is only suitable for conventional full-sentence MT
- we propose prefix-to-prefix, tailed to simultaneous MT
  - special case: wait-*k* policy: translation is always *k* words behind source sentence
  - training in this way enables anticipation

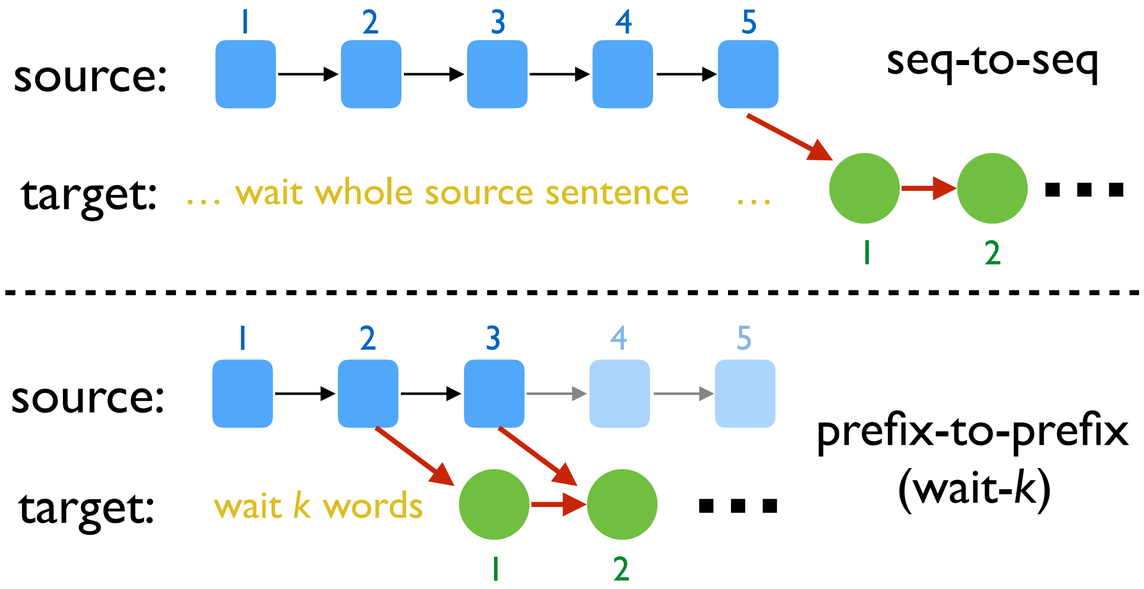


Bùshí	*zǒngtǒng*	*zài*	*Mòsīkē*	*yǔ*	*Éluósī*
布什	总统	在	莫斯科	与	俄罗斯
Bush	*President*	*in*	*Moscow*	*with*	*Russian*

President Bush meets with Russian

Key Idea: Prefix-to-Prefix, not Seq-to-Seq

- seq-to-seq is only suitable for conventional full-sentence MT
- we propose prefix-to-prefix, tailed to simultaneous MT
  - special case: wait-*k* policy: translation is always *k* words behind source sentence
  - training in this way enables anticipation

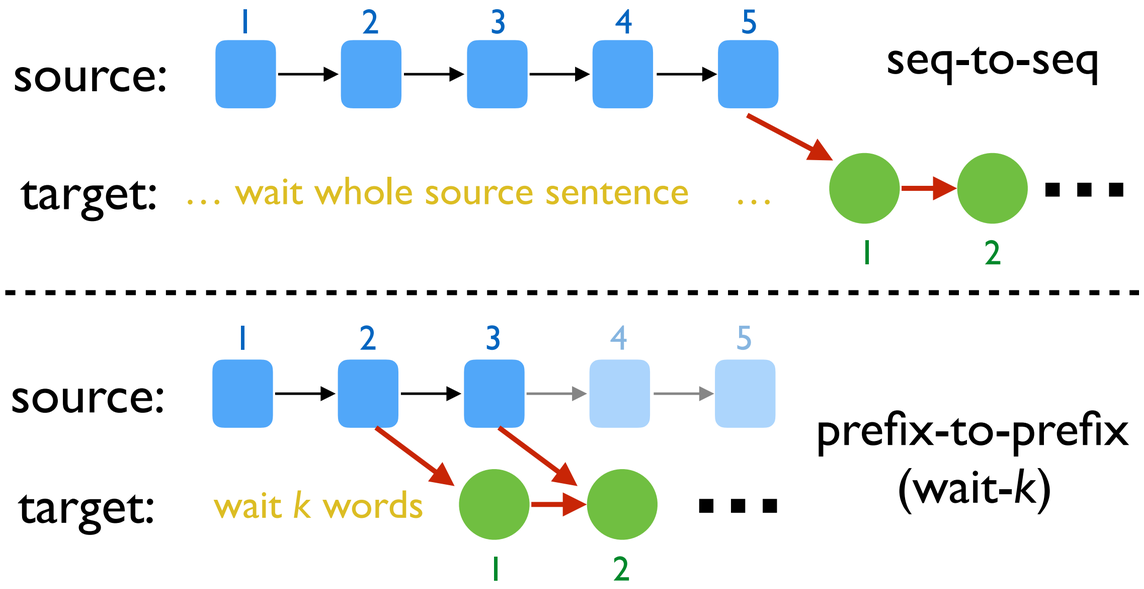


Bùshí	*zǒngtǒng*	*zài*	*Mòsīkē*	*yǔ*	*Éluósī*	*zǒngtǒng*
布什	总统	在	莫斯科	与	俄罗斯	总统
Bush	*President*	*in*	*Moscow*	*with*	*Russian*	*President*

President Bush meets with Russian President

Key Idea: Prefix-to-Prefix, not Seq-to-Seq

- seq-to-seq is only suitable for conventional full-sentence MT
- we propose prefix-to-prefix, tailed to simultaneous MT
  - special case: wait-*k* policy: translation is always *k* words behind source sentence
  - training in this way enables anticipation

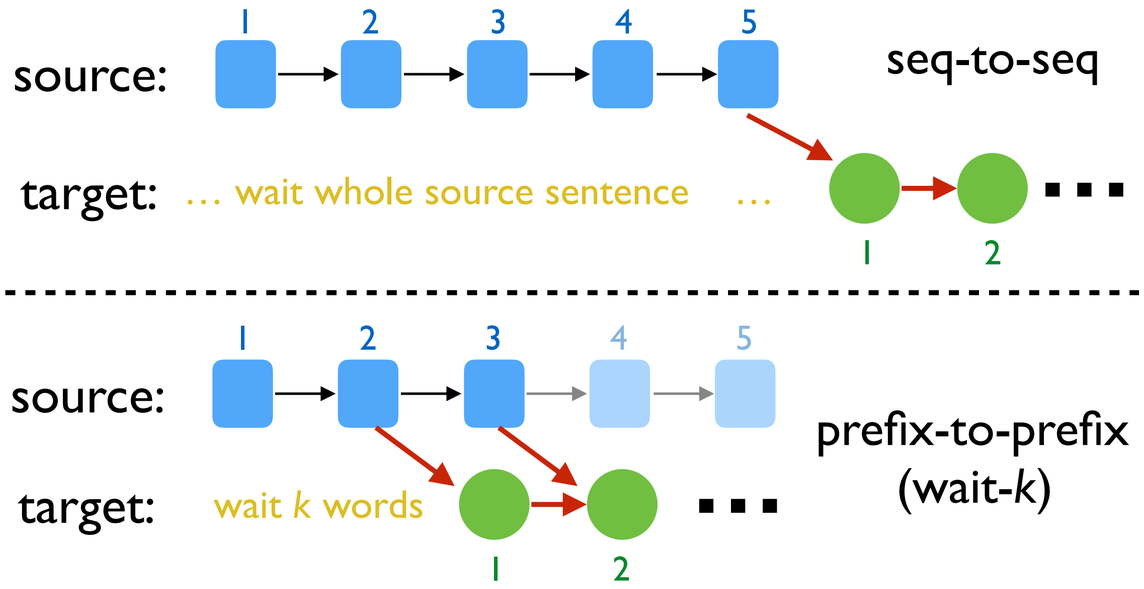


Bùshí	*zǒngtǒng*	*zài*	*Mòsīkē*	*yǔ*	*Éluósī*	*zǒngtǒng*	*Pǔjīng*
布什	总统	在	莫斯科	与	俄罗斯	总统	普京
Bush	*President*	*in*	*Moscow*	*with*	*Russian*	*President*	*Putin*

President Bush meets with Russian President Putin

Key Idea: Prefix-to-Prefix, not Seq-to-Seq

- seq-to-seq is only suitable for conventional full-sentence MT
- we propose prefix-to-prefix, tailed to simultaneous MT
  - special case: wait-*k* policy: translation is always *k* words behind source sentence
  - training in this way enables anticipation



Bùshí	*zǒngtǒng*	*zài*	*Mòsīkē*	*yǔ*	*Éluósī*	*zǒngtǒng*	*Pǔjīng*	*huìwù*
布什	总统	在	莫斯科	与	俄罗斯	总统	普京	会晤
Bush	*President*	*in*	*Moscow*	*with*	*Russian*	*President*	*Putin*	*meet*

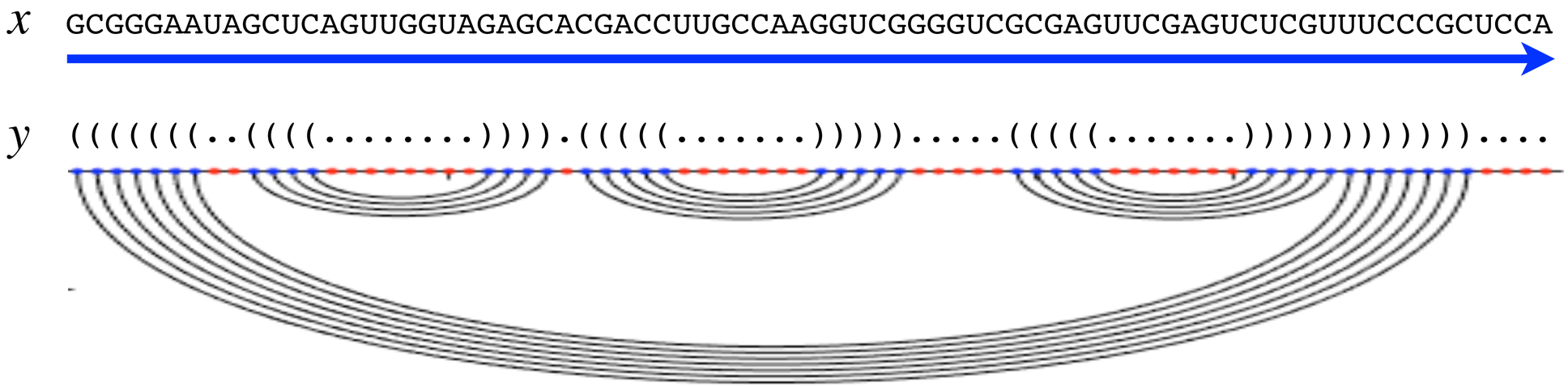
President Bush meets with Russian President Putin in Moscow

Entreé: Linear-Time RNA Folding

x GCGGGAAUAGCUCAGUUGGUAGAGCACGACCUUGCCAAGGUCGGGGUCGCGAGUUCGAGUCUCGUUUCCCGCUCCA

y ((((((..((((........))))).((((.......)))).....((((.......))))))))))....

Entreé: Linear-Time RNA Folding

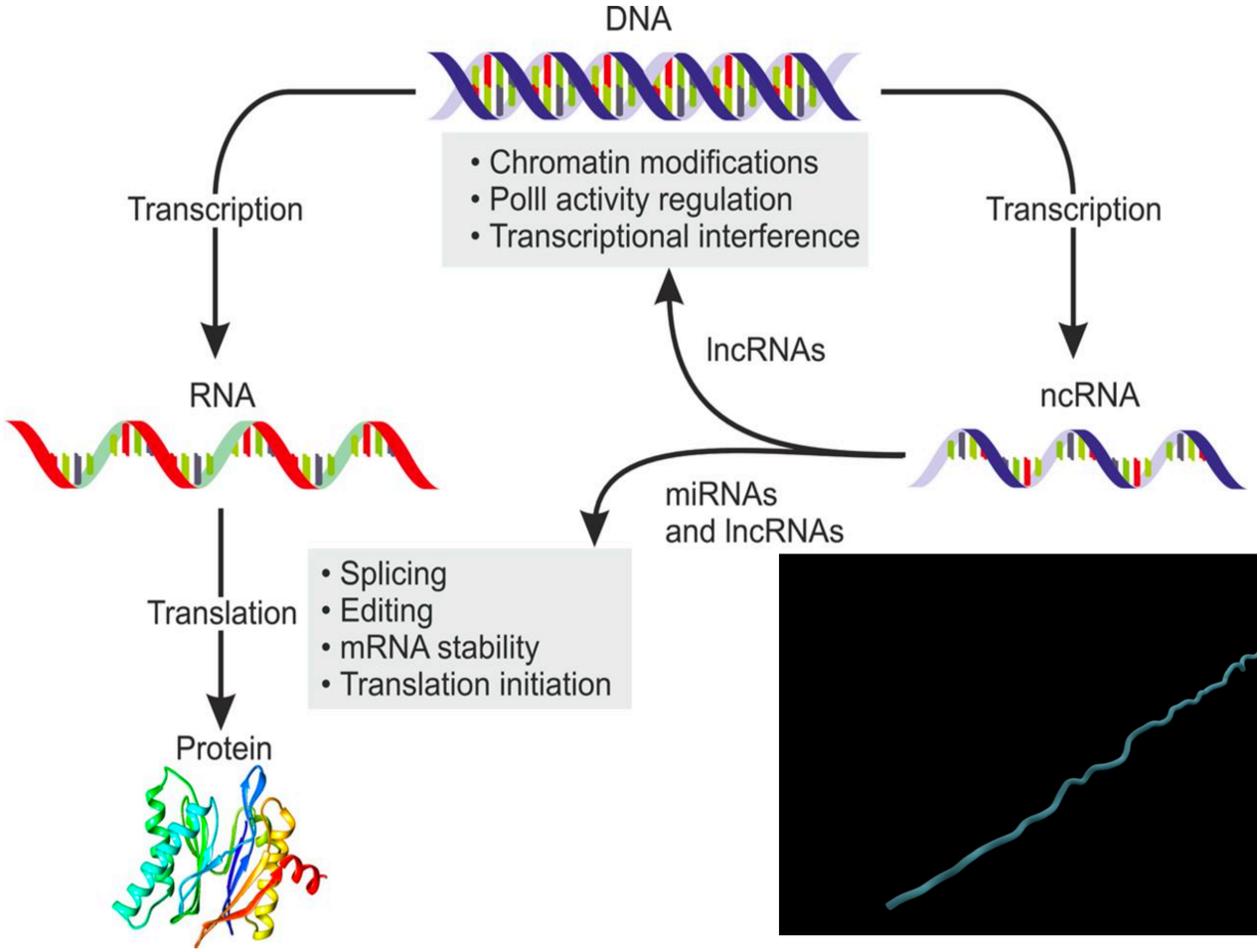


Background: RNAs and RNA folding

RNA has dual roles:
informational (DNA=>RNA=>protein)
functional (non-coding RNAs)
knowing structures can infer function

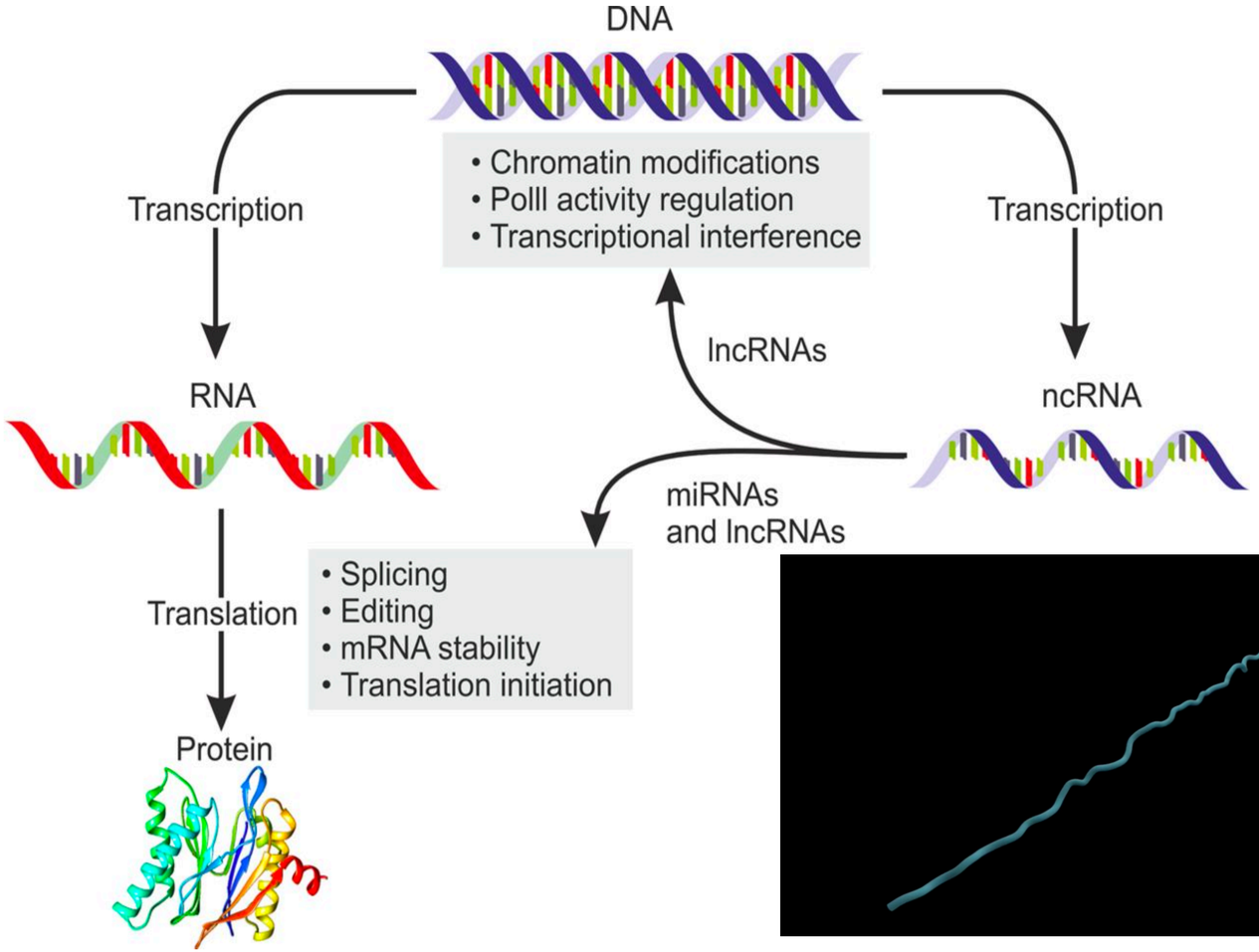
RNA sequence

GCGGGAAUAGCUCAGUUGGUAGAGCACGACCUUGCCAAGGUCGGGGUCGCGAGUUCGAGUCUCGUUUCCCGCUCCA



Background: RNAs and RNA folding

RNA has dual roles:
informational (DNA=>RNA=>protein)
functional (non-coding RNAs)
knowing structures can infer function

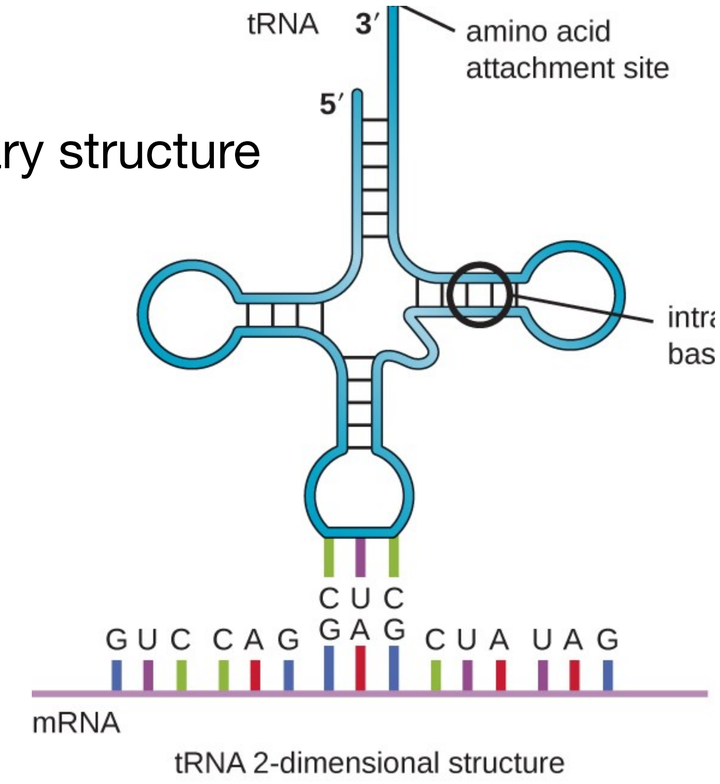


RNA sequence

GCGGGAAUAGCUCAGUUGGUAGAGCACGACCUUGCCAAGGUCGGGGUCGCGAGUUCGAGUCUCGUUUCCCGCUCCA

structure prediction
("RNA folding")

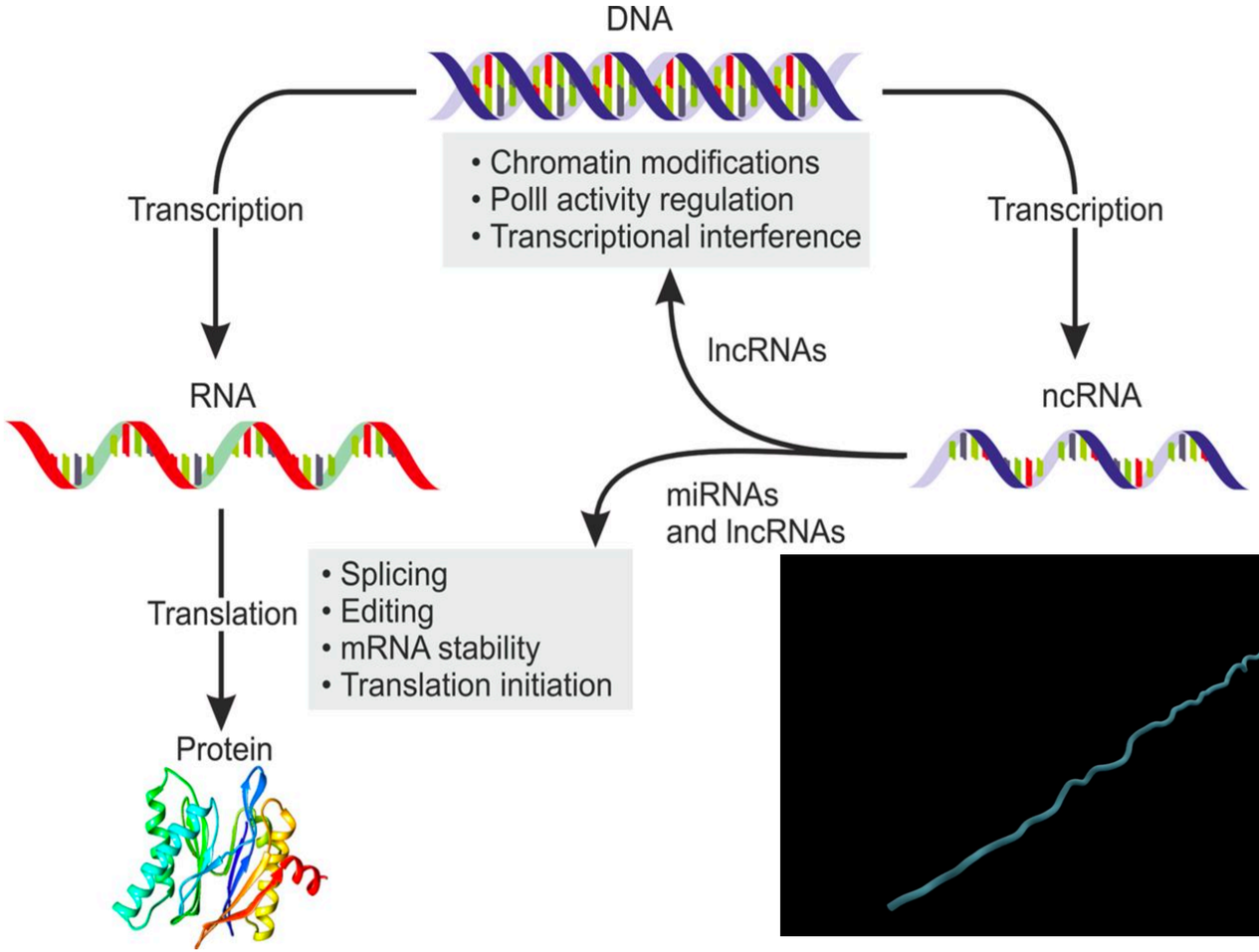
secondary structure



tRNA 2-dimensional structure

Background: RNAs and RNA folding

RNA has dual roles:
informational (DNA=>RNA=>protein)
functional (non-coding RNAs)
knowing structures can infer function

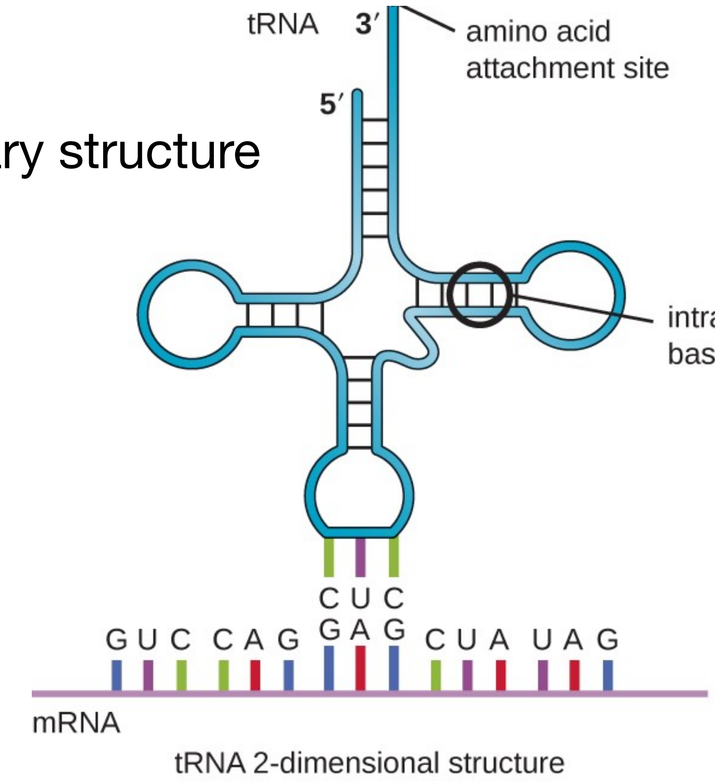


RNA sequence

GCGGGAAUAGCUCAGUUGGUAGAGCACGACCUUGCCAAGGUCGGGGUCGCGAGUUCGAGUCUCGUUUCCCGCUCCA

structure prediction
("RNA folding")

secondary structure



tRNA 2-dimensional structure

RNA Secondary Structure Prediction

allowed pairs: G-C A-U G-U

assume no crossing pairs
(no pseudoknots)

example: transfer RNA (tRNA)

input x GCGGGAAUAGCUCAGUUGGUAGAGCACGACCUUGCCAAGGUCGGGGUCGCGAGUUCGAGUCUCGUUUCCCGCUCCA

RNA Secondary Structure Prediction

allowed pairs: G-C A-U G-U

assume no crossing pairs
(no pseudoknots)

example: transfer RNA (tRNA)

input x `GCGGGAAUAGCUCAGUUGGUAGAGCACGACCUUGCCAAGGUCGGGGUCGCGAGUUCGAGUCUCGUUUCCCGCUCCA`

output y `((((((((..((((........))))).((((.......))))).....((((.......))))))))))....`

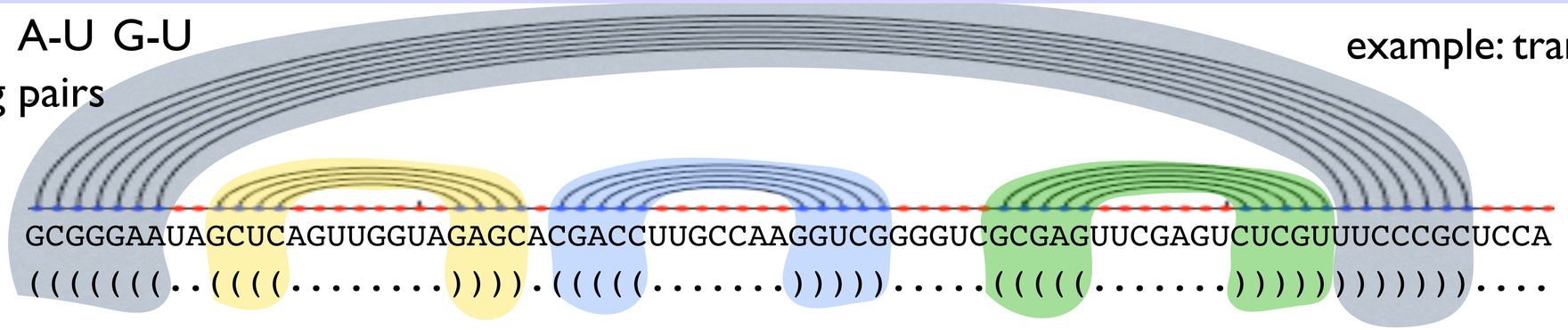
RNA Secondary Structure Prediction

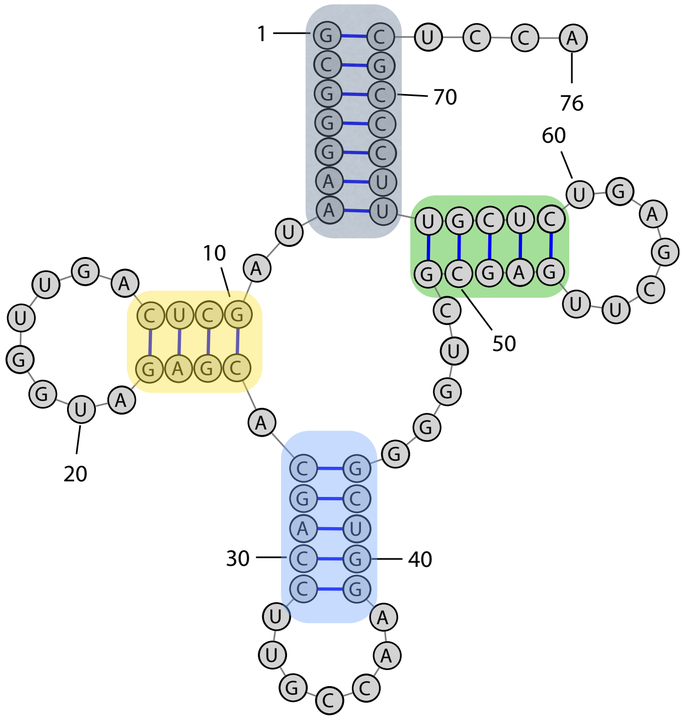
allowed pairs: G-C A-U G-U

assume no crossing pairs

(no pseudoknots)

example: transfer RNA (tRNA)





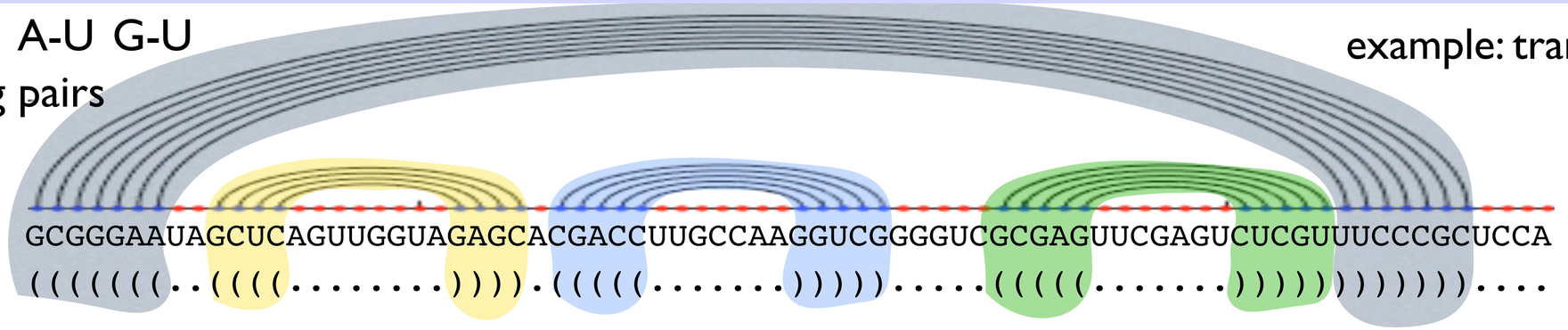
RNA Secondary Structure Prediction

allowed pairs: G-C A-U G-U

assume no crossing pairs

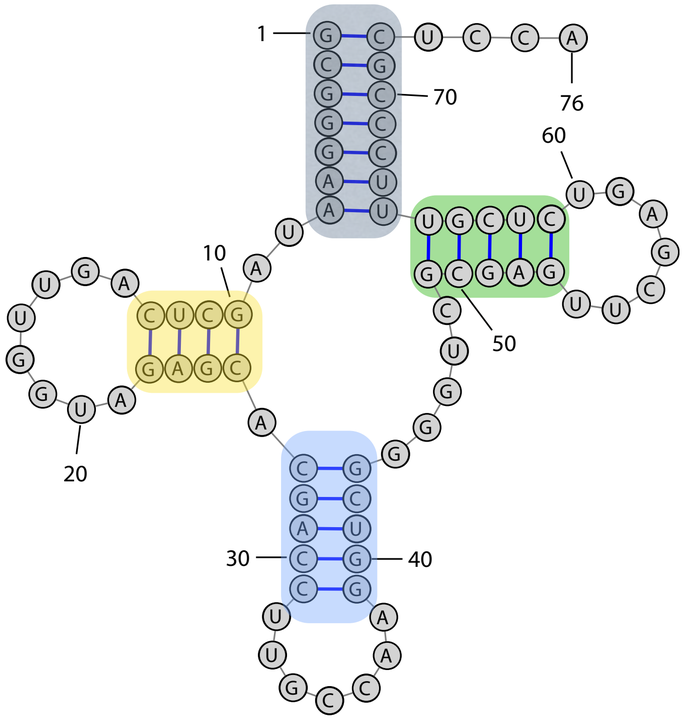
(no pseudoknots)

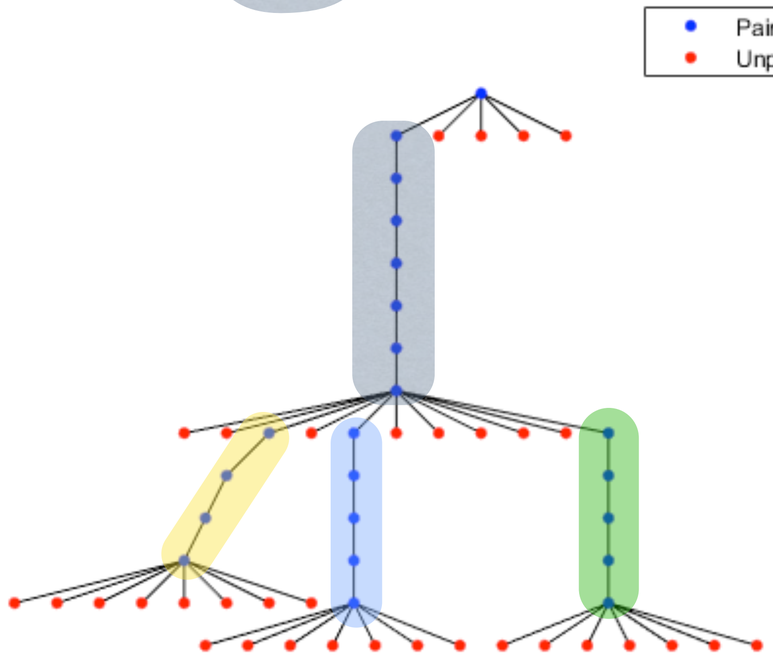
example: transfer RNA (tRNA)



input x GCGGGAAUAGCUCAGUUGGUAGAGCACGACCUUGCCAAGGUCGGGGUCGCGAGUUCGAGUCUCGUUUCCCGCUCCA

output y ((((((..((((........))))..((((.......))))).....((((.......))))))))))....





parse tree

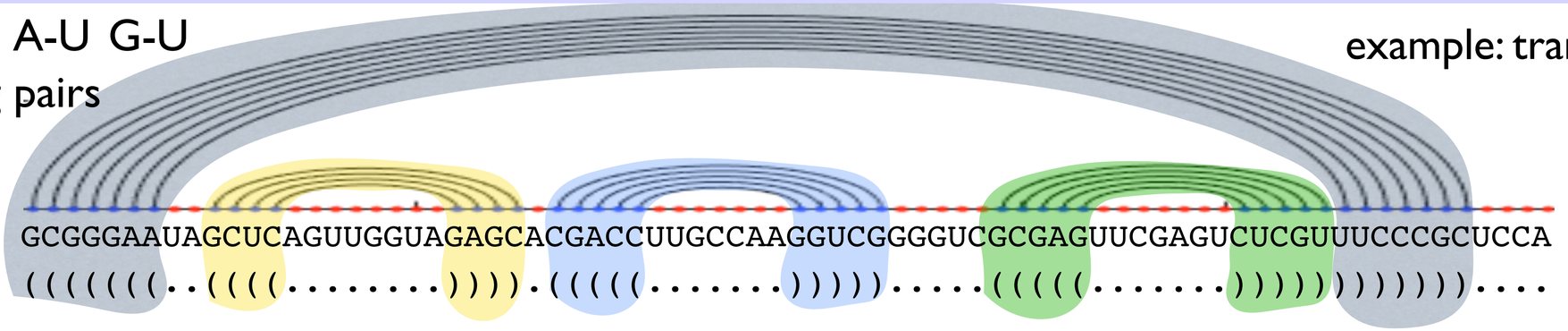
RNA Secondary Structure Prediction

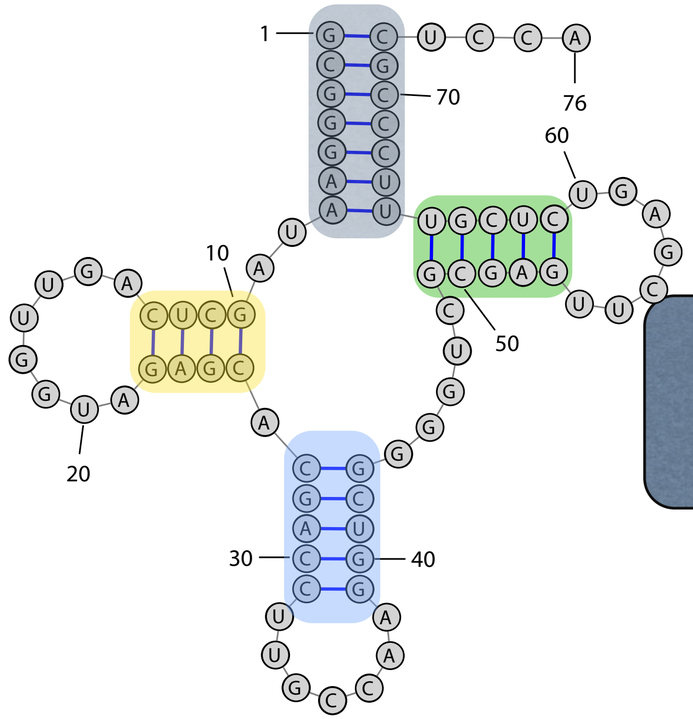
allowed pairs: G-C A-U G-U

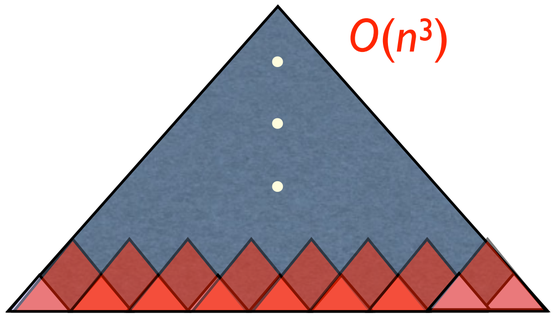
assume no crossing pairs

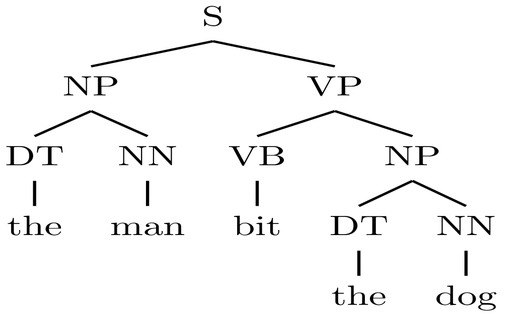
(no pseudoknots)

example: transfer RNA (tRNA)

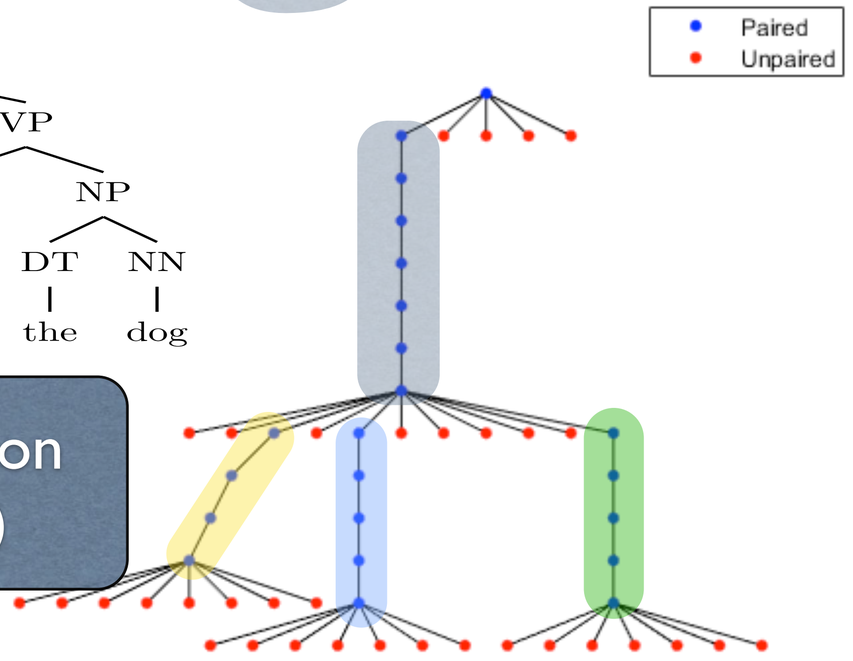








problem: standard structure prediction algorithms are way too slow: $O(n^3)$



parse tree

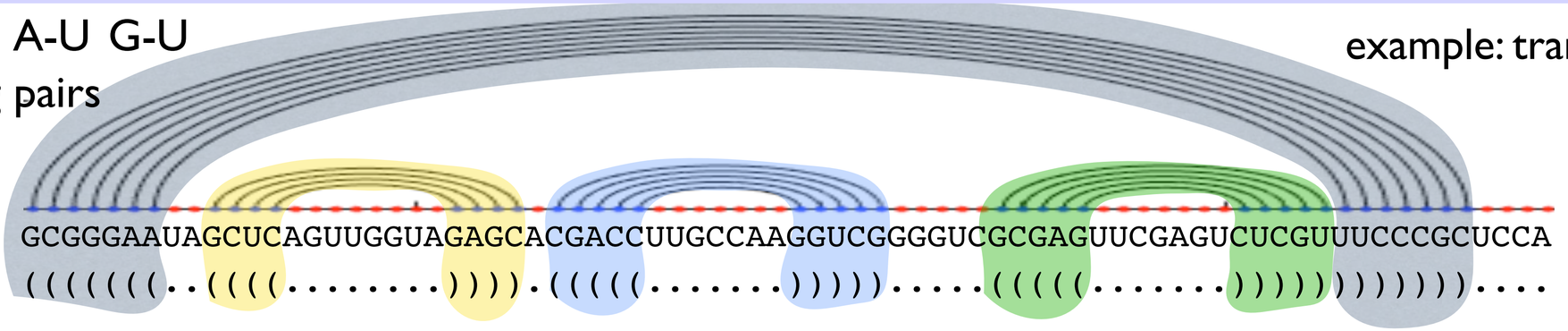
RNA Secondary Structure Prediction

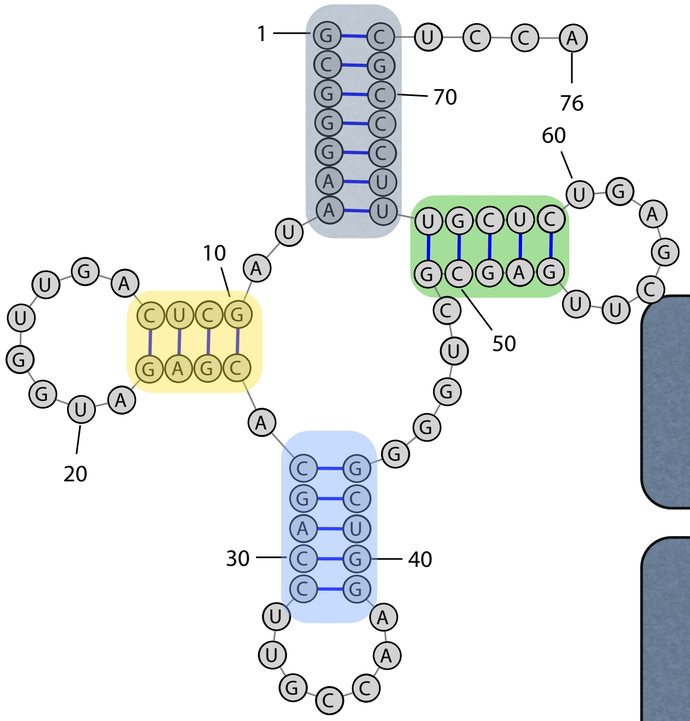
allowed pairs: G-C A-U G-U

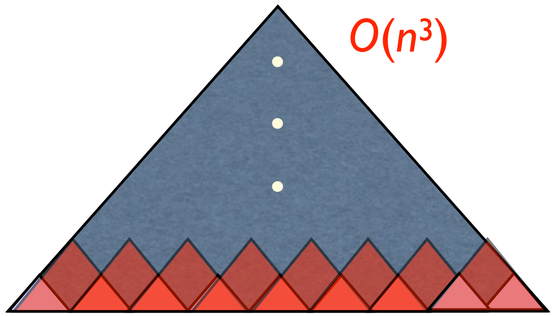
assume no crossing pairs

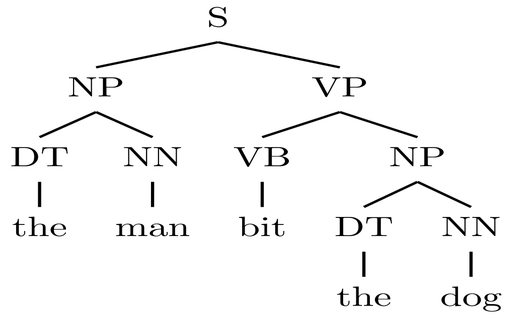
(no pseudoknots)

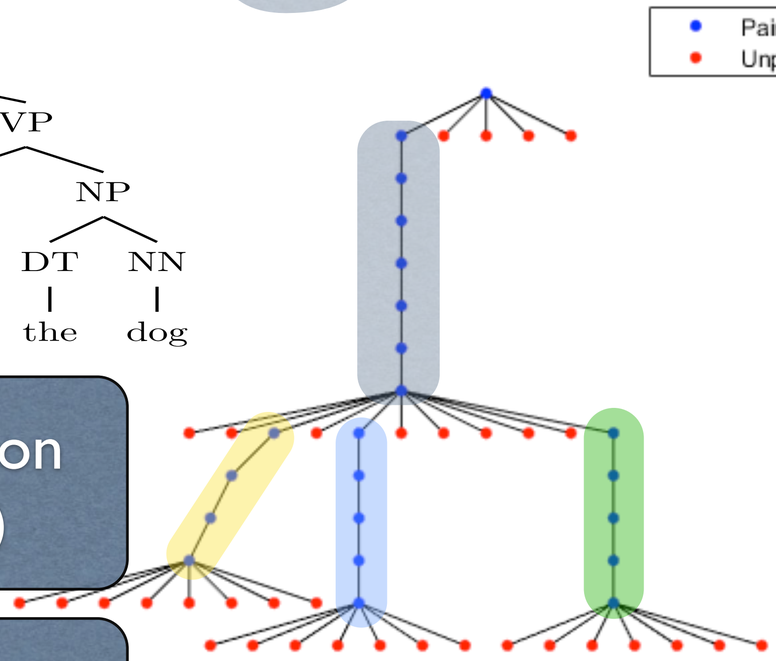
example: transfer RNA (tRNA)











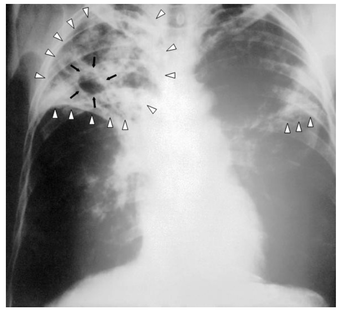
parse tree

problem: standard structure prediction algorithms are way too slow: $O(n^3)$

solution: adapt my linear-time dynamic programming algorithms from parsing

Fast Structure Prediction Enables RNA Design

detecting active TB using RNA design
which needs our fast RNA folding



Professor **Rhiju Das**
Stanford Medical School

EteRNA game
(RNA design)

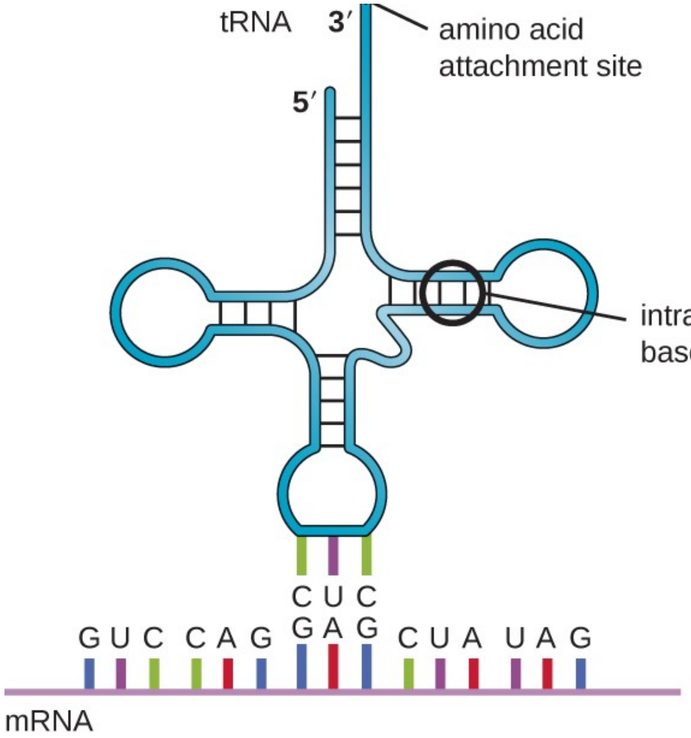
RNA sequence

GCGGGAAUAGCUCAGUUGGUAGAGCACGACCUUGCCAAGGUCGGGGUCGCGAGUUCGAGUCUCGUUUCCCGCUCCA

structure prediction ("folding")

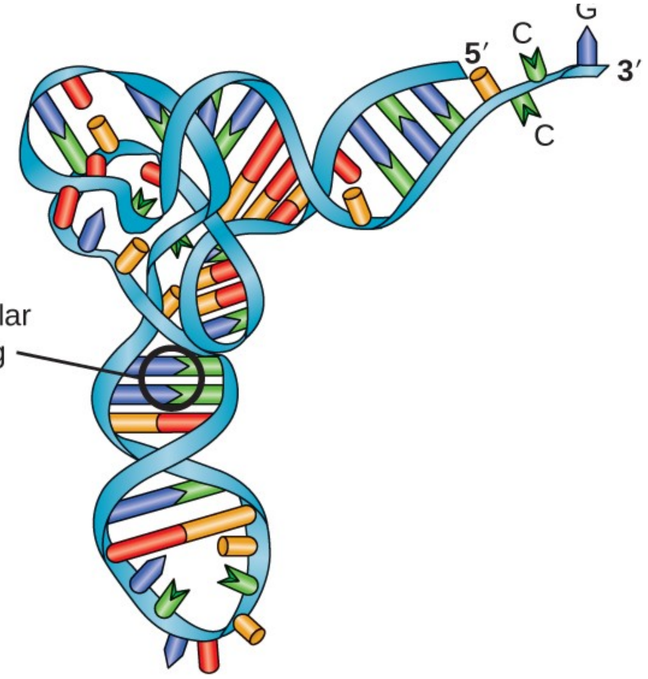
design

RNA secondary structure



tRNA 2-dimensional structure

RNA 3D structure



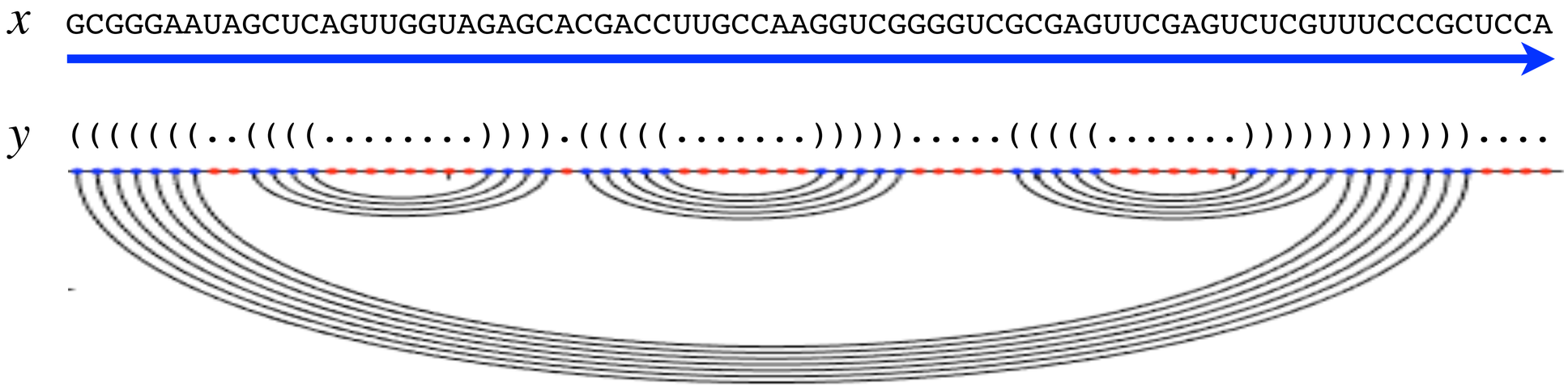
tRNA 3-dimensional structure

From Linguistics to Biology

x GCGGGAAUAGCUCAGUUGGUAGAGCACGACCUUGCCAAGGUCGGGGUCGCGAGUUCGAGUCUCGUUUCCCGCUCCA

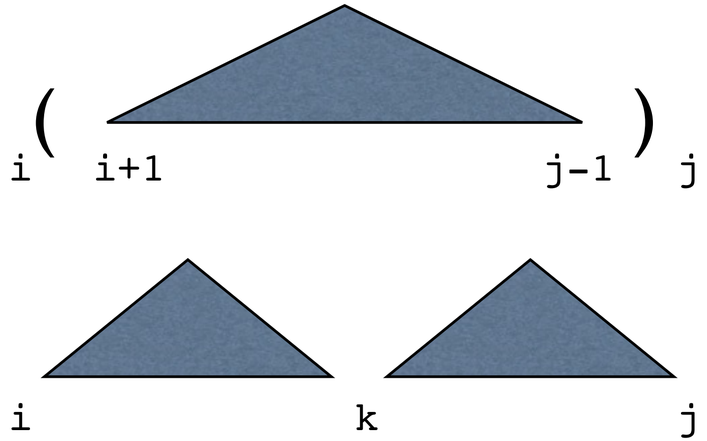
y ((((((..((((........))))).((((.......)))).....((((.......))))))))))....

From Linguistics to Biology



Background: CKY for RNA Folding: $O(n^3)$

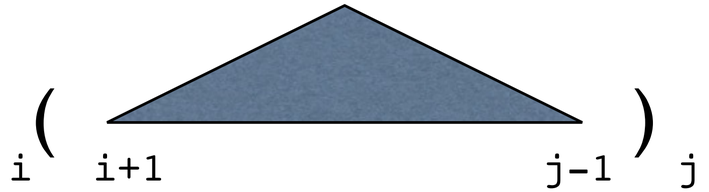
- Dynamic Programming — $O(n^3)$
  - bottom-up CKY parsing
  - example: maximize # of pairs (A-U, G-C, or G-U)

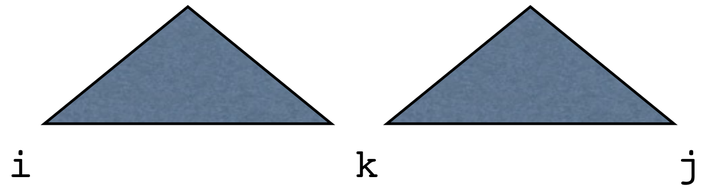


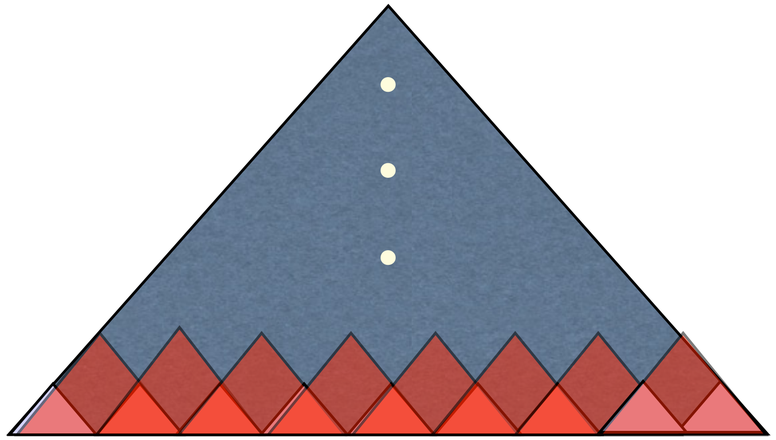
A C A G U

Background: CKY for RNA Folding: $O(n^3)$

- Dynamic Programming — $O(n^3)$
  - bottom-up CKY parsing
  - example: maximize # of pairs (A-U, G-C, or G-U)



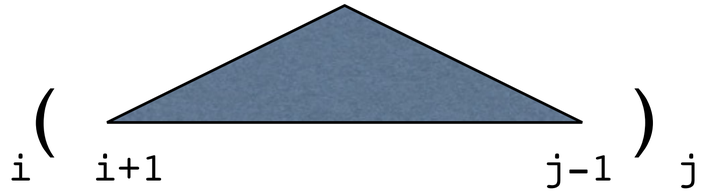


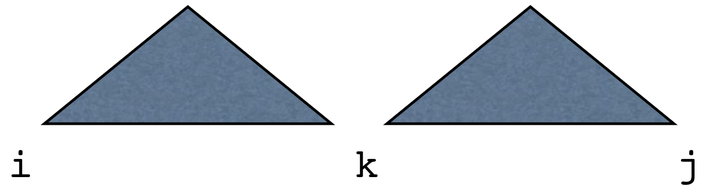


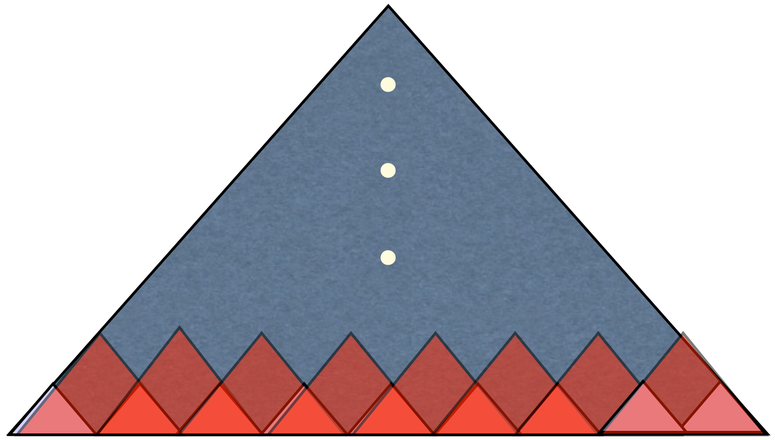
A C A G U

Background: CKY for RNA Folding: $O(n^3)$

- Dynamic Programming — $O(n^3)$
  - bottom-up CKY parsing
  - example: maximize # of pairs (A-U, G-C, or G-U)



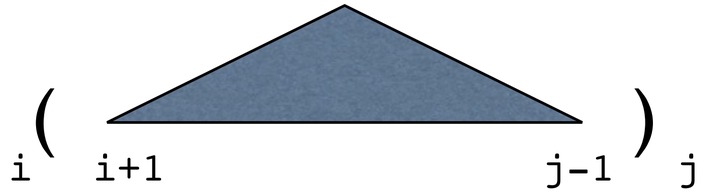


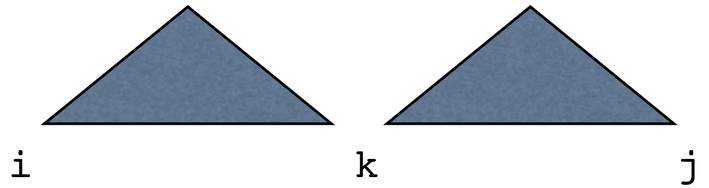


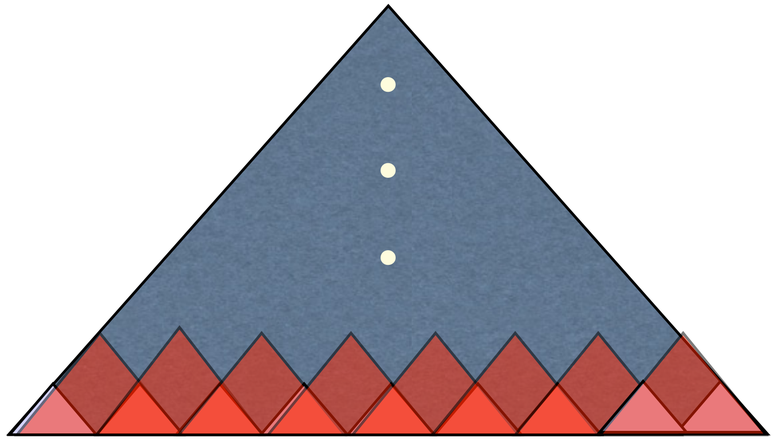
A C A G U

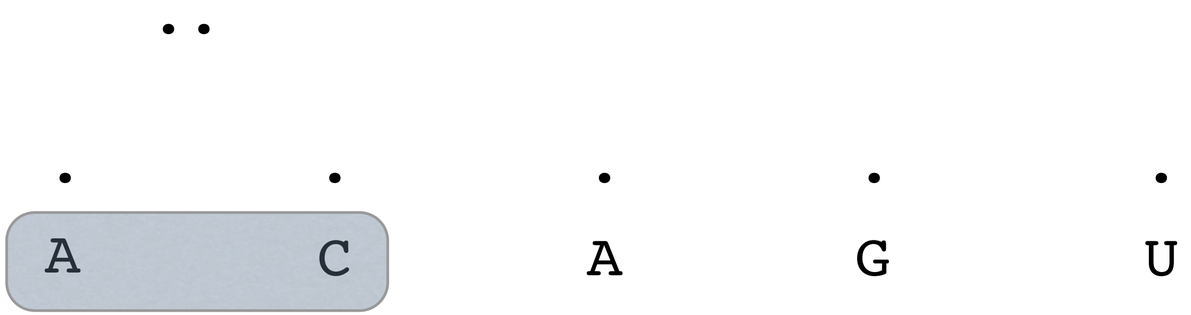
Background: CKY for RNA Folding: $O(n^3)$

- Dynamic Programming — $O(n^3)$
  - bottom-up CKY parsing
  - example: maximize # of pairs (A-U, G-C, or G-U)



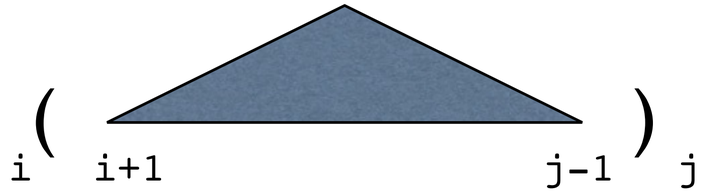


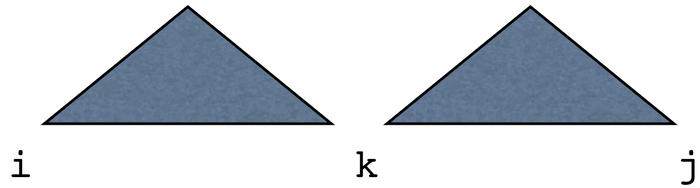


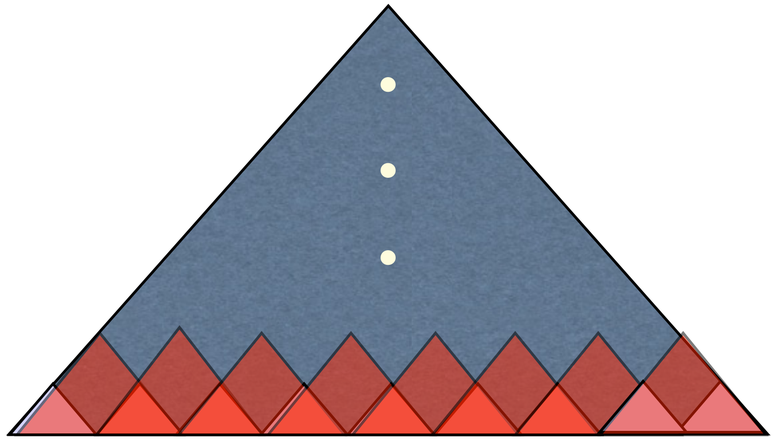


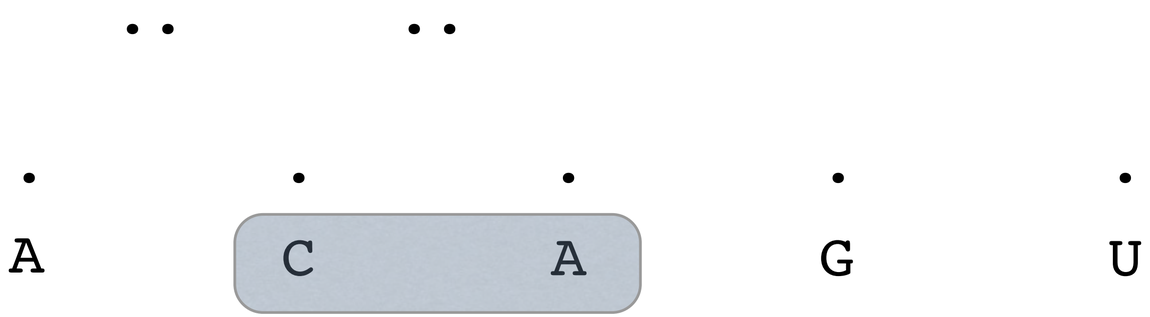
Background: CKY for RNA Folding: $O(n^3)$

- Dynamic Programming — $O(n^3)$
  - bottom-up CKY parsing
  - example: maximize # of pairs (A-U, G-C, or G-U)



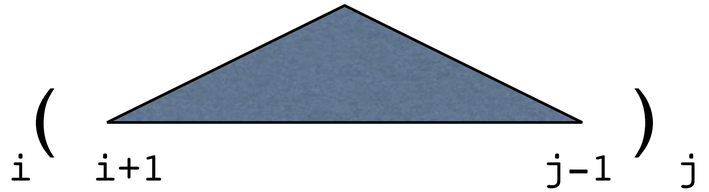


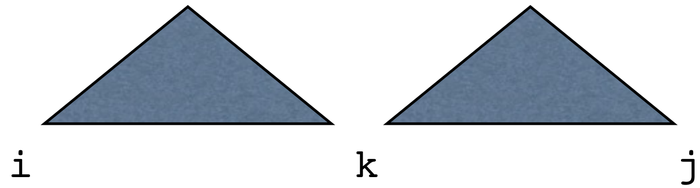


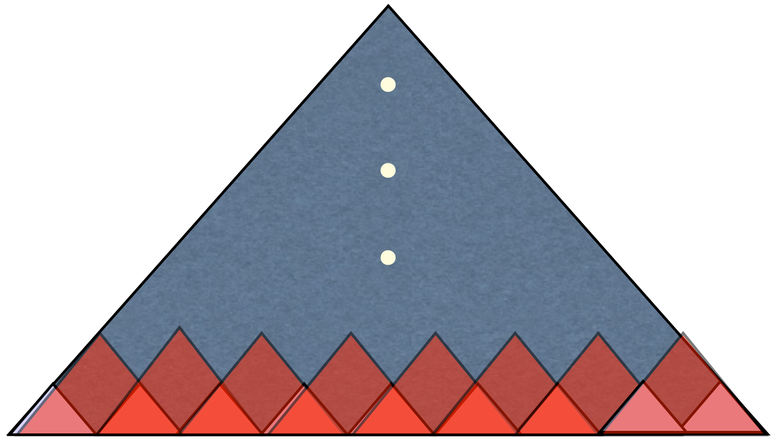


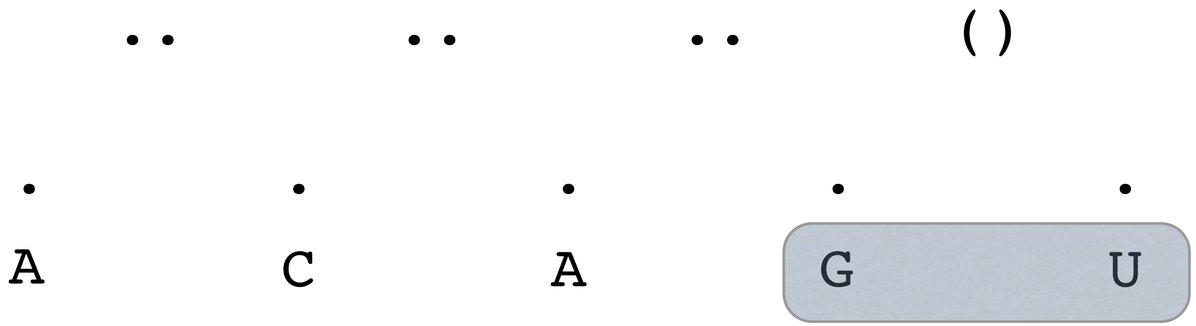
Background: CKY for RNA Folding: $O(n^3)$

- Dynamic Programming — $O(n^3)$
  - bottom-up CKY parsing
  - example: maximize # of pairs (A-U, G-C, or G-U)



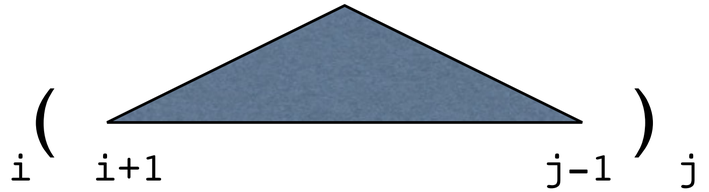


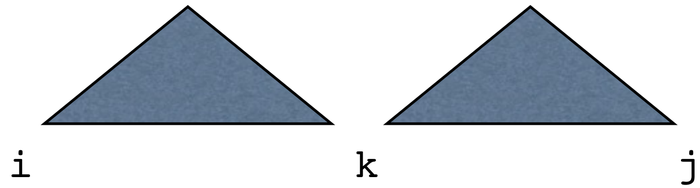


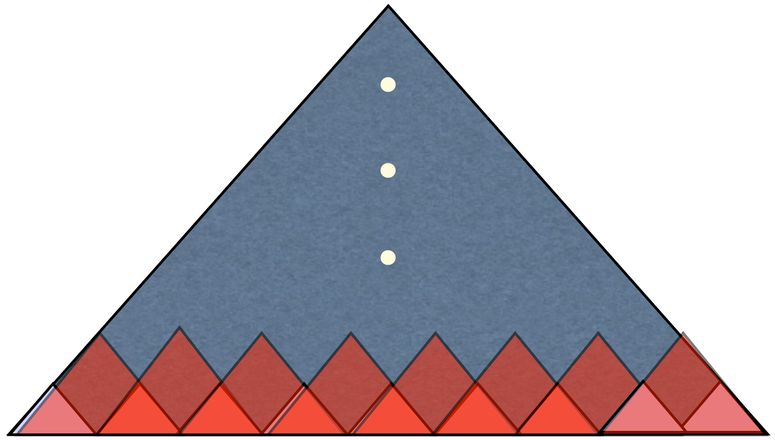


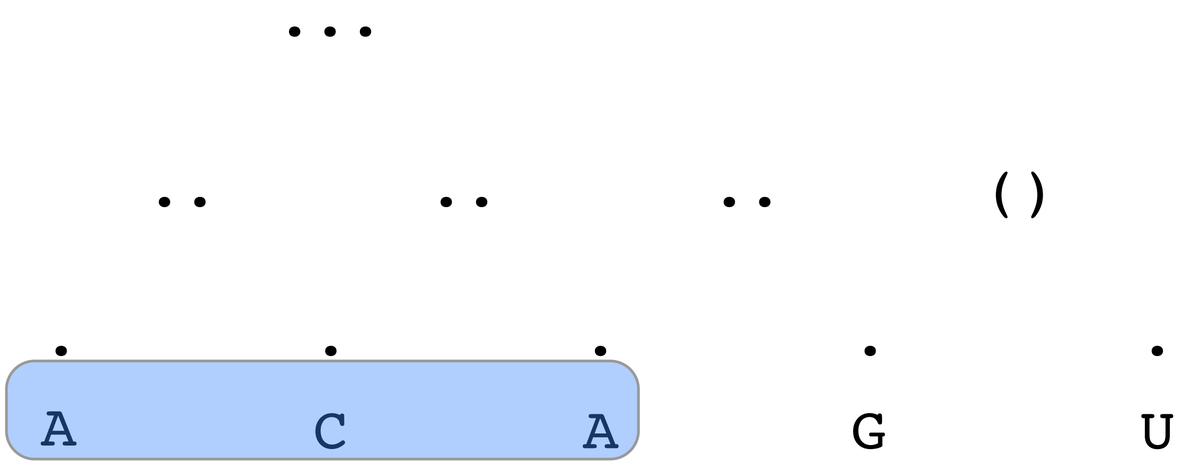
Background: CKY for RNA Folding: $O(n^3)$

- Dynamic Programming — $O(n^3)$
  - bottom-up CKY parsing
  - example: maximize # of pairs (A-U, G-C, or G-U)



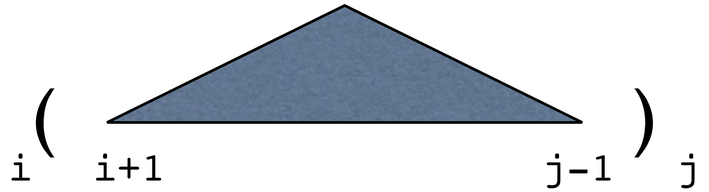


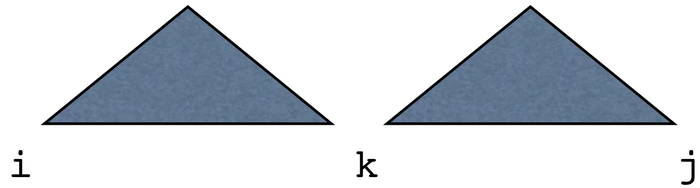


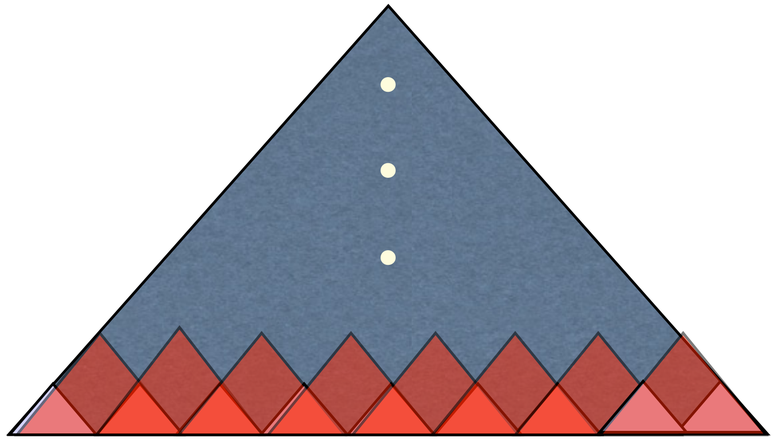


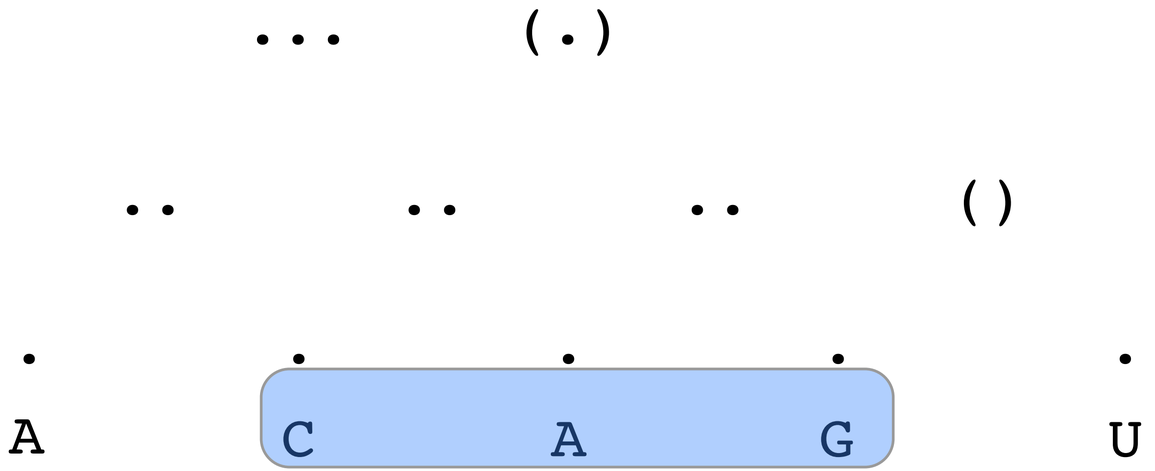
Background: CKY for RNA Folding: $O(n^3)$

- Dynamic Programming — $O(n^3)$
  - bottom-up CKY parsing
  - example: maximize # of pairs (A-U, G-C, or G-U)



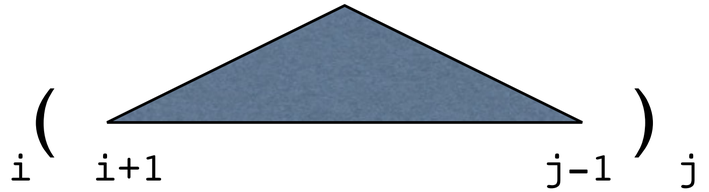


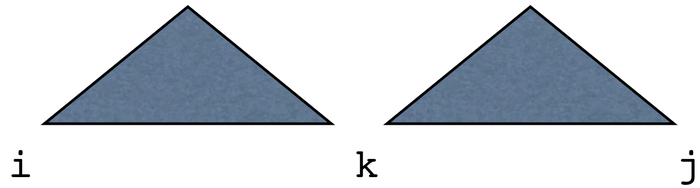


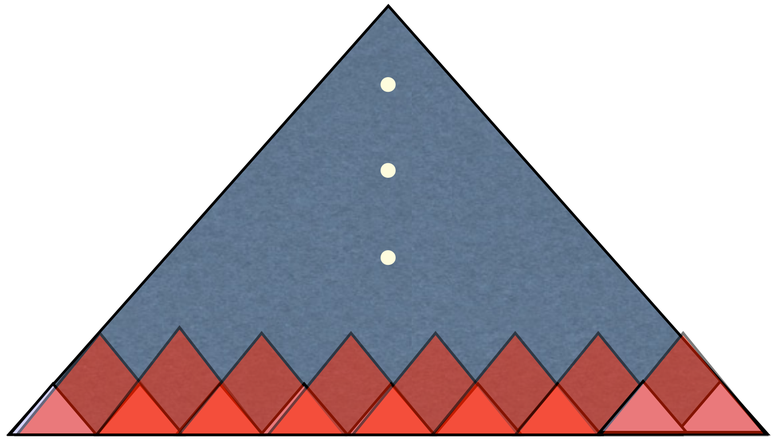


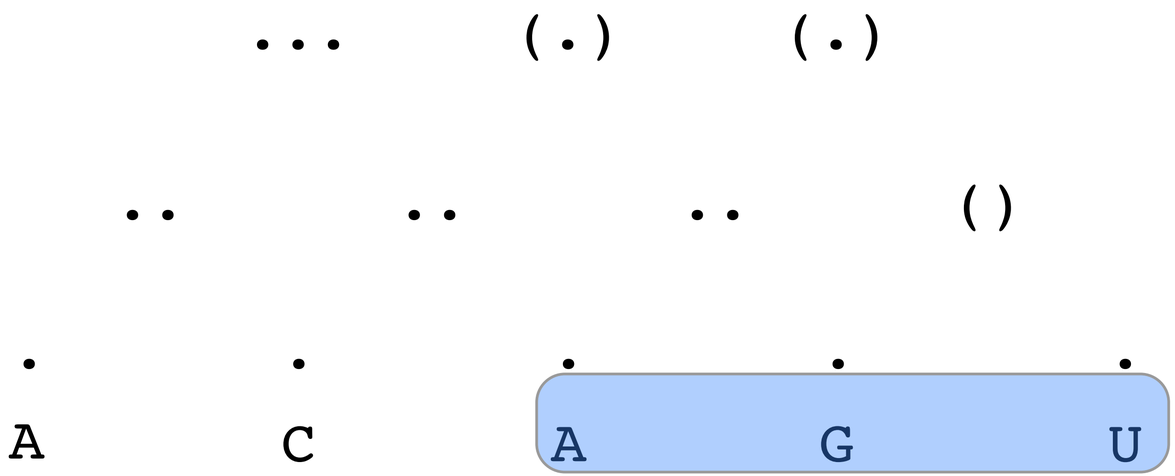
Background: CKY for RNA Folding: $O(n^3)$

- Dynamic Programming — $O(n^3)$
  - bottom-up CKY parsing
  - example: maximize # of pairs (A-U, G-C, or G-U)



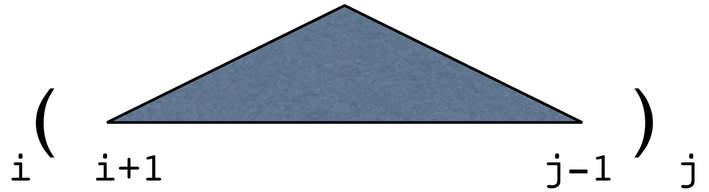


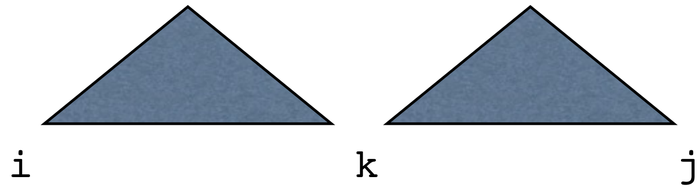


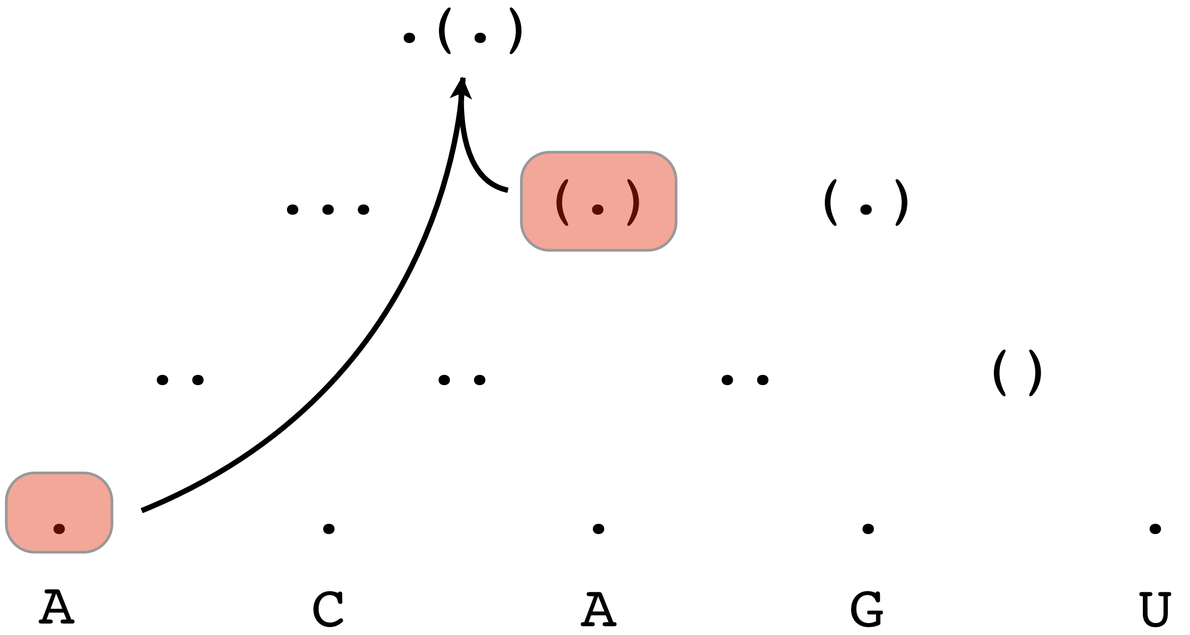


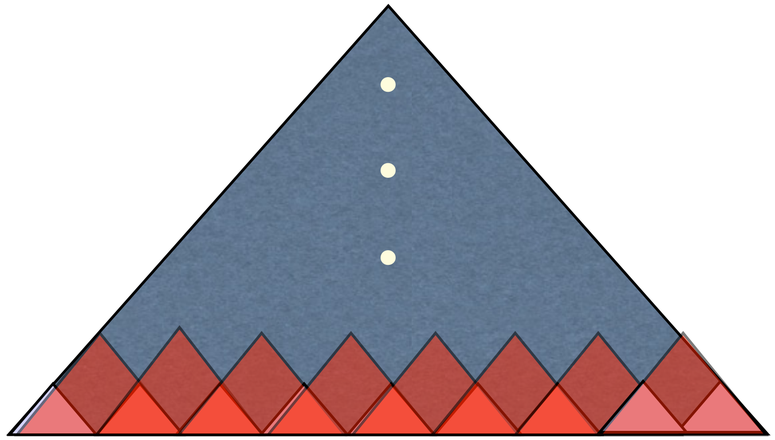
Background: CKY for RNA Folding: $O(n^3)$

- Dynamic Programming — $O(n^3)$
  - bottom-up CKY parsing
  - example: maximize # of pairs (A-U, G-C, or G-U)



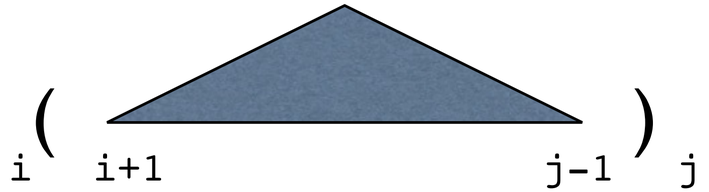


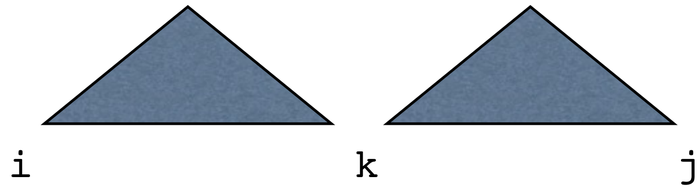


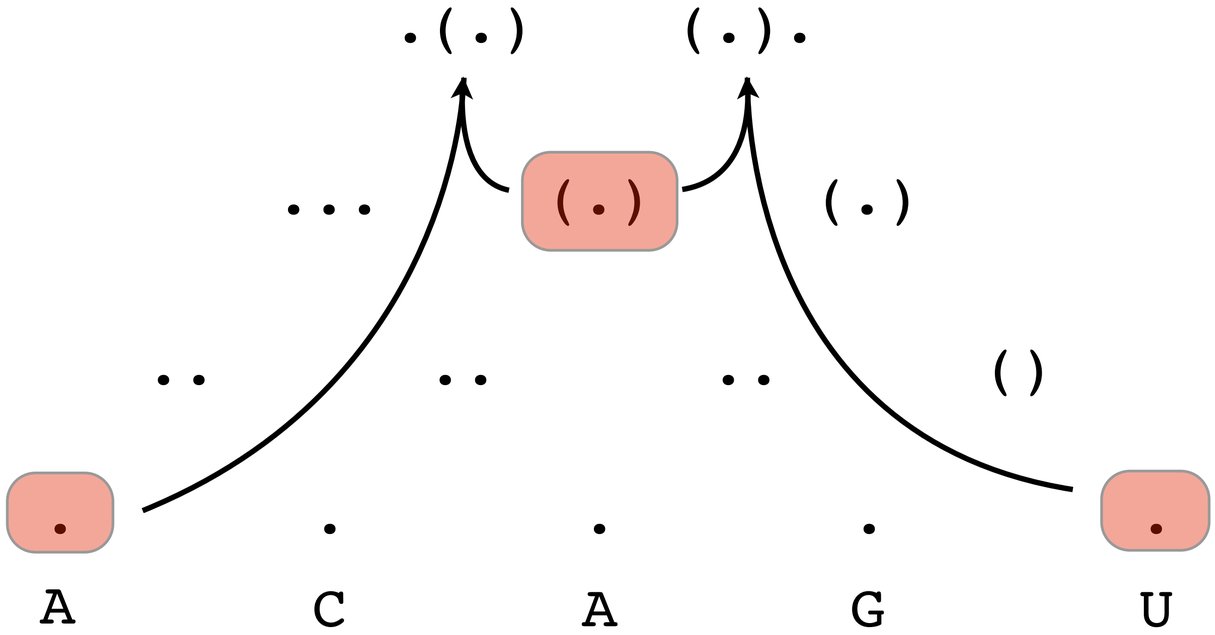


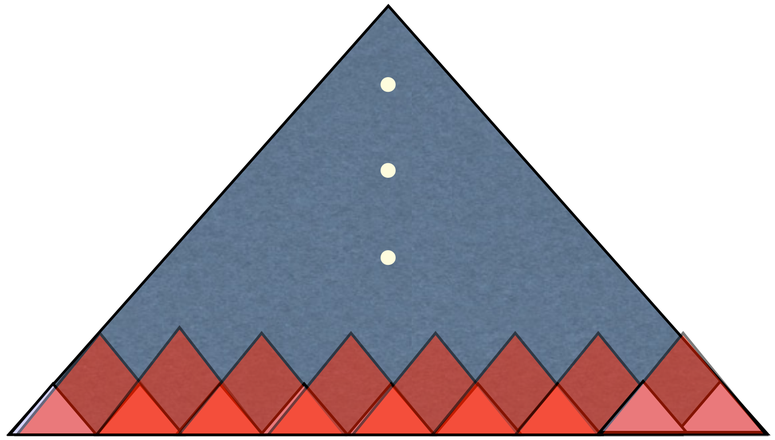
Background: CKY for RNA Folding: $O(n^3)$

- Dynamic Programming — $O(n^3)$
  - bottom-up CKY parsing
  - example: maximize # of pairs (A-U, G-C, or G-U)



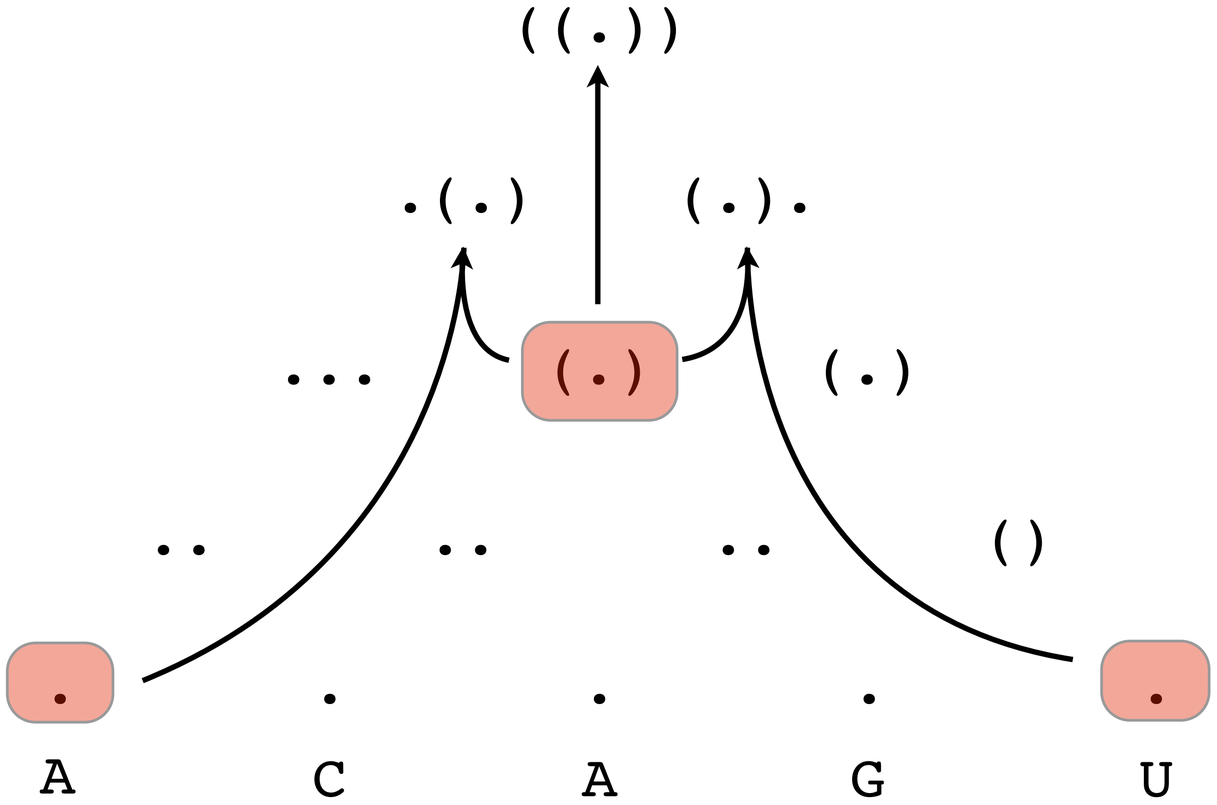


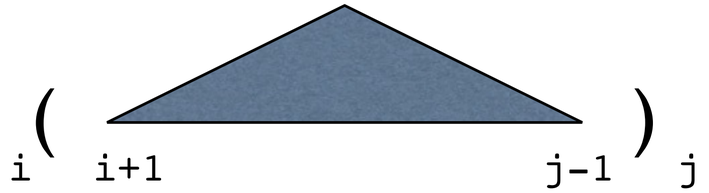


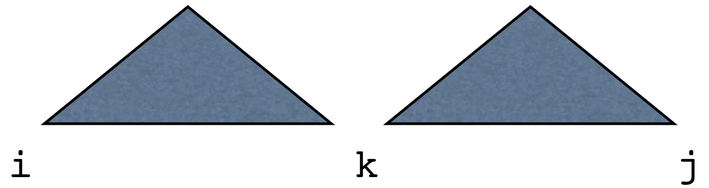


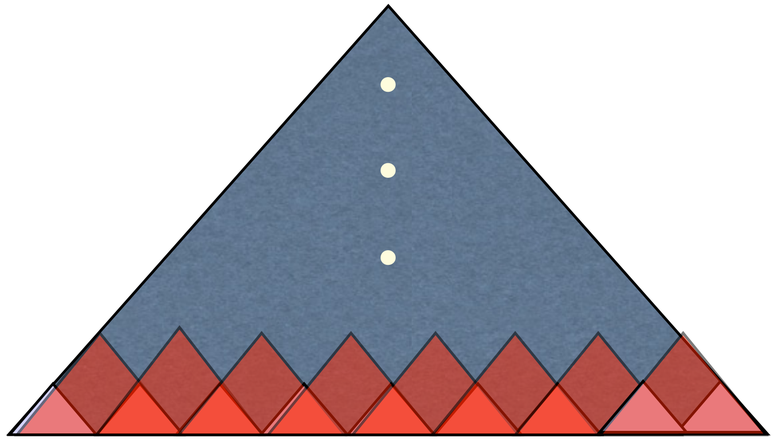
Background: CKY for RNA Folding: $O(n^3)$

- Dynamic Programming — $O(n^3)$
  - bottom-up CKY parsing
  - example: maximize # of pairs (A-U, G-C, or G-U)





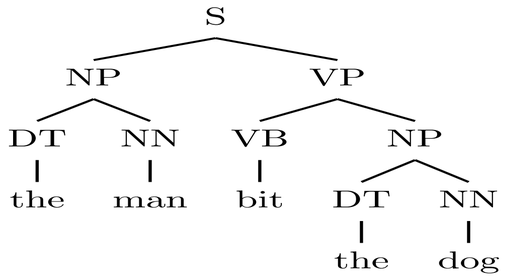




Computational Linguistics => Computational Biology

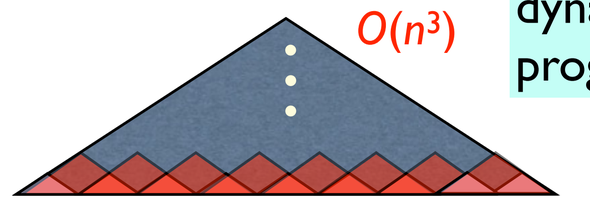
linguistics | **compiler theory** | **comp. linguistics** | **computational biology**

1955 Chomsky: context-free grammars (CFGs)

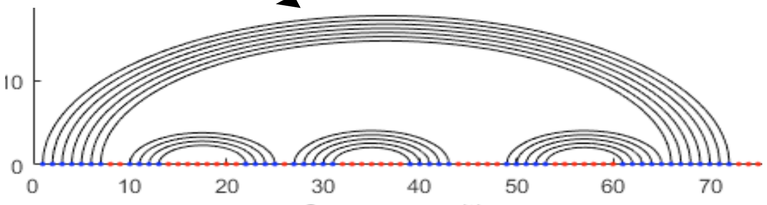


1958 Backus & Naur: CFGs for program, lang.

1964 **C**ocke \ bottom-up
1965 **K**asami - **CKY** $O(n^3)$
1967 **Y**ounger / for all CFGs



dynamic programming

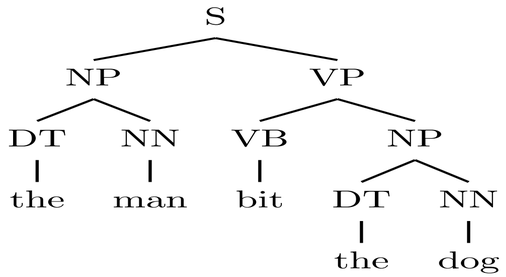


1978: Nussinov $O(n^3)$ RNA folding
1981: Zuker & Siegler

Computational Linguistics => Computational Biology

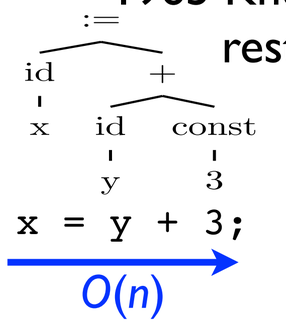
linguistics | **compiler theory** | **comp. linguistics** | **computational biology**

1955 Chomsky: context-free grammars (CFGs)

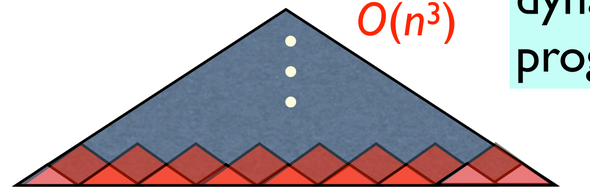


1958 Backus & Naur: CFGs for program, lang.

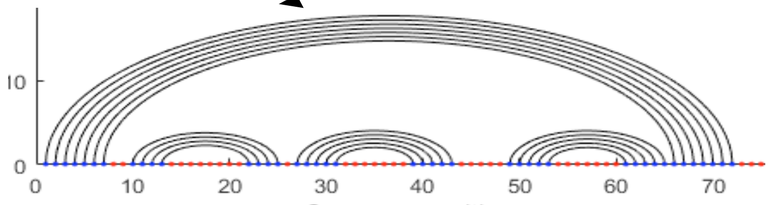
1965 Knuth: **LR** parsing for restricted CFGs: ***O(n)***



1964 **C**ocke \ bottom-up
1965 **K**asami - **CKY** ***O(n³)***
1967 **Y**ounger / for all CFGs



dynamic programming

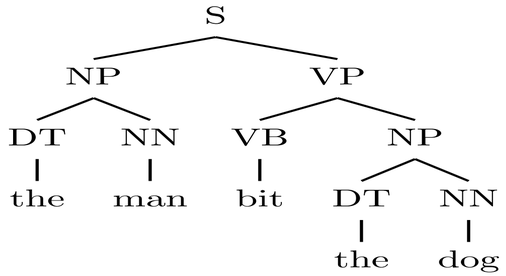


1978: Nussinov ***O(n³)*** RNA folding
1981: Zuker & Siegler

Computational Linguistics => Computational Biology

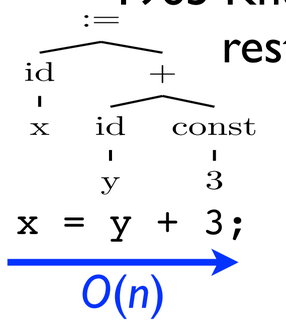
linguistics | **compiler theory** | **comp. linguistics** | **computational biology**

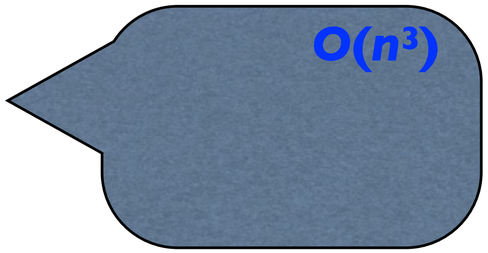
1955 Chomsky: context-free grammars (CFGs)



1958 Backus & Naur: CFGs for program, lang.

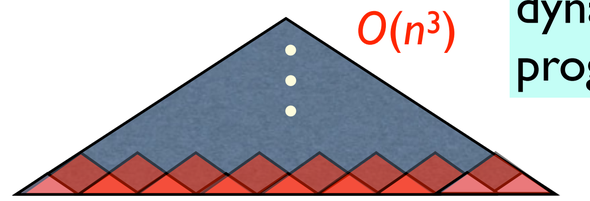
1965 Knuth: **LR** parsing for restricted CFGs: ***O(n)***



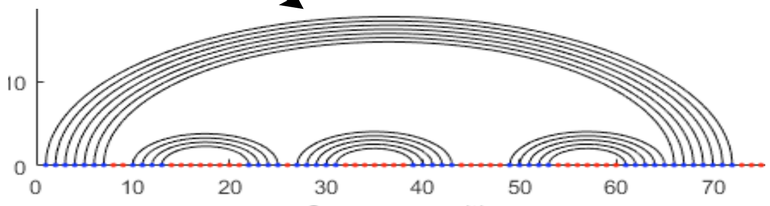


1964 **C**ocke \ bottom-up
1965 **K**asami - **CKY** ***O(n³)***
1967 **Y**ounger / for all CFGs

1986 Tomita: **G**eneralized **LR** for all CFGs: ***O(n³)***



dynamic programming

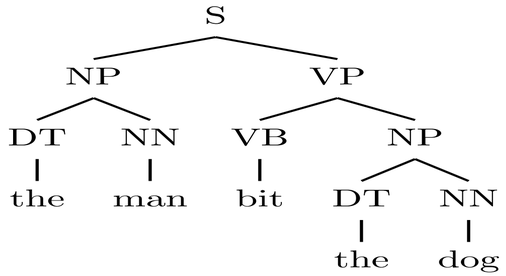


1978: Nussinov ***O(n³)*** RNA folding
1981: Zuker & Siegler

Computational Linguistics => Computational Biology

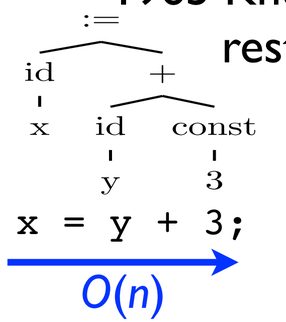
linguistics | **compiler theory** | **comp. linguistics** | **computational biology**

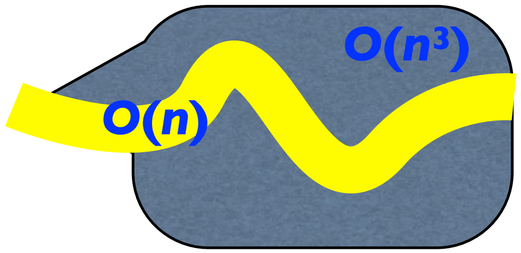
1955 Chomsky: context-free grammars (CFGs)



1958 Backus & Naur: CFGs for program, lang.

1965 Knuth: **LR** parsing for restricted CFGs: ***O(n)***

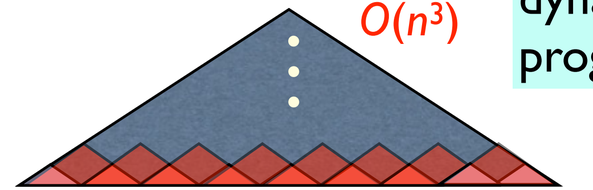




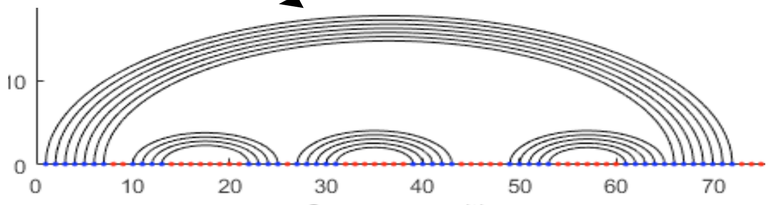
1964 **C**ocke \ bottom-up
1965 **K**asami - **CKY** ***O(n³)***
1967 **Y**ounger / for all CFGs

1986 Tomita: **G**eneralized **LR** for all CFGs: ***O(n³)***

2010: Huang & Sagae: ***O(n)*** (approx.) DP for all CFGs



dynamic programming

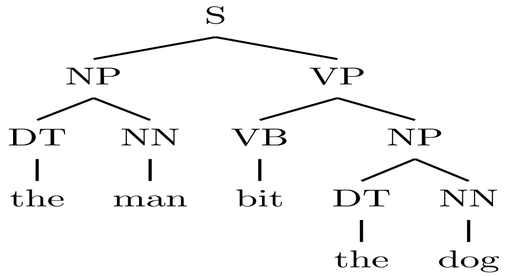


1978: Nussinov ***O(n³)*** RNA folding
1981: Zuker & Siegler

Computational Linguistics => Computational Biology

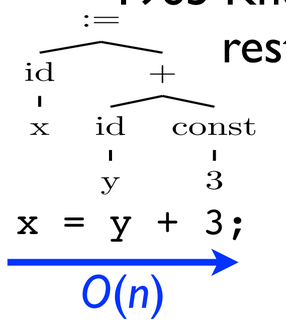
linguistics | **compiler theory** | **comp. linguistics** | **computational biology**

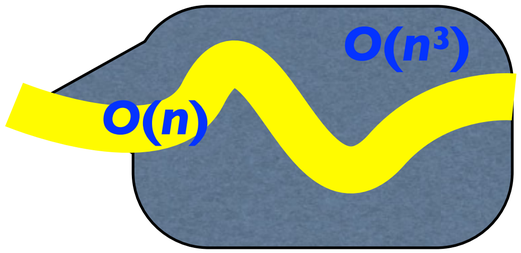
1955 Chomsky: context-free grammars (CFGs)



1958 Backus & Naur: CFGs for program, lang.

1965 Knuth: **LR** parsing for restricted CFGs: ***O(n)***

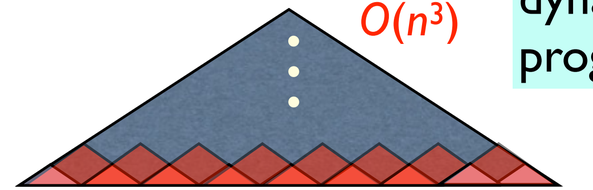




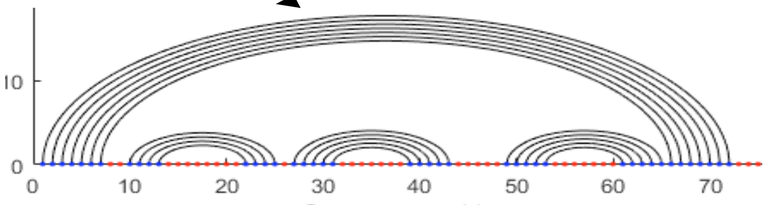
1964 **C**ocke \ bottom-up
1965 **K**asami - **CKY** ***O(n³)***
1967 **Y**ounger / for all CFGs

1986 Tomita: **G**eneralized **LR** for all CFGs: ***O(n³)***

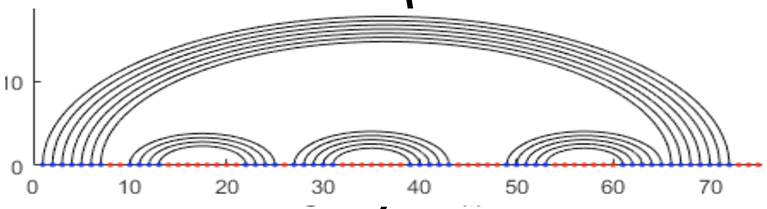
2010: Huang & Sagae: ***O(n)*** (approx.) DP for all CFGs



dynamic programming



1978: Nussinov ***O(n³)*** RNA folding
1981: Zuker & Siegler

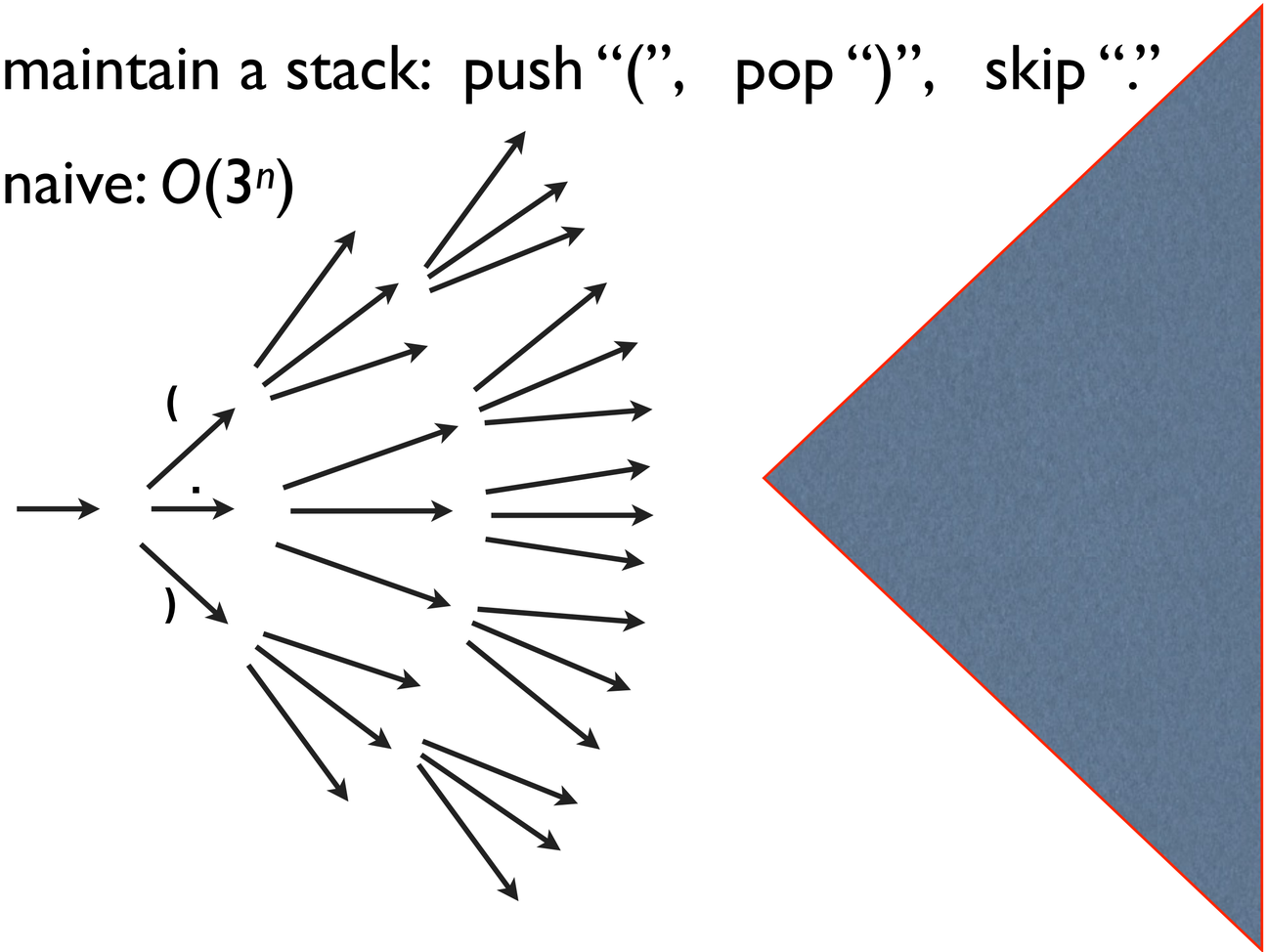


2019: LinearFold O(n) (approx.) RNA folding

How to Fold RNAs in Linear-Time?

GCGGGAAUAGCUCAGUUGGUAGAGCACGACCUUGCCAAGGUCGGGGUCGCGAGUUCGAGUCUCGUUUCCCGCUCCA

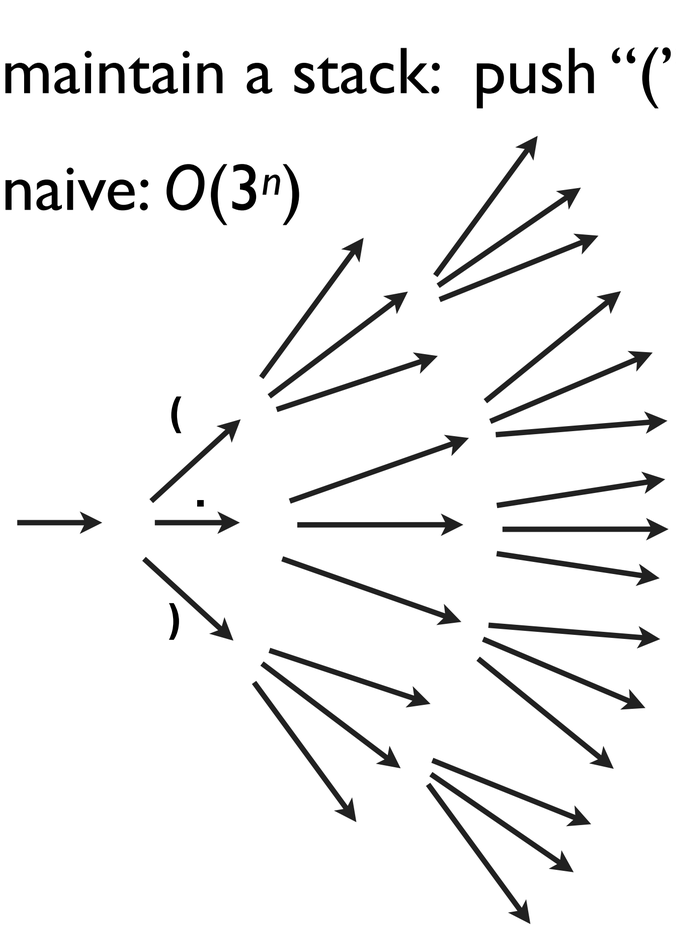
- idea 0: tag each nucleotide from left to right
  - maintain a stack: push "(", pop ")", skip "."
  - naive: $O(3^n)$

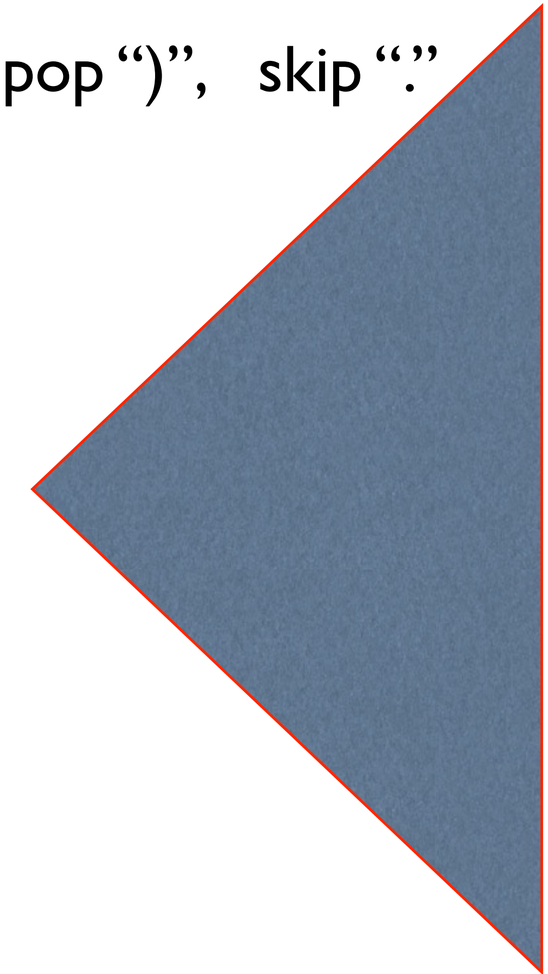


How to Fold RNAs in Linear-Time?

```
GCGGGAAUAGCUCAGUUGGUAGAGCACGACCUUGCCAAGGUCGGGGUCGCGAGUUCGAGUCUCGUUUCCCGCUCCA
((((((( . . ((((. . . . . . . )))) . ((((. . . . . . )))) . . . . ((((. . . . . . )))))))))))) . . . .
```

- idea 0: tag each nucleotide from left to right
  - maintain a stack: push "(", pop ")", skip "."
  - naive: $O(3^n)$

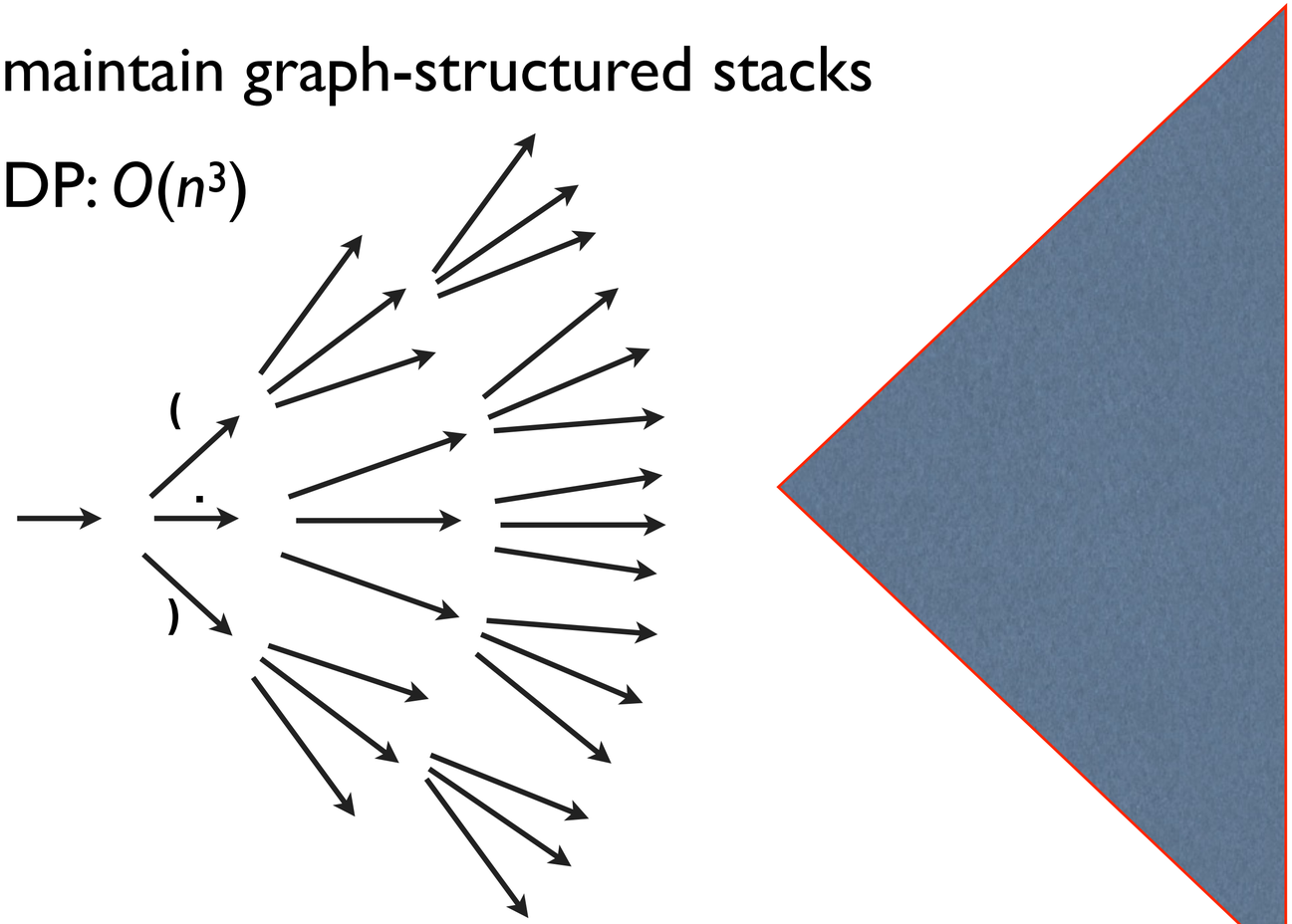




How to Fold RNAs in Linear-Time?

```
GCGGGAAUAGCUCAGUUGGUAGAGCACGACCUUGCCAAGGUCGGGGUCGCGAGUUCGAGUCUCGUUUCCCGCUCCA
((((((((..((((.........)))).((((.........))))).....((((.........))))))))))....
```

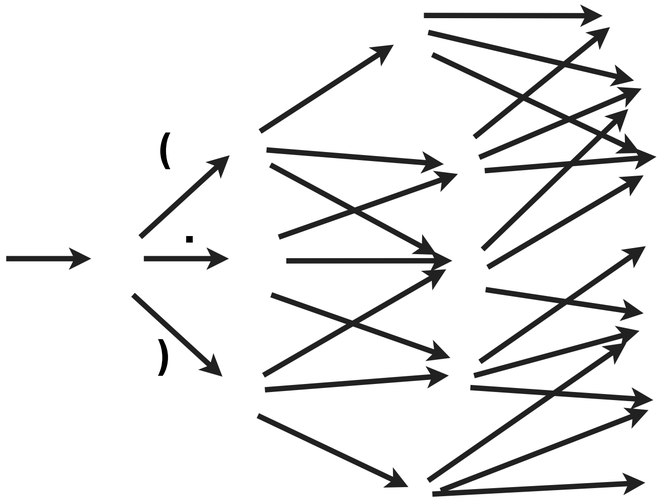
- idea 1: DP by packing "equivalent states"
  - maintain graph-structured stacks
  - DP: $O(n^3)$

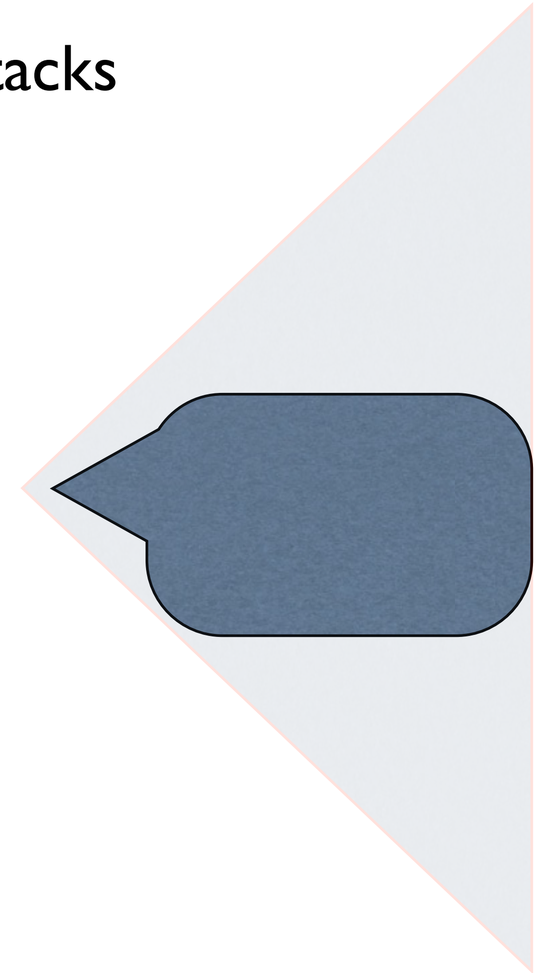


How to Fold RNAs in Linear-Time?

```
GCGGGAAUAGCUCAGUUGGUAGAGCACGACCUUGCCAAGGUCGGGGUCGCGAGUUCGAGUCUCGUUUCCCGCUCCA
(((((((..(((((.....)))))..(((((.....)))))).....(((((.....)))))))))))....
```

- idea 1: DP by packing "equivalent states"
  - maintain graph-structured stacks
  - DP: $O(n^3)$

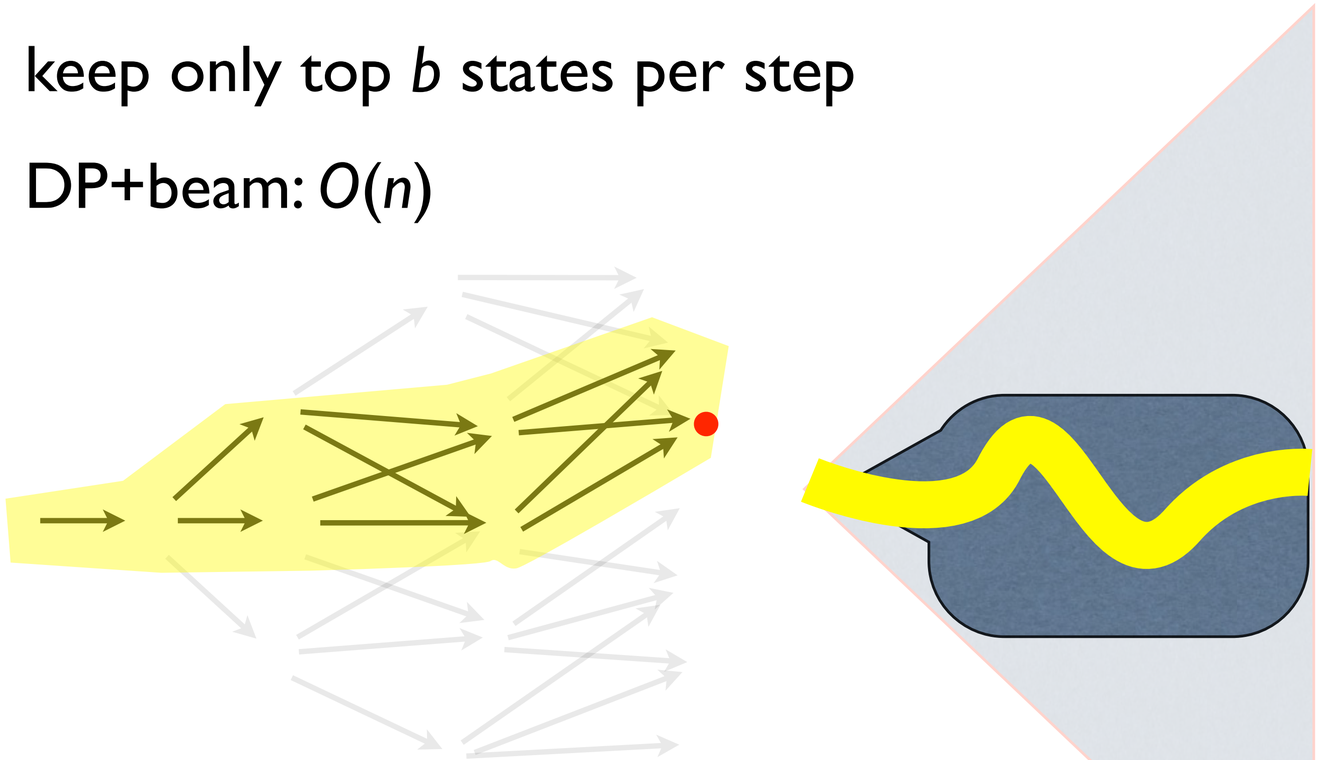




How to Fold RNAs in Linear-Time?

```
GCGGGAAUAGCUCAGUUGGUAGAGCACGACCUUGCCAAGGUCGGGGUCGCGAGUUCGAGUCUCGUUUCCCGCUCCA
((((((((..((((.........)))).((((.........))))).....((((.........))))))))))....
```

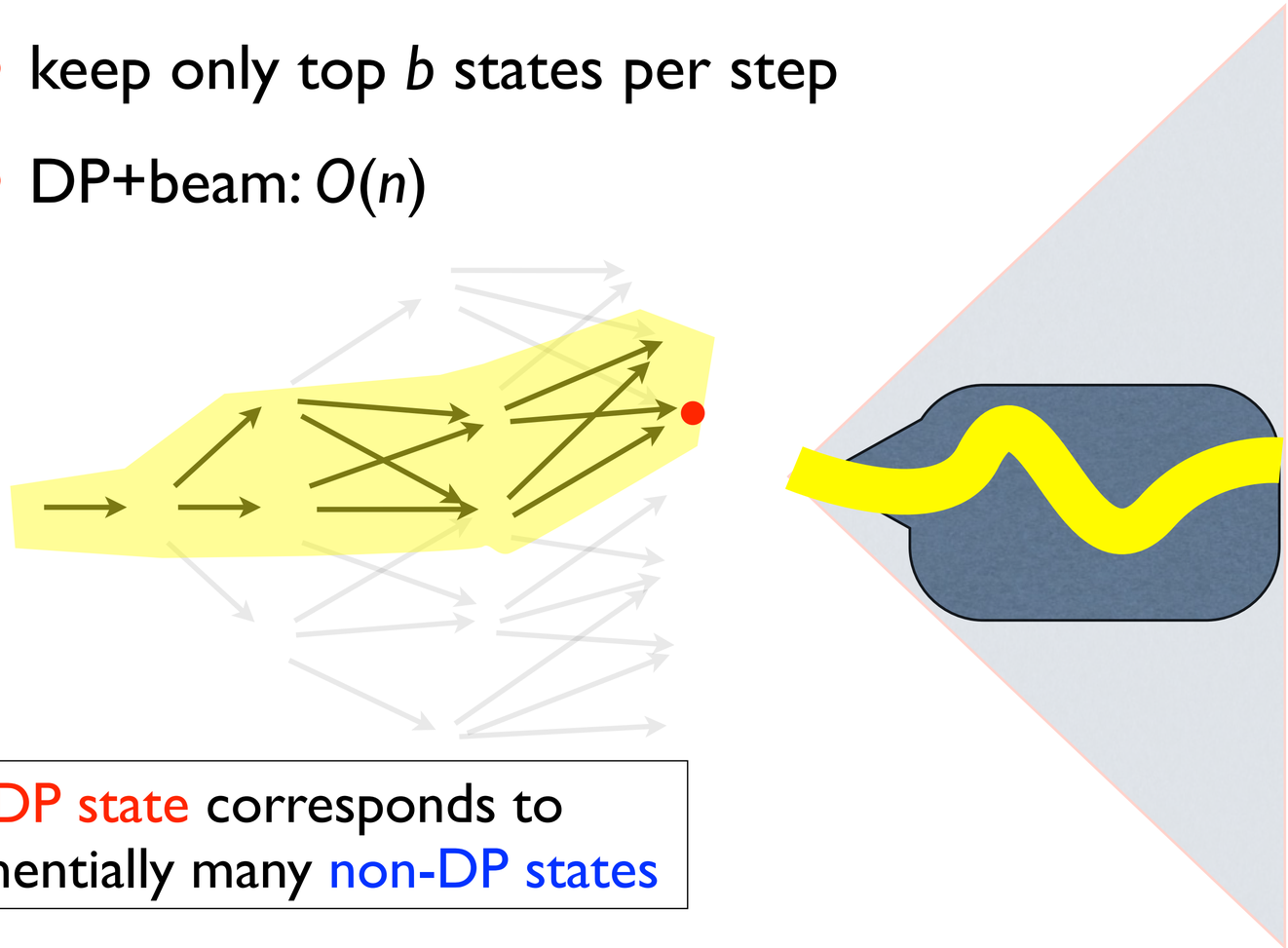
- idea 2: approximate search: beam pruning
  - keep only top *b* states per step
  - DP+beam: *O*(*n*)



How to Fold RNAs in Linear-Time?

```
GCGGGAAUAGCUCAGUUGGUAGAGCACGACCUUGCCAAGGUCGGGGUCGCGAGUUCGAGUCUCGUUUCCCGCUCCA
((((((( . . ((((. . . . . . . )))) . ((((. . . . . . )))) . . . . ((((. . . . . . ))))))))))))) . . . .
```

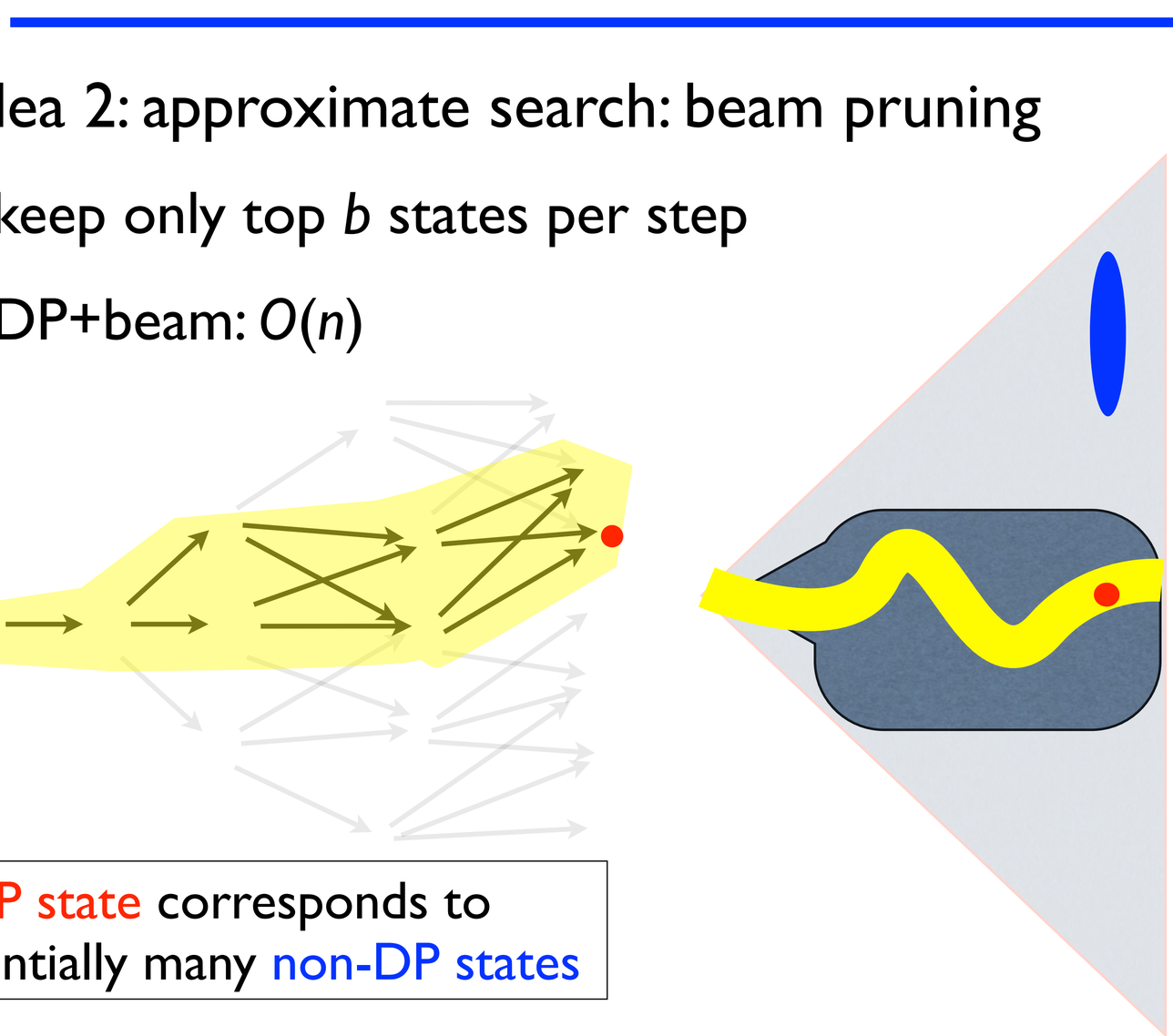
- idea 2: approximate search: beam pruning
  - keep only top *b* states per step
  - DP+beam: *O*(*n*)



each DP state corresponds to exponentially many non-DP states

How to Fold RNAs in Linear-Time?

```
GCGGGAAUAGCUCAGUUGGUAGAGCACGACCUUGCCAAGGUCGGGGUCGCGAGUUCGAGUCUCGUUUCCCGCUCCA
((((((( . . ((((. . . . . . . )))) . ((((. . . . . . )))) . . . . ((((. . . . . . )))))))))))) . . . .
```



- idea 2: approximate search: beam pruning
  - keep only top *b* states per step
  - DP+beam: *O*(*n*)

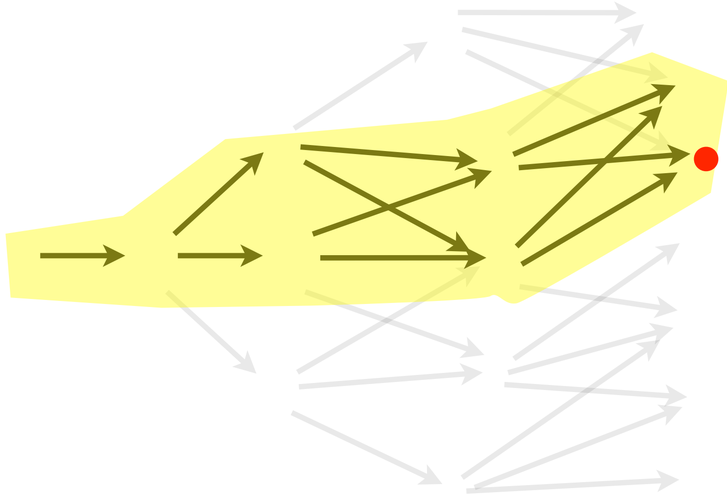
each DP state corresponds to exponentially many non-DP states

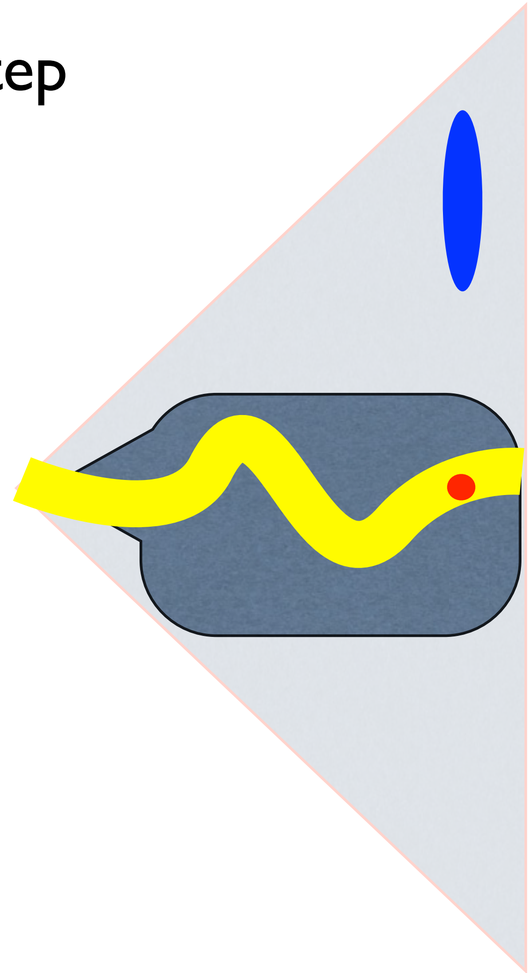
How to Fold RNAs in Linear-Time?

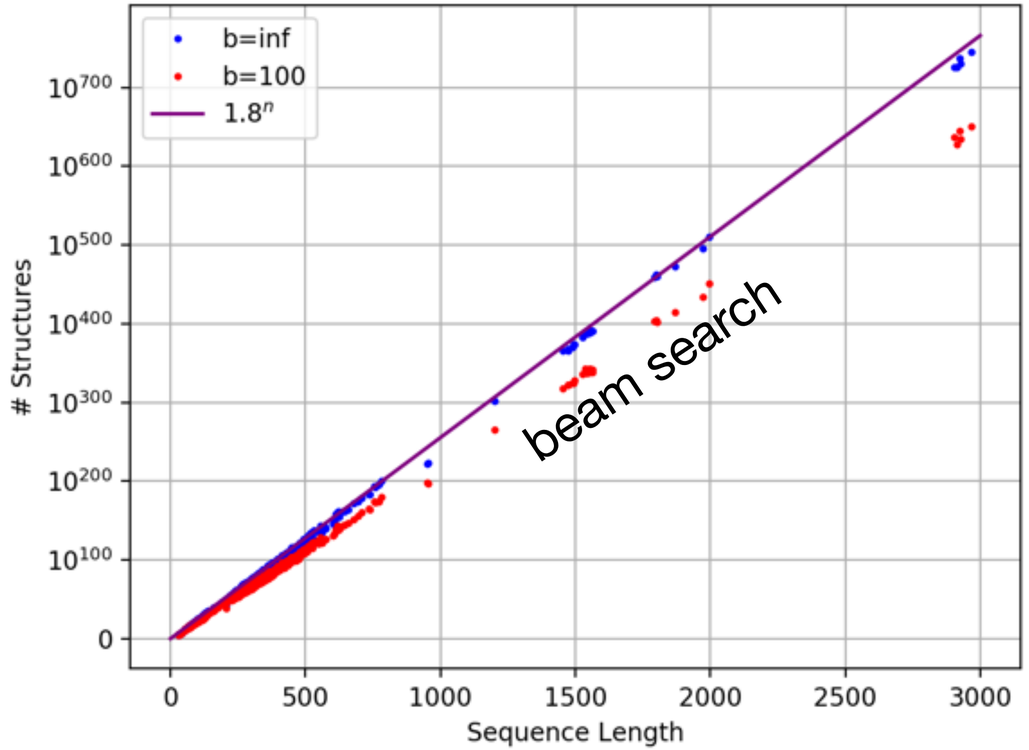
GCGGGAAUAGCUCAGUUGGUAGAGCACGACCUUGCCAAGGUCGGGGUCGCGAGUUCGAGUCUCGUUUCCCGCUCCA

(((((((. . ((((. . . . .)))) . ((((. . . . .)))) . . . . ((((. . . . .)))))))))))) . . .

- idea 2: approximate search: beam pruning
  - keep only top *b* states per step
  - DP+beam: $O(n)$







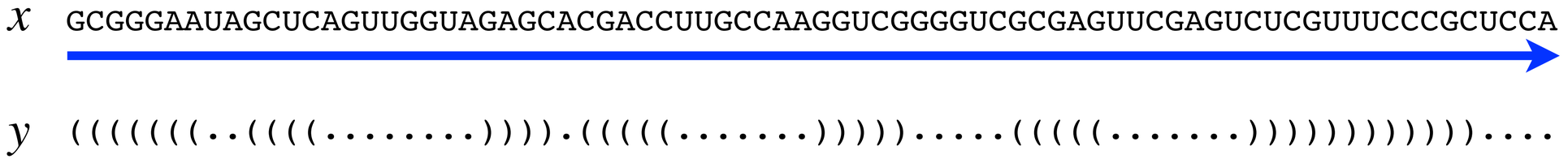
each DP state corresponds to exponentially many non-DP states

On to details...

x GCGGGAAUAGCUCAGUUGGUAGAGCACGACCUUGCCAAGGUCGGGGUCGCGAGUUCGAGUCUCGUUUCCCGCUCCA

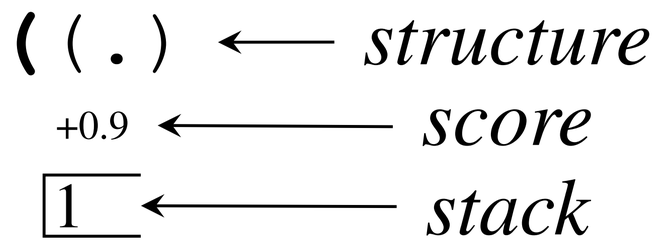
y ((((((((..((((........))))).((((.......)))).....((((.......))))))))))).....

On to details...



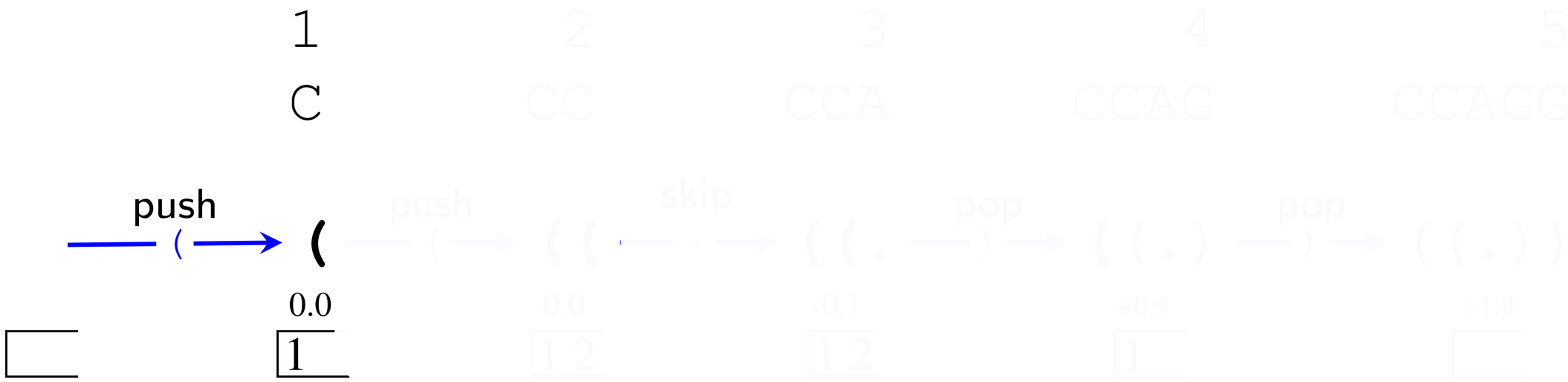
An Example (Optimal) Path

Example state:



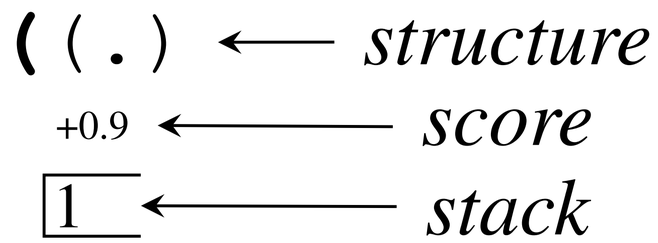
Nussinov-style scoring model:
+1.0 for each pair
-0.1 for each unpaired nucleotide

Example path:



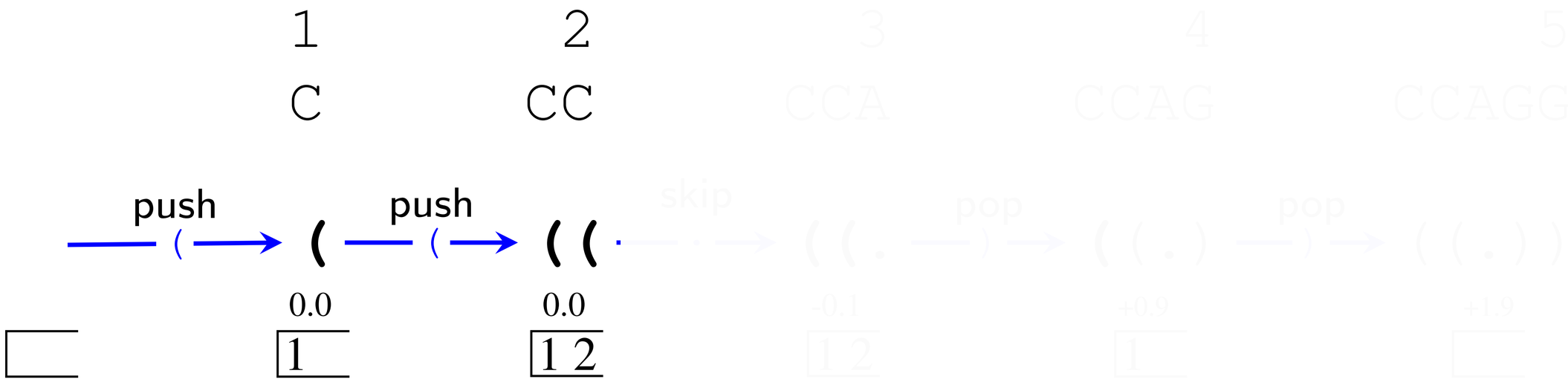
An Example (Optimal) Path

Example state:



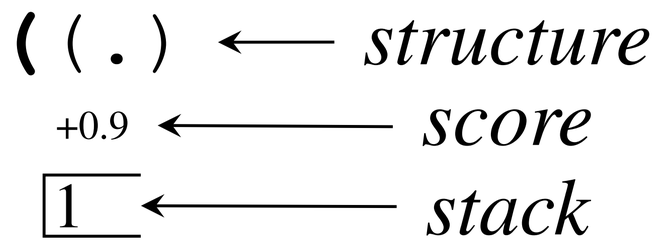
Nussinov-style scoring model:
+1.0 for each pair
-0.1 for each unpaired nucleotide

Example path:



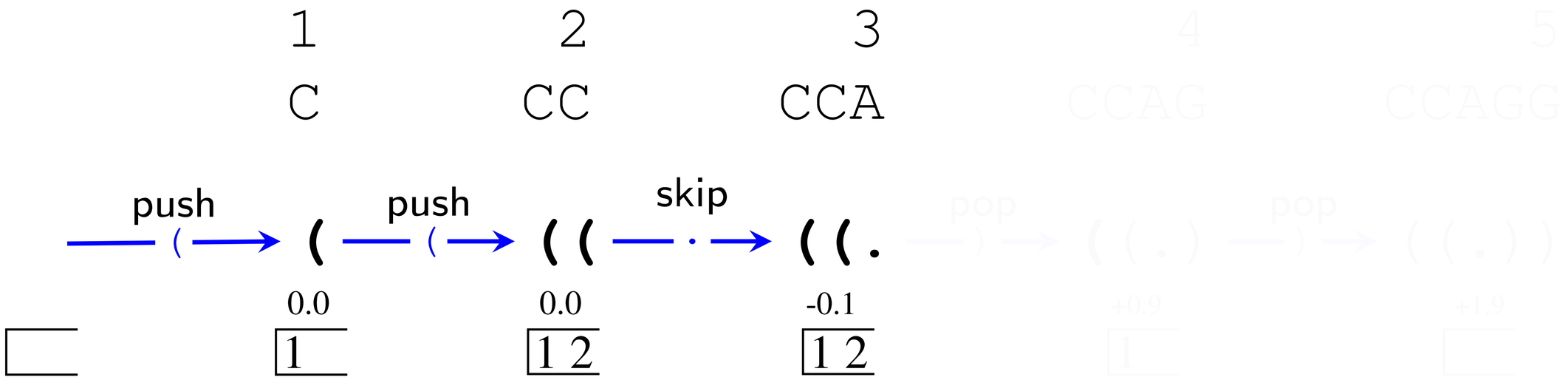
An Example (Optimal) Path

Example state:



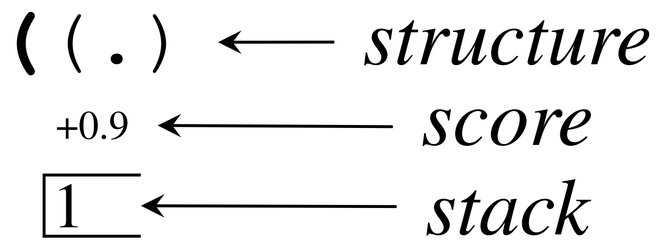
Nussinov-style scoring model:
+1.0 for each pair
-0.1 for each unpaired nucleotide

Example path:



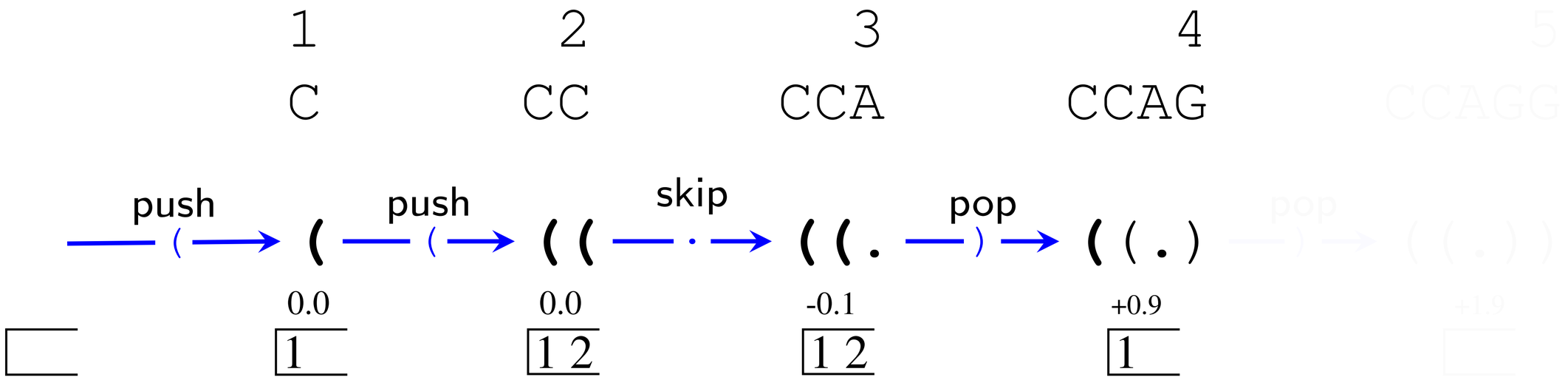
An Example (Optimal) Path

Example state:



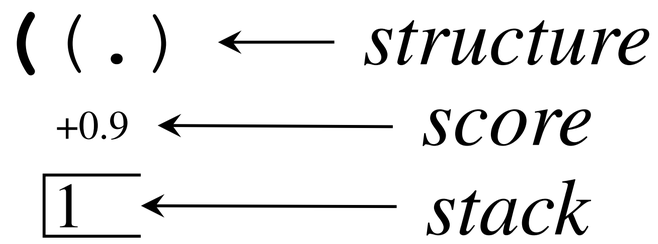
Nussinov-style scoring model:
+1.0 for each pair
-0.1 for each unpaired nucleotide

Example path:



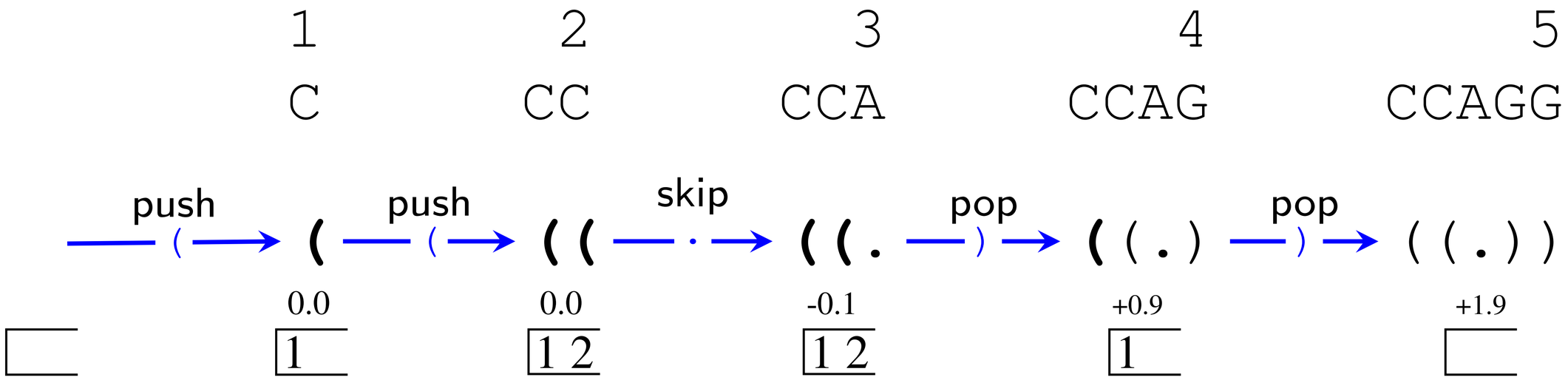
An Example (Optimal) Path

Example state:

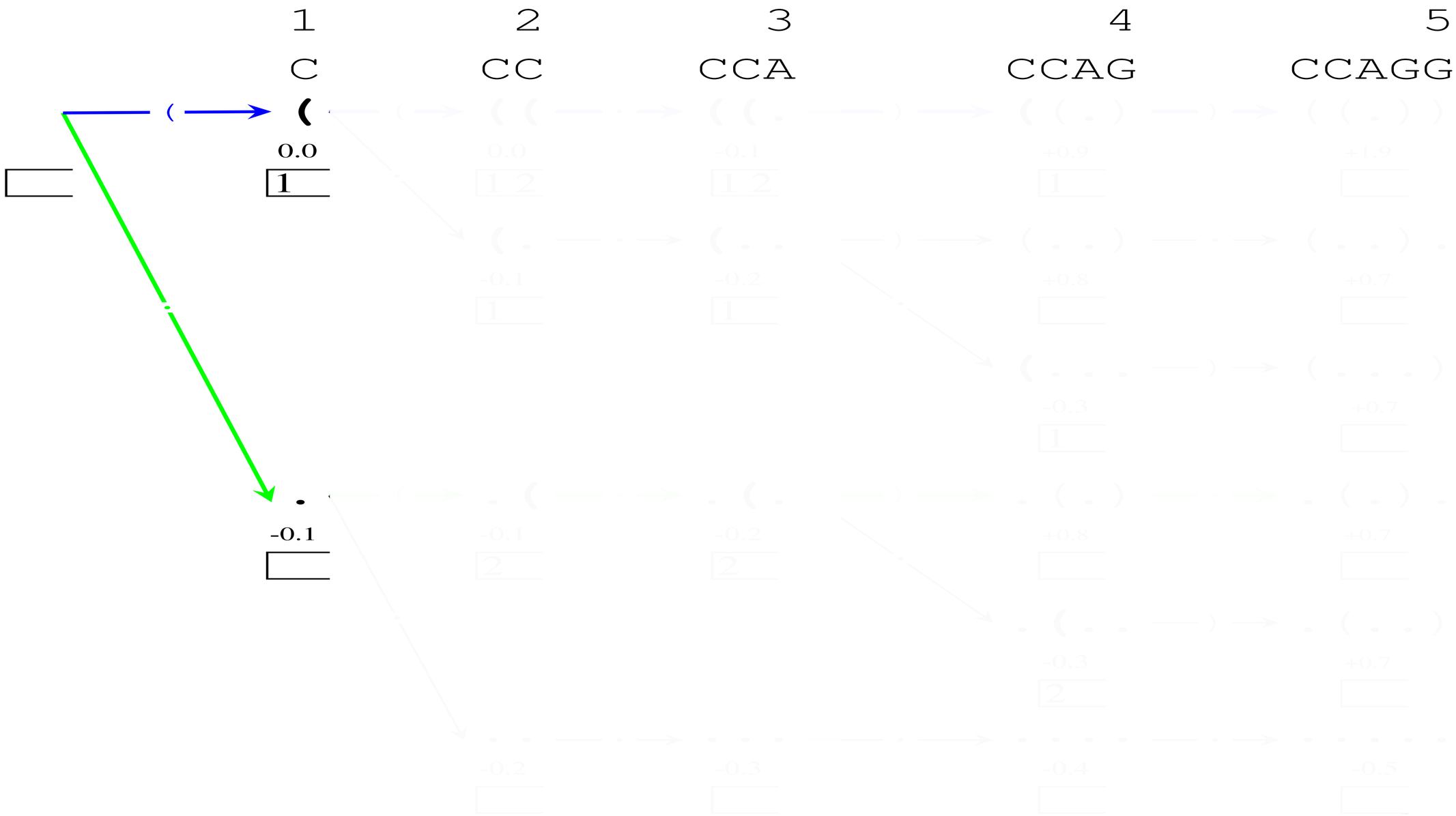


Nussinov-style scoring model:
+1.0 for each pair
-0.1 for each unpaired nucleotide

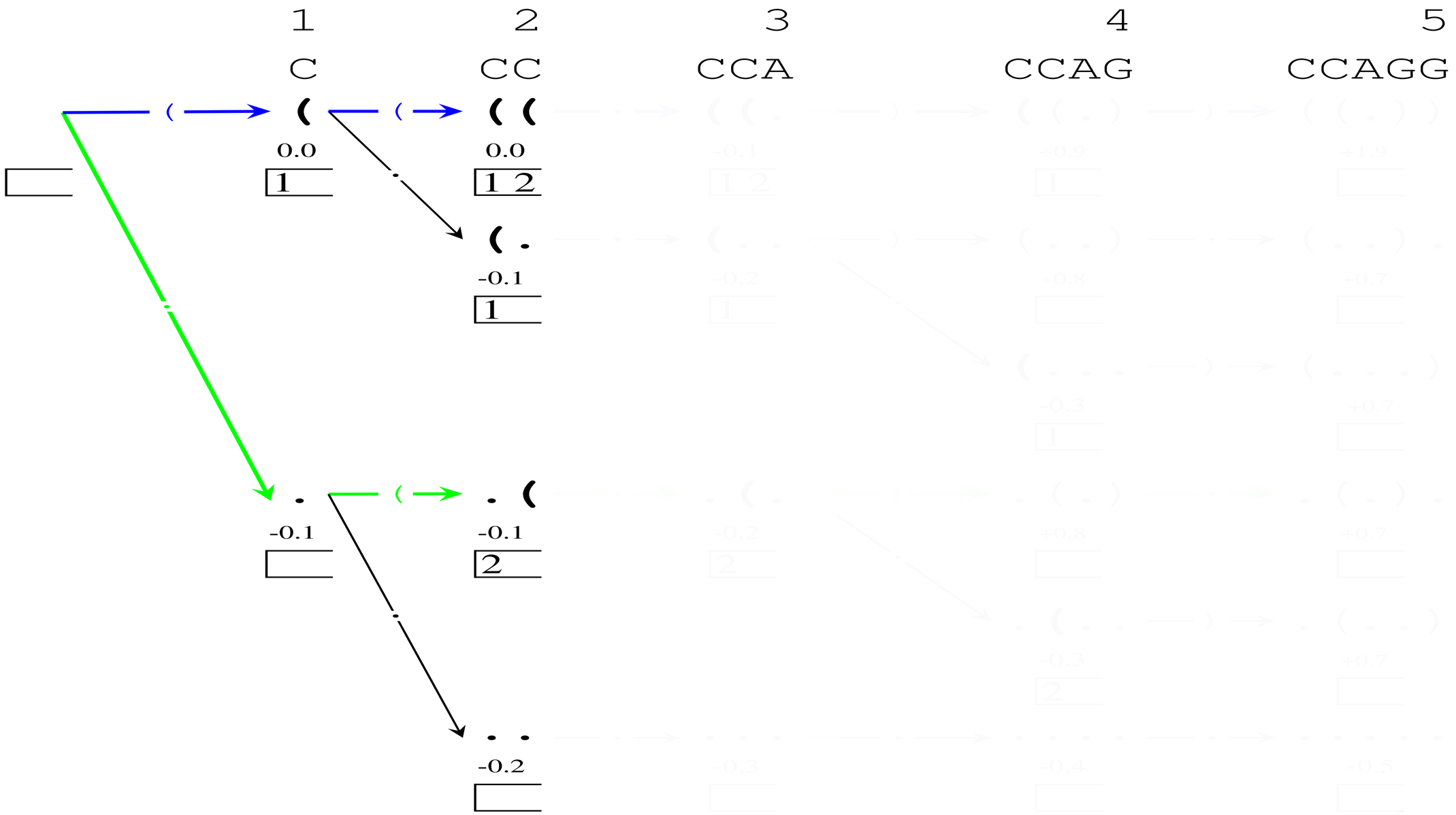
Example path:



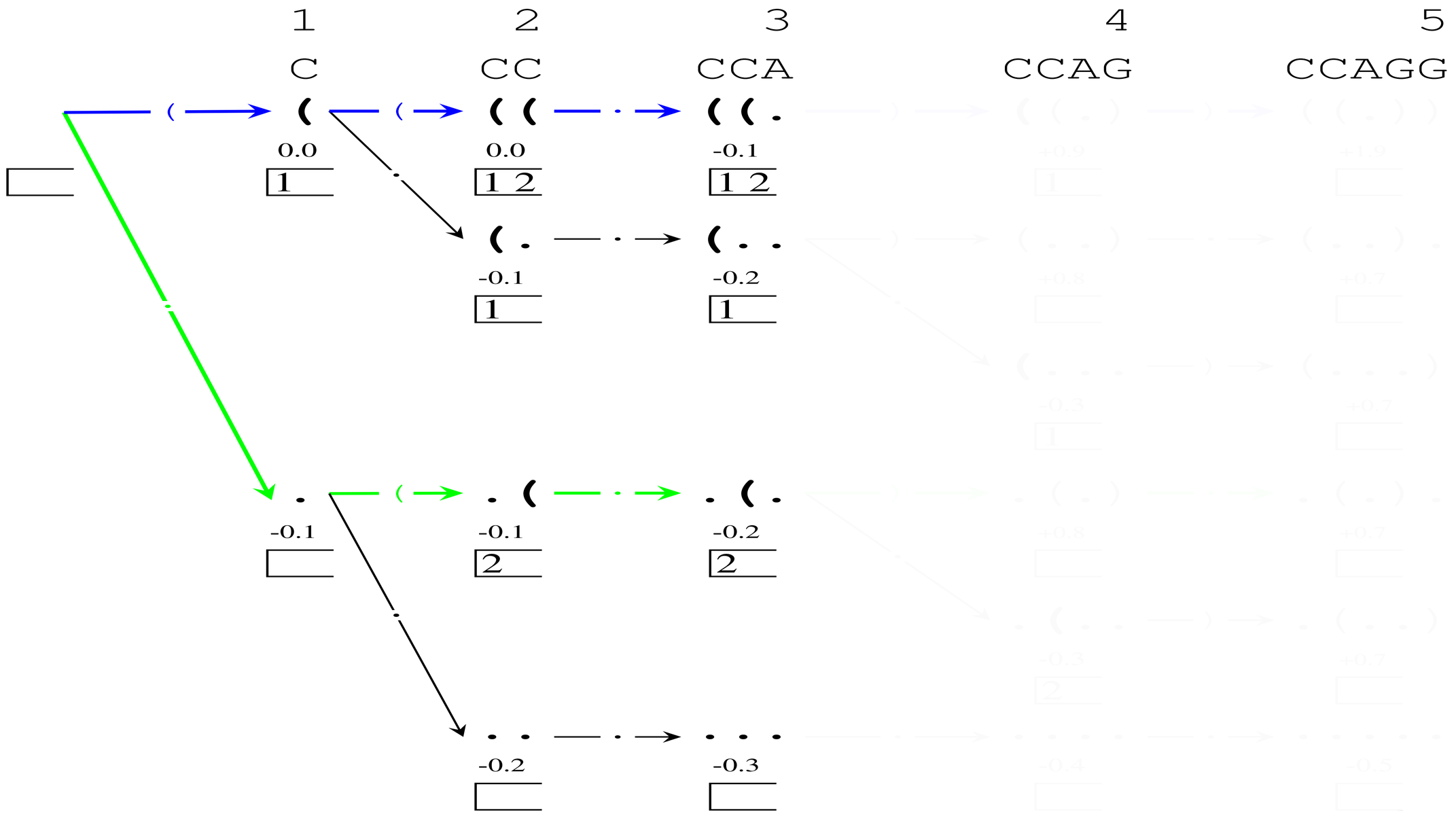
Version 1: Naive Exhaustive Search: $O(3^n)$



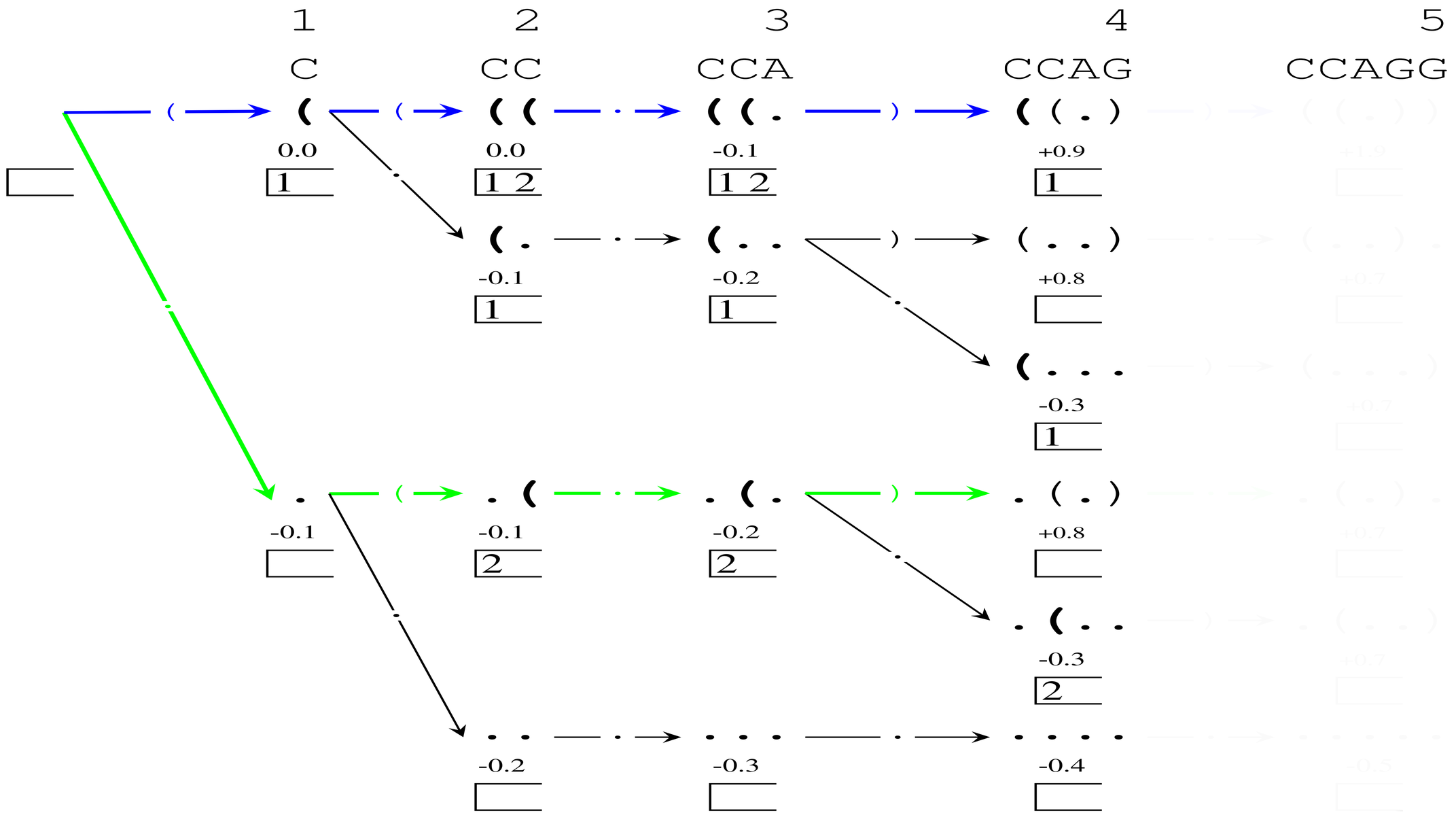
Version 1: Naive Exhaustive Search: $O(3^n)$



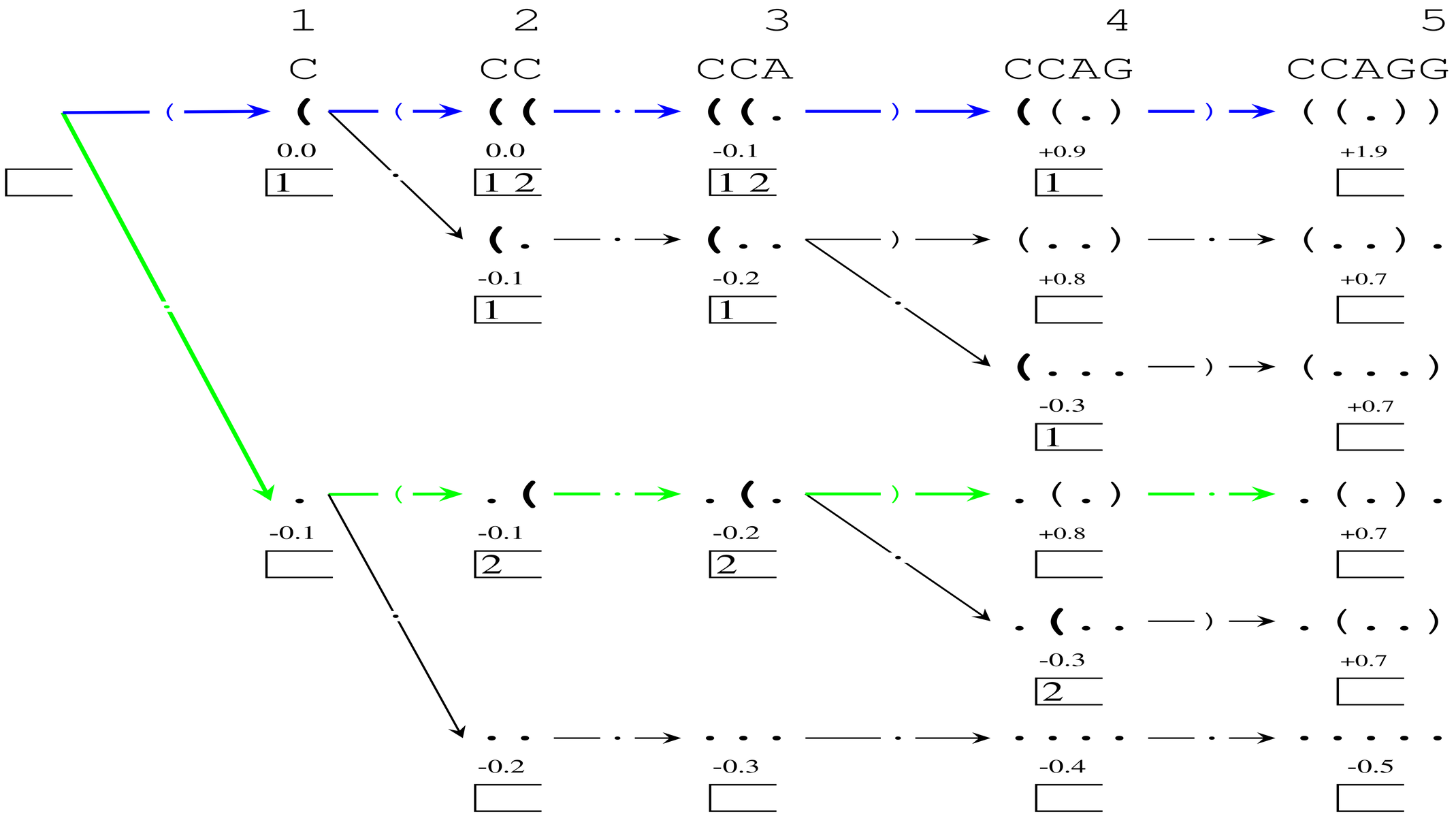
Version 1: Naive Exhaustive Search: $O(3^n)$



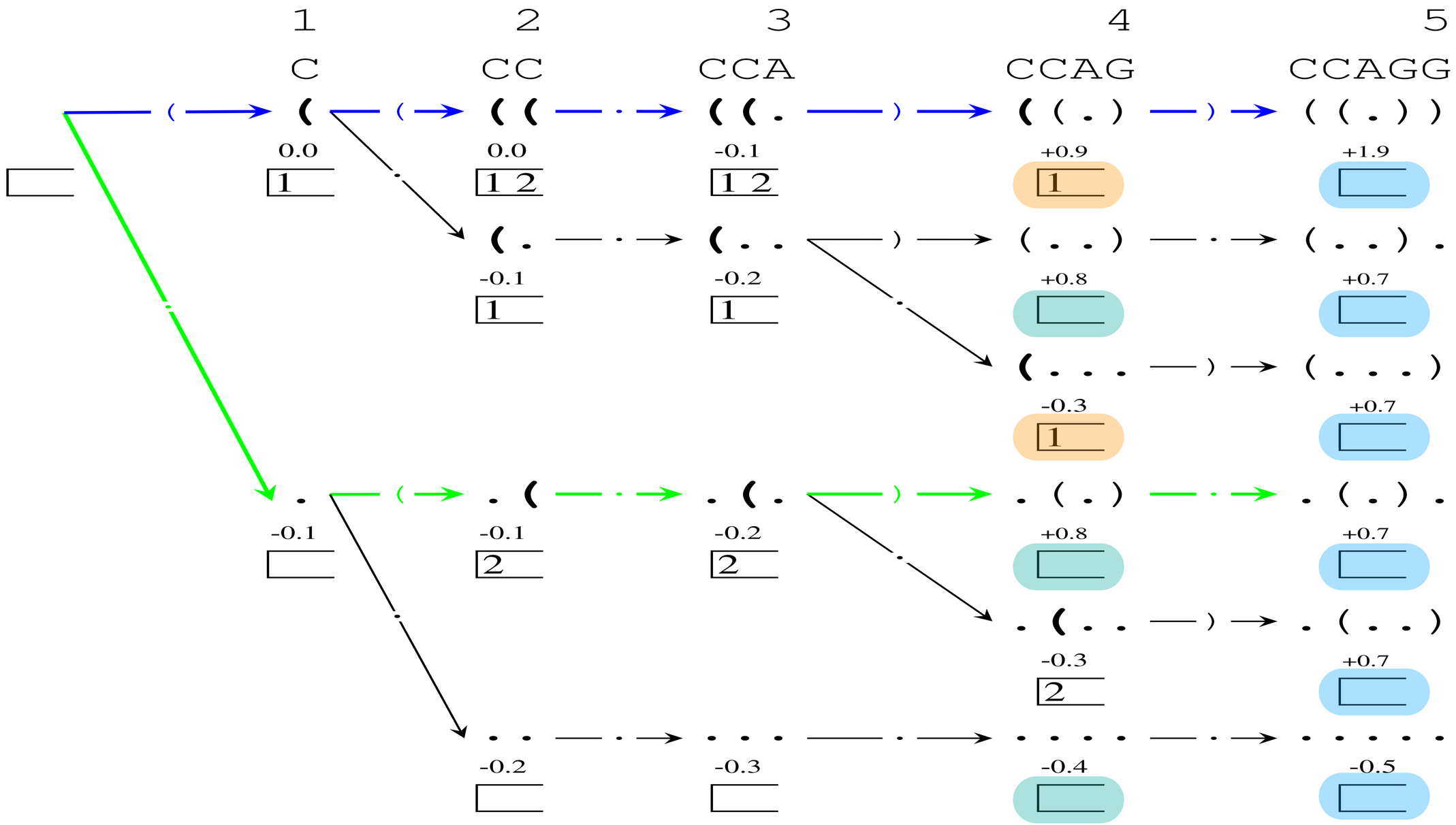
Version 1: Naive Exhaustive Search: $O(3^n)$



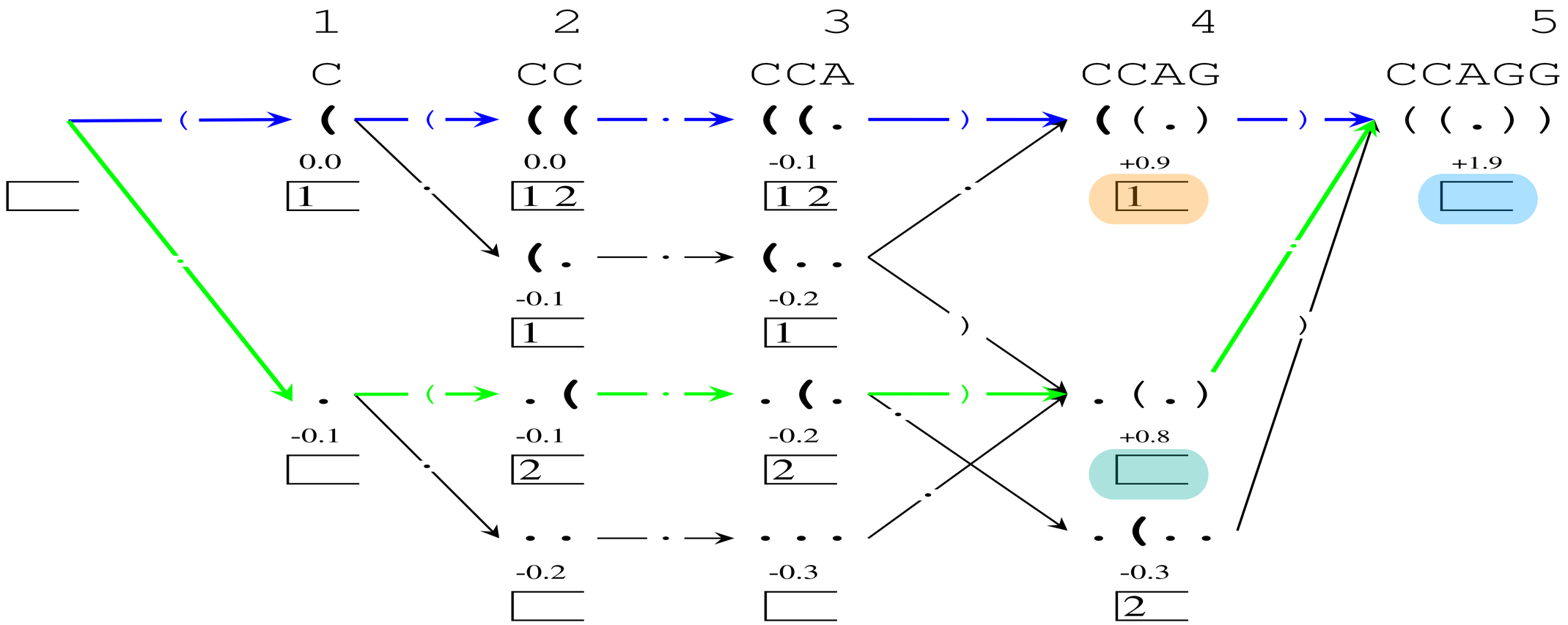
Version 1: Naive Exhaustive Search: $O(3^n)$



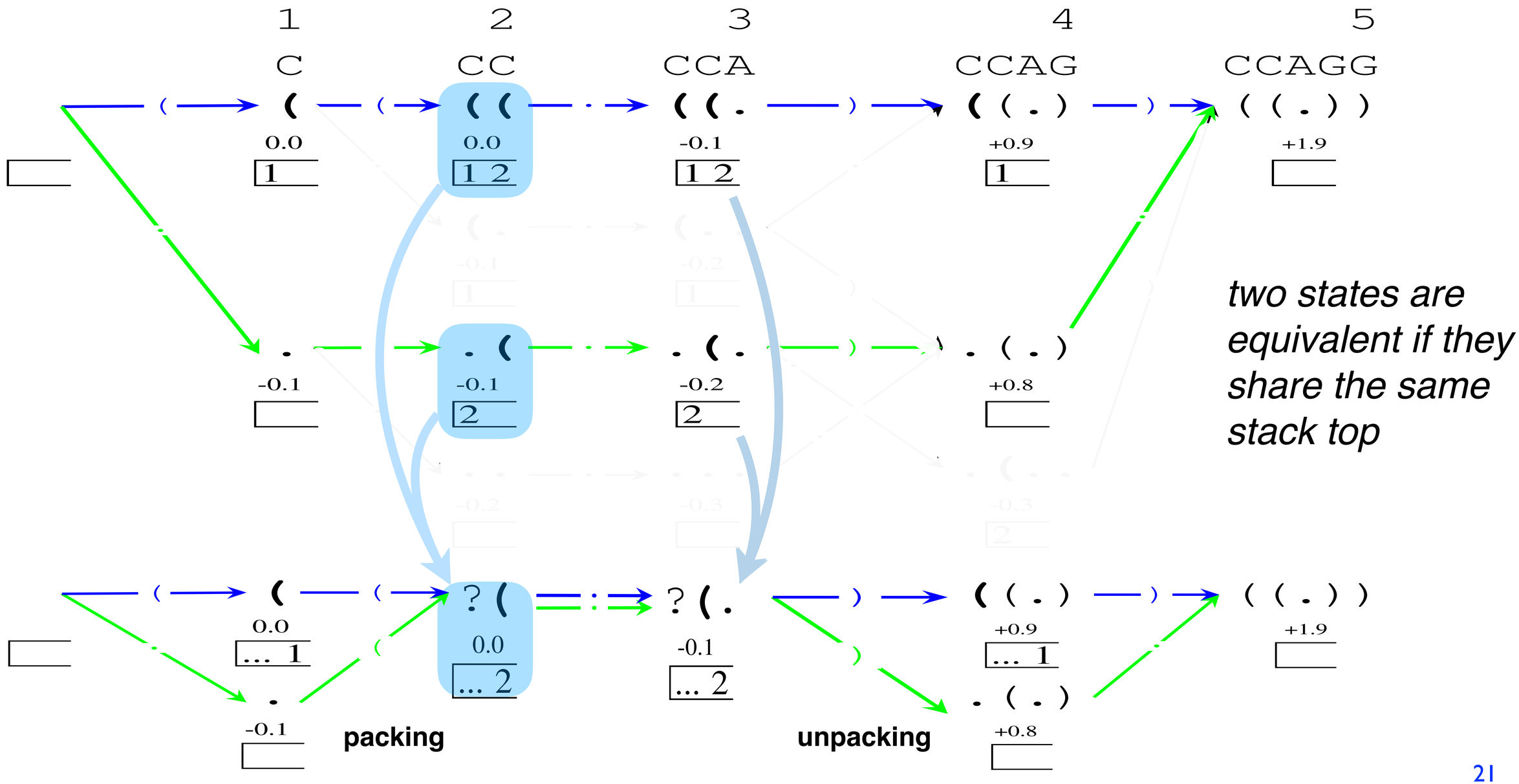
Idea 1: Merge Identical Stacks => Version 2: $O(2^n)$



Idea 1: Merge Identical Stacks => Version 2: $O(2^n)$

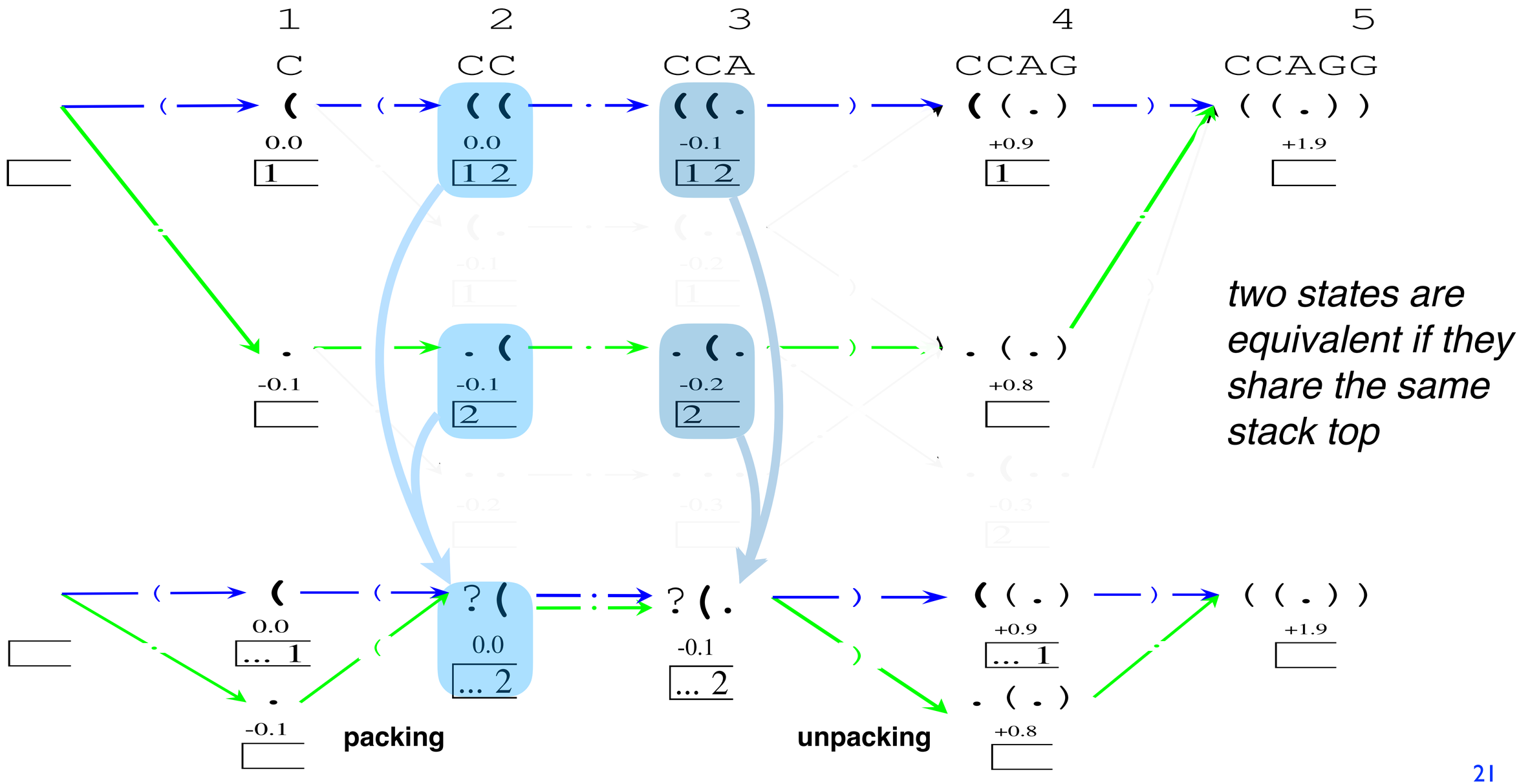


Details: Pack "Equivalent" States => $O(n^3)$



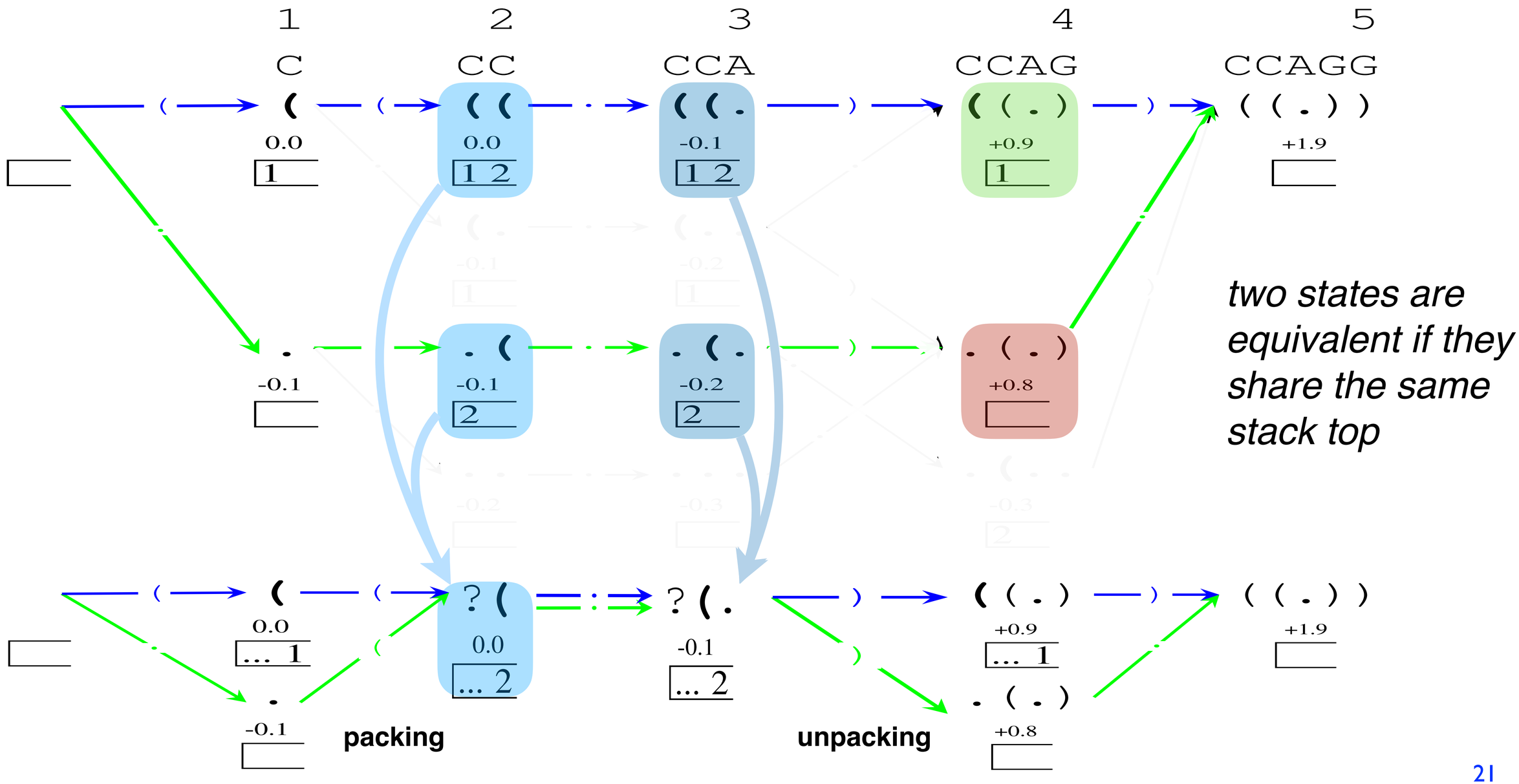
two states are equivalent if they share the same stack top

Details: Pack "Equivalent" States => $O(n^3)$



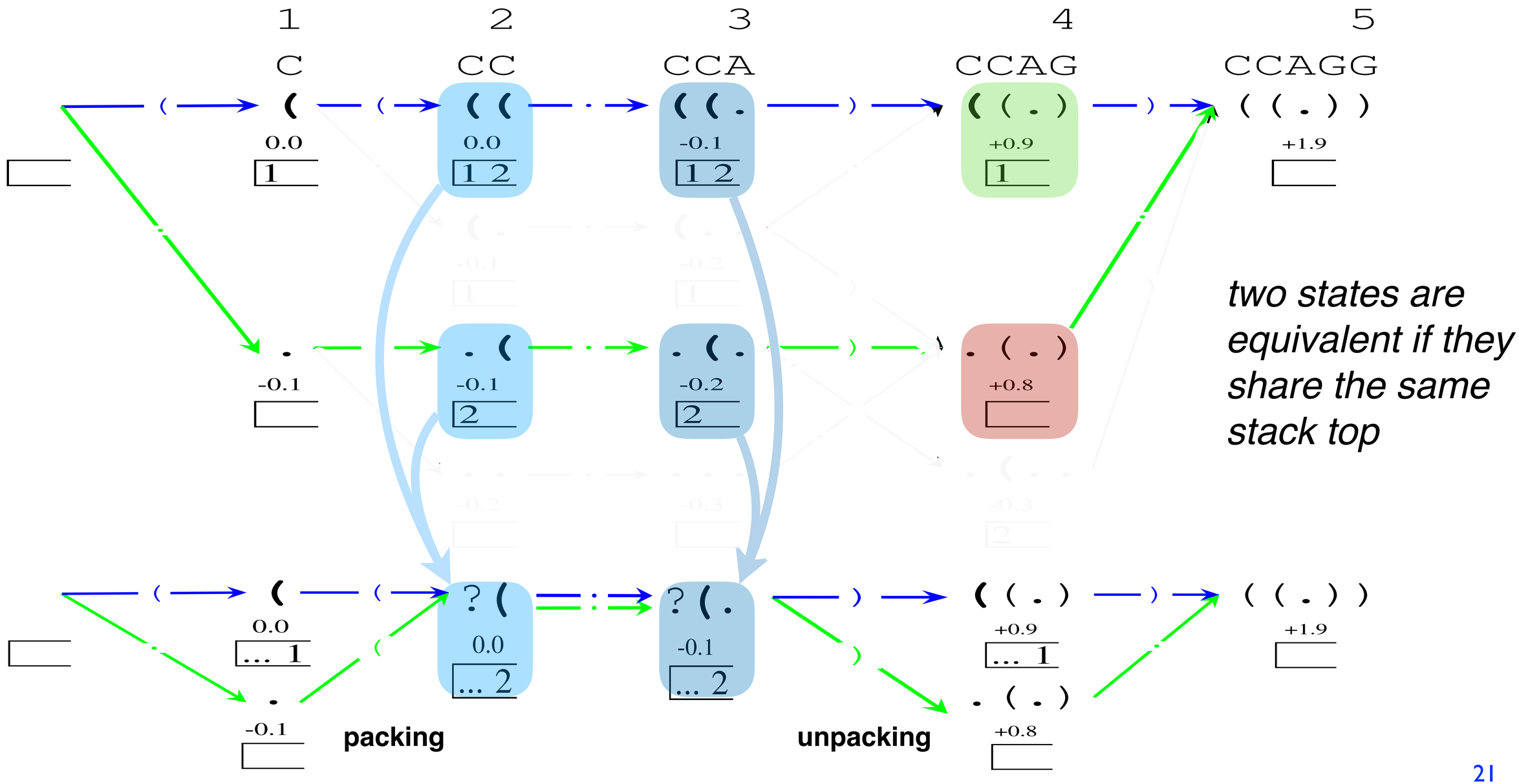
two states are equivalent if they share the same stack top

Details: Pack "Equivalent" States => $O(n^3)$



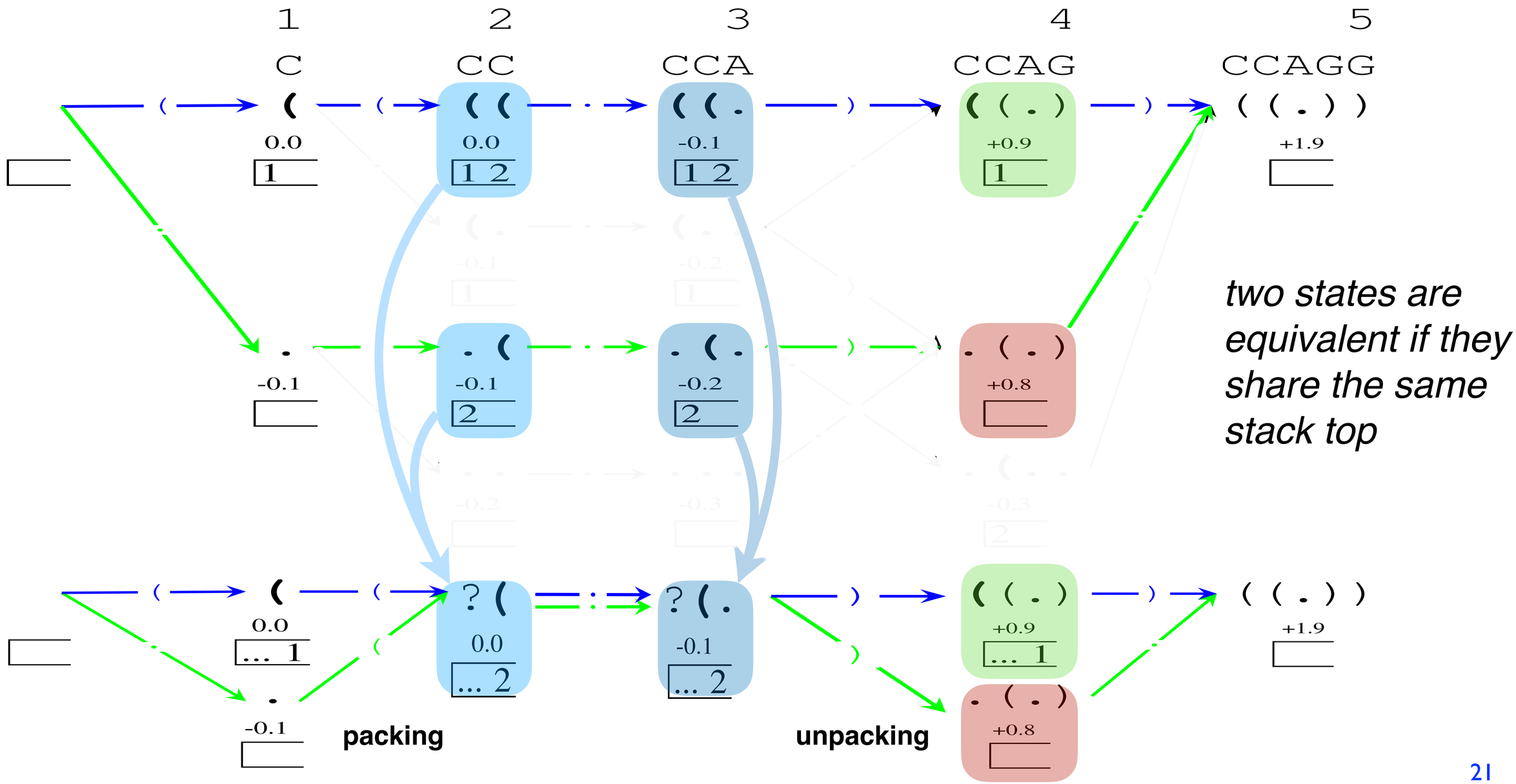
two states are equivalent if they share the same stack top

Details: Pack "Equivalent" States => $O(n^3)$



two states are equivalent if they share the same stack top

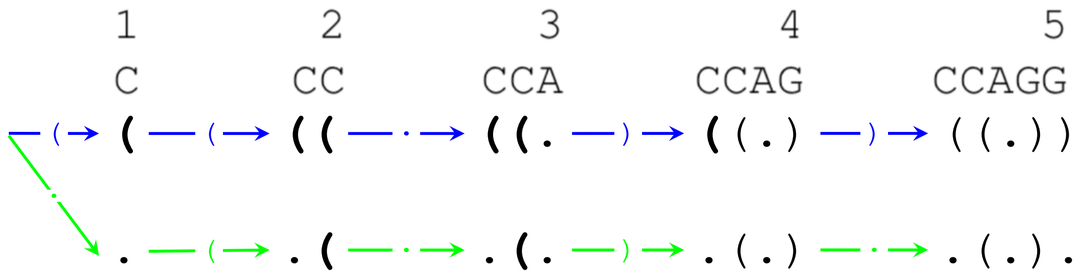
Details: Pack "Equivalent" States => $O(n^3)$

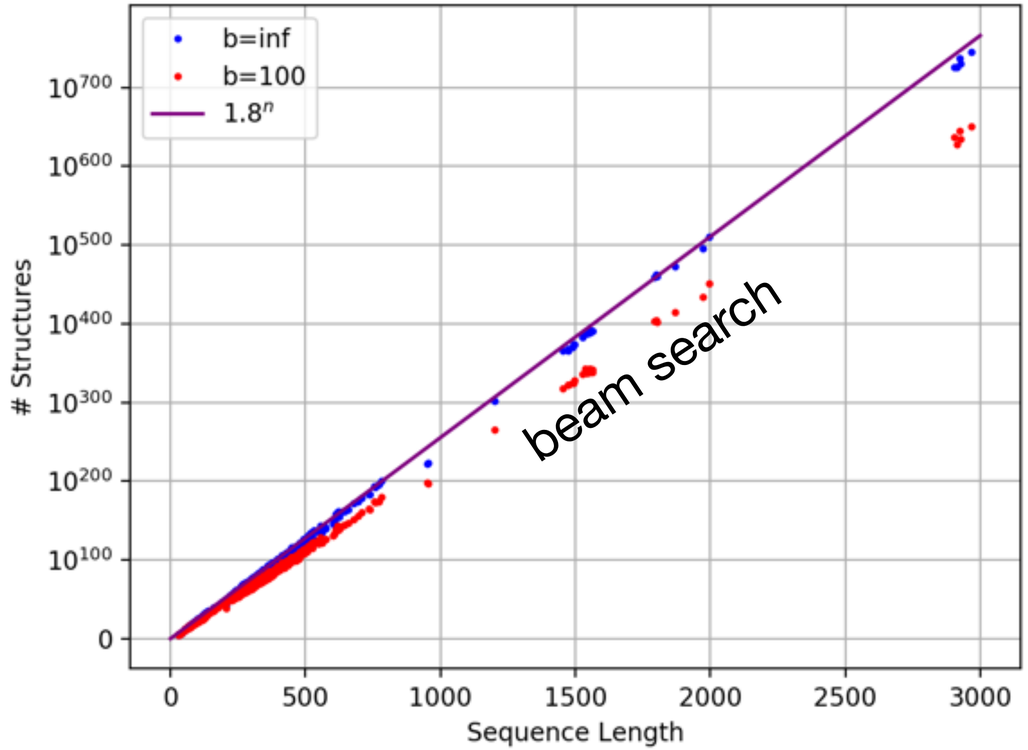


two states are equivalent if they share the same stack top

Details: Packing "Temporarily Equivalent" States

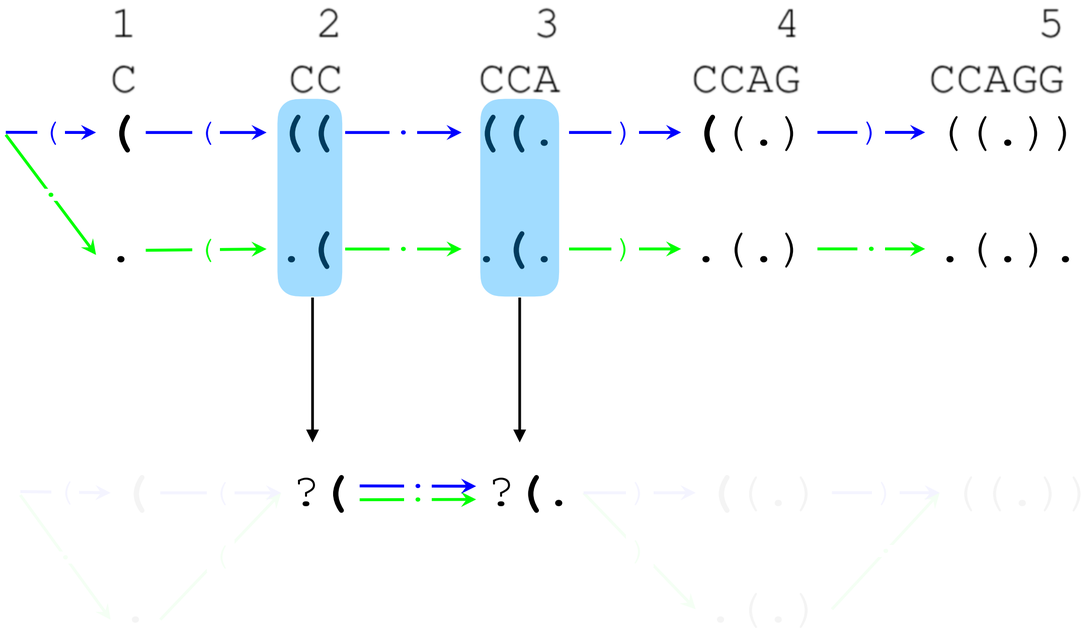
- two states are "temporarily equivalent" if they share the same stack top index
  - because they are looking for the same nucleotide(s) to match this stack-top nucleotide
  - we pack them as a single state until a matching is found, when they unpack
  - this is how we can explore exponentially many structures in linear time

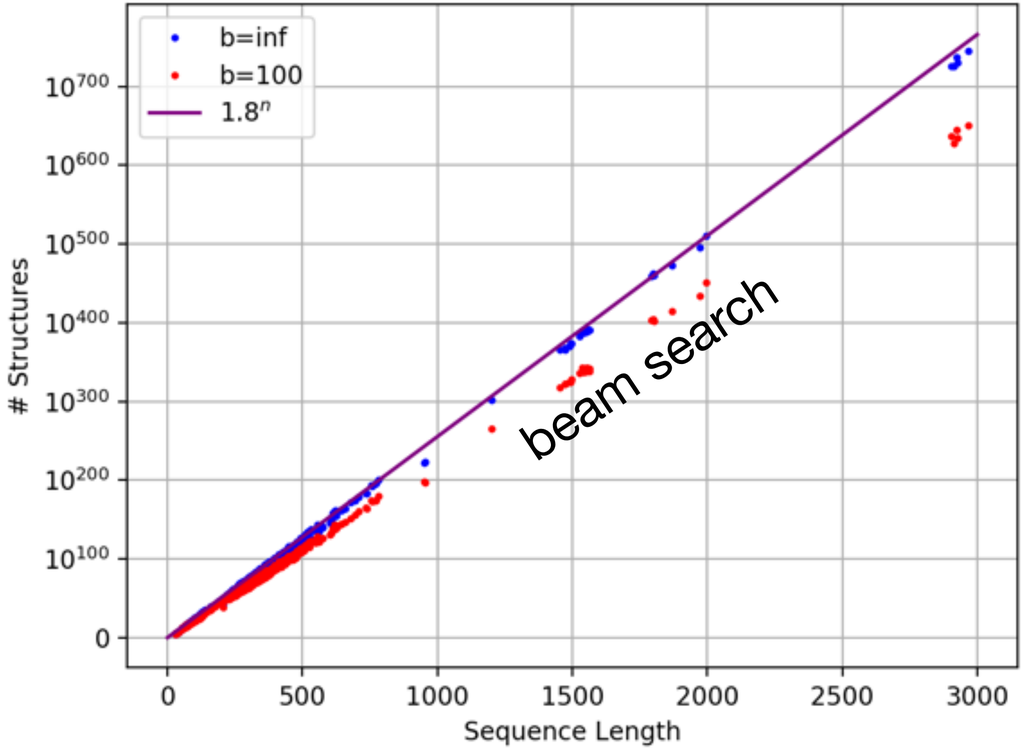




Details: Packing "Temporarily Equivalent" States

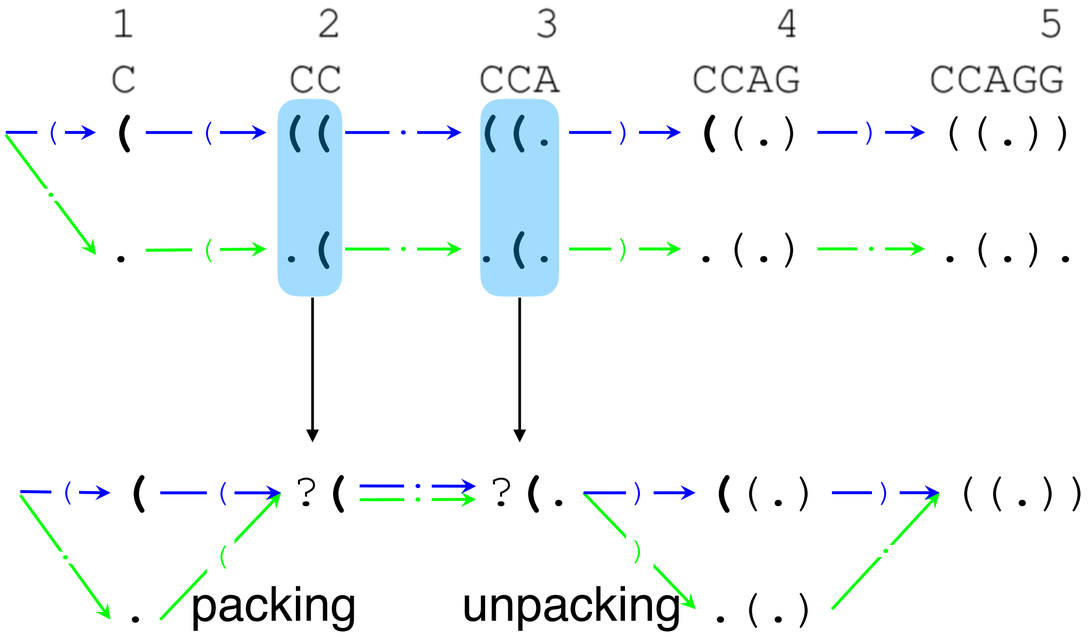
- two states are "temporarily equivalent" if they share the same stack top index
  - because they are looking for the same nucleotide(s) to match this stack-top nucleotide
  - we pack them as a single state until a matching is found, when they unpack
  - this is how we can explore exponentially many structures in linear time

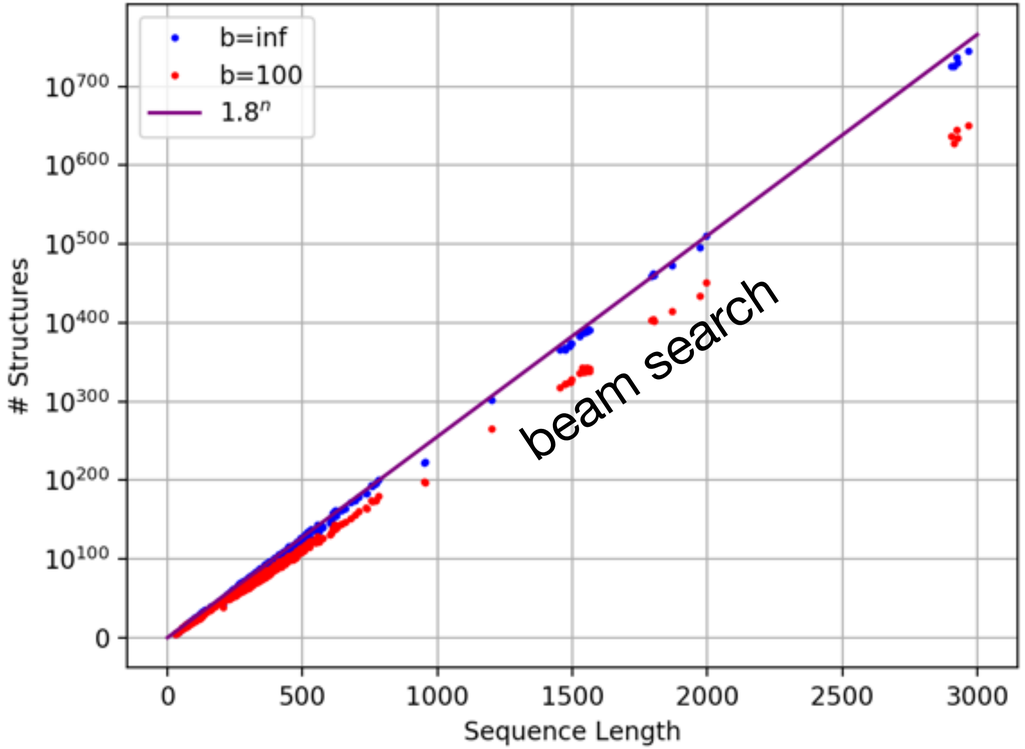




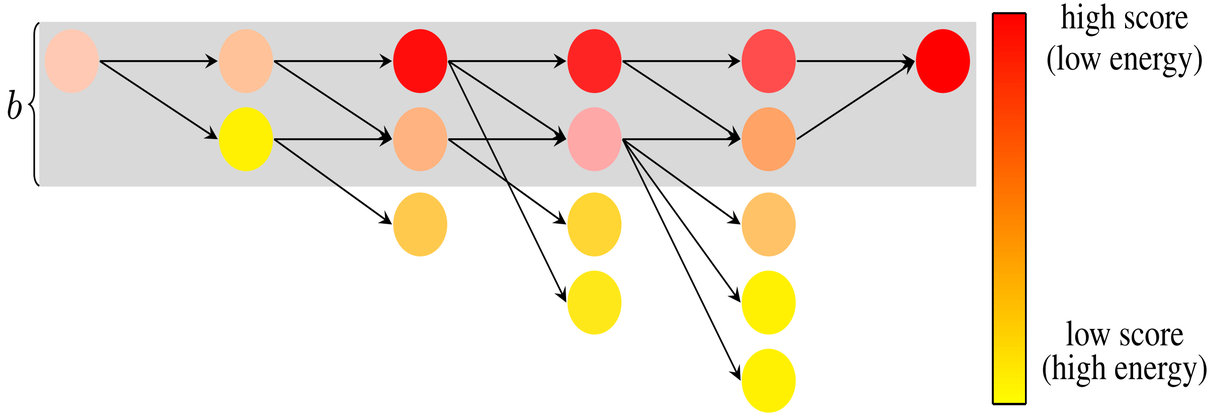
Details: Packing "Temporarily Equivalent" States

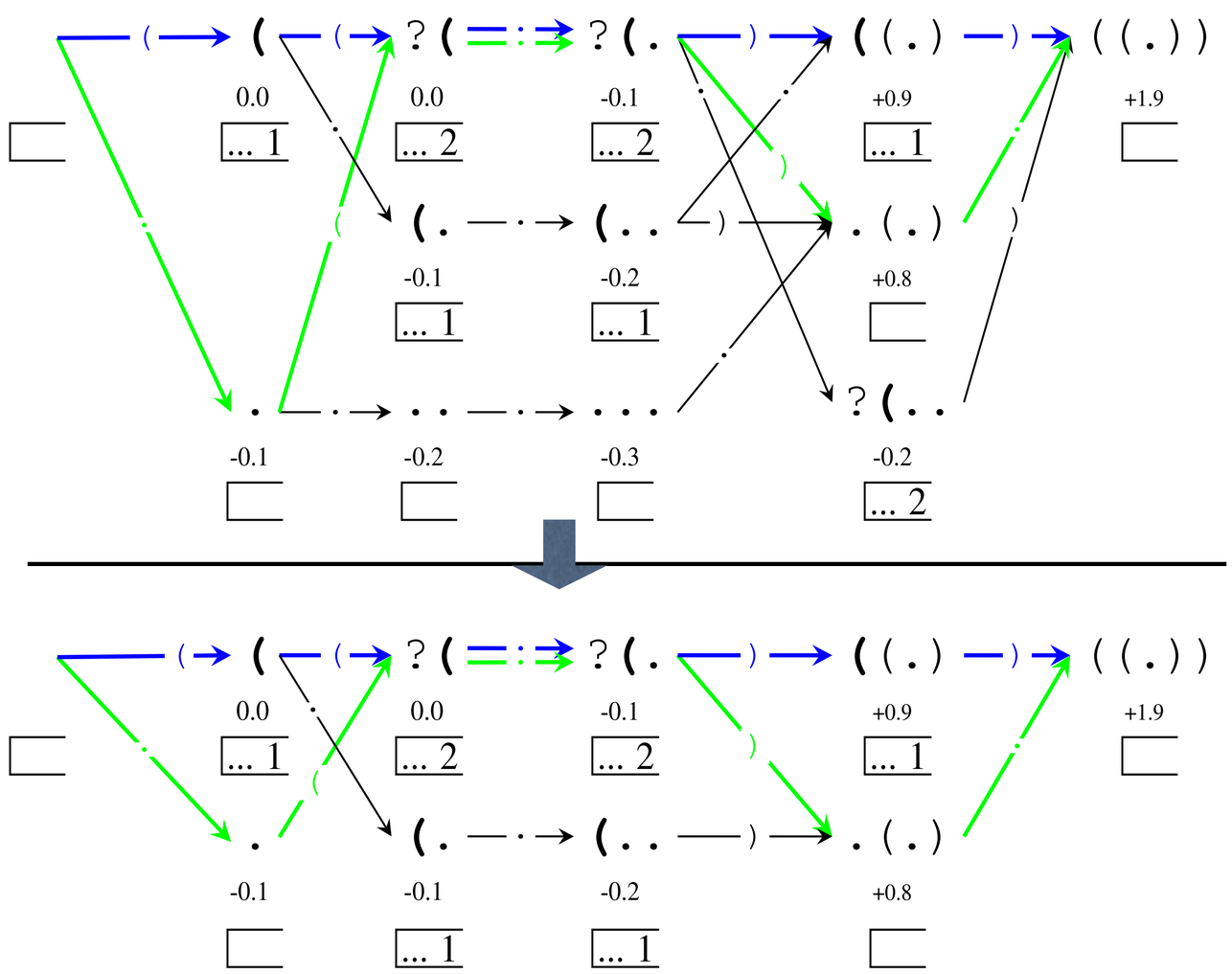
- two states are "temporarily equivalent" if they share the same stack top index
  - because they are looking for the same nucleotide(s) to match this stack-top nucleotide
  - we pack them as a single state until a matching is found, when they unpack
  - this is how we can explore exponentially many structures in linear time





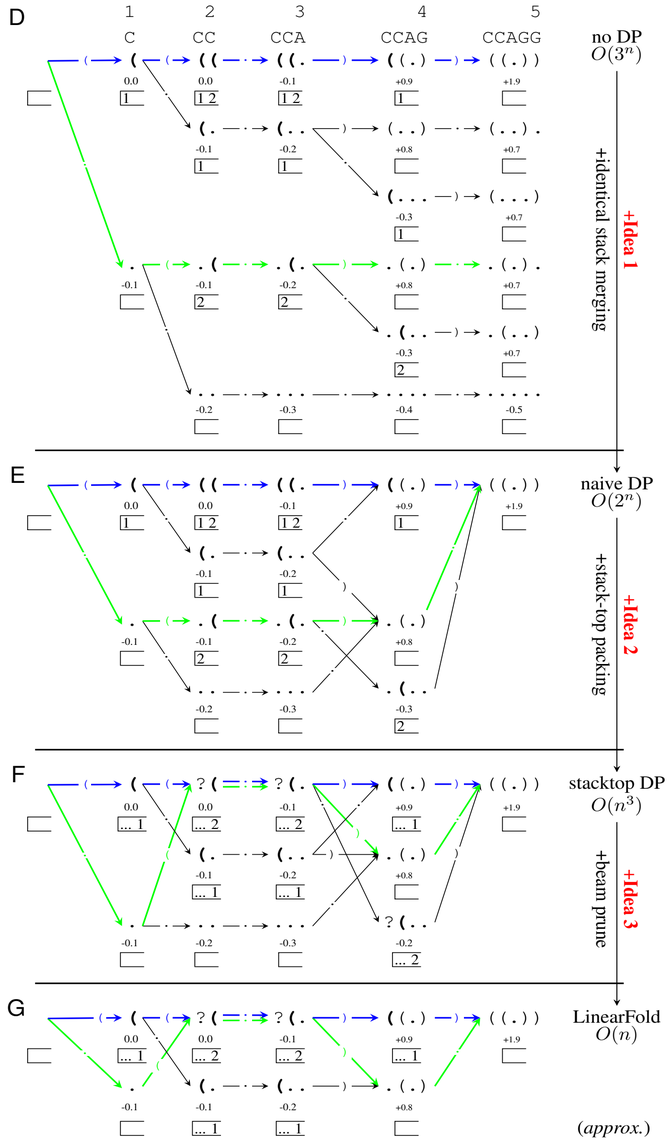
Idea 3: Beam Search => Version 4: $O(n)$, approx.





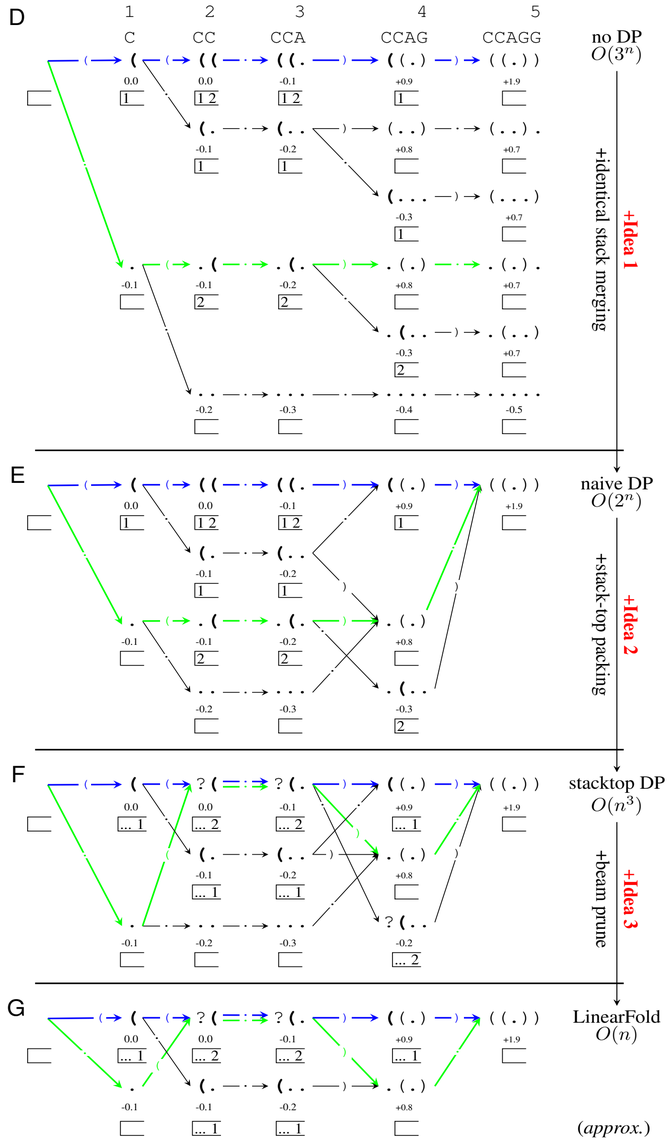
From $O(n^3)$ to $O(3^n)$ back to $O(n^3)$ finally to $O(n)$

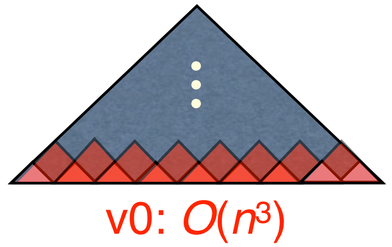
- v0: conventional bottom-up DP: $O(n^3)$
- 4 versions of incremental (5'-to-3') search
  - v1: exhaustive: $O(3^n)$
  - v2: DP, merge by full stack: $O(2^n)$
  - v3: DP, pack by stack top: $O(n^3)$
  - v4: approx. DP via beam search: $O(n)$
- this is a simple illustration on the toy "Nussinov-like" scoring model
  - our real systems have full scoring functions



From $O(n^3)$ to $O(3^n)$ back to $O(n^3)$ finally to $O(n)$

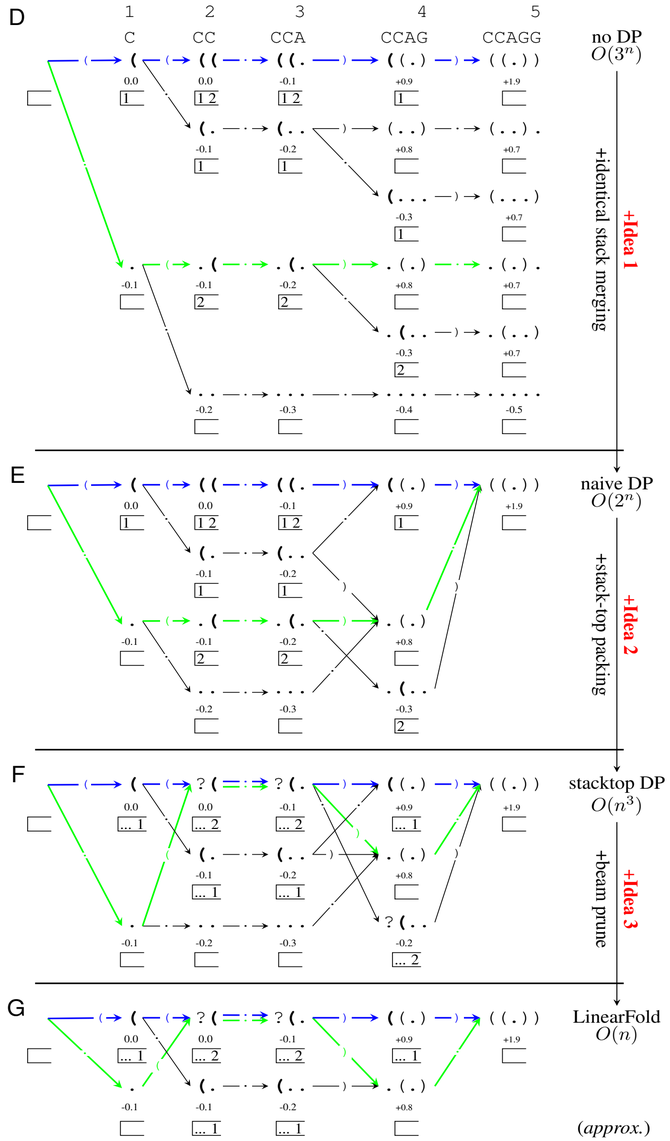
- v0: conventional bottom-up DP: $O(n^3)$
- 4 versions of incremental (5'-to-3') search
  - v1: exhaustive: $O(3^n)$
  - v2: DP, merge by full stack: $O(2^n)$
  - v3: DP, pack by stack top: $O(n^3)$
  - v4: approx. DP via beam search: $O(n)$
- this is a simple illustration on the toy "Nussinov-like" scoring model
  - our real systems have full scoring functions

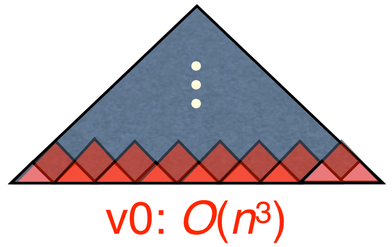


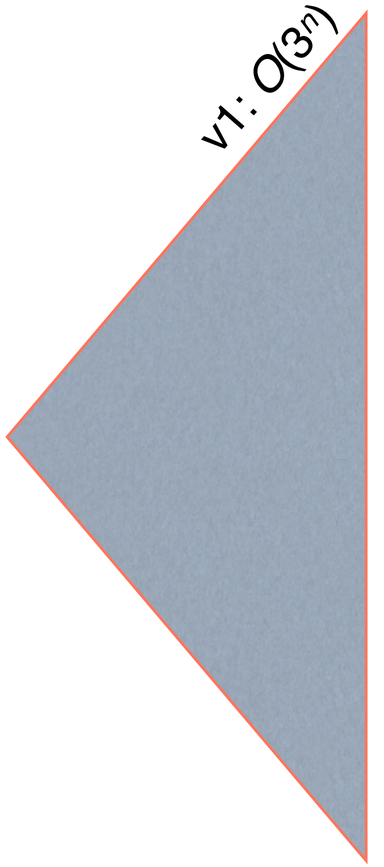


From $O(n^3)$ to $O(3^n)$ back to $O(n^3)$ finally to $O(n)$

- v0: conventional bottom-up DP: $O(n^3)$
- 4 versions of incremental (5'-to-3') search
  - v1: exhaustive: $O(3^n)$
  - v2: DP, merge by full stack: $O(2^n)$
  - v3: DP, pack by stack top: $O(n^3)$
  - v4: approx. DP via beam search: $O(n)$
- this is a simple illustration on the toy "Nussinov-like" scoring model
  - our real systems have full scoring functions

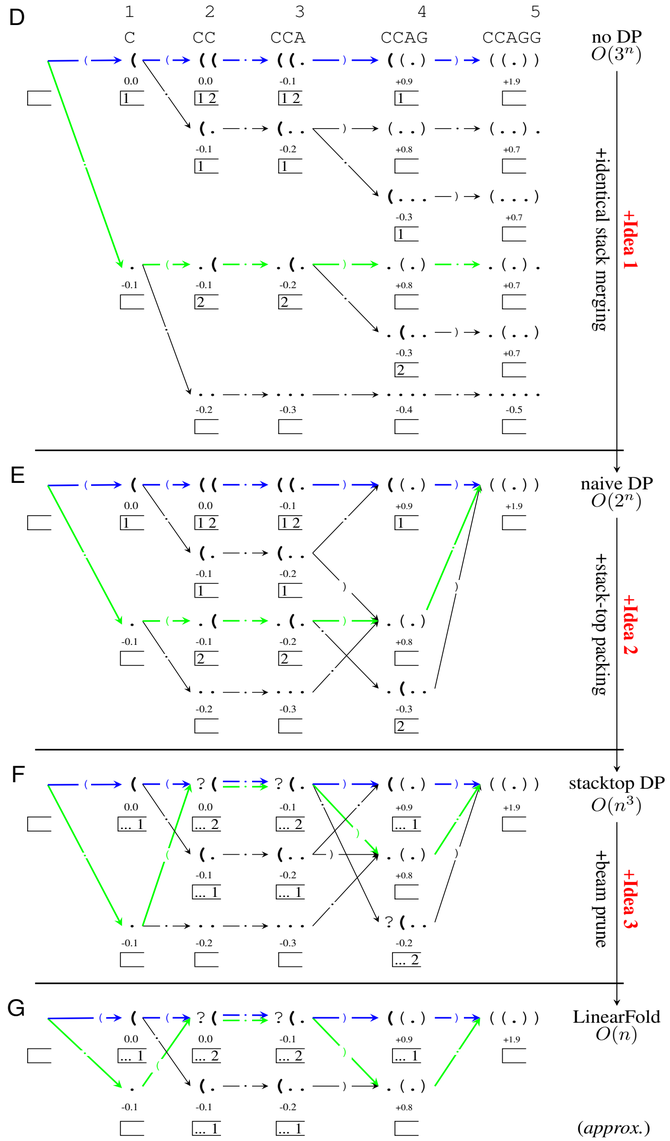


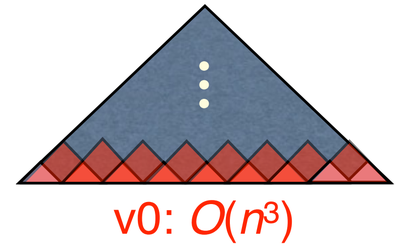


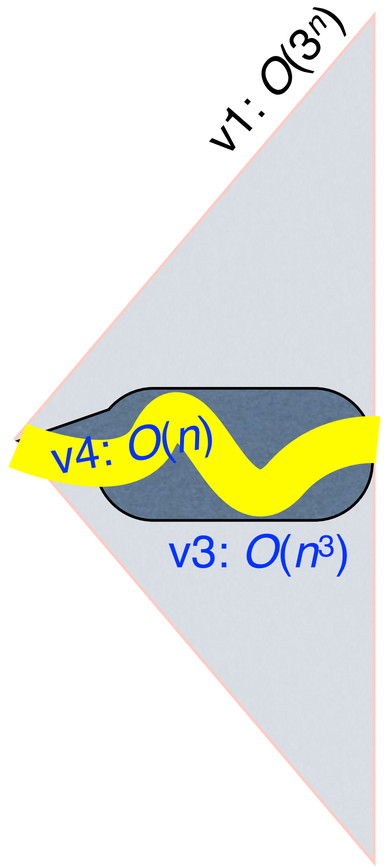


From $O(n^3)$ to $O(3^n)$ back to $O(n^3)$ finally to $O(n)$

- v0: conventional bottom-up DP: $O(n^3)$
- 4 versions of incremental (5'-to-3') search
  - v1: exhaustive: $O(3^n)$
  - v2: DP, merge by full stack: $O(2^n)$
  - v3: DP, pack by stack top: $O(n^3)$
  - v4: approx. DP via beam search: $O(n)$
- this is a simple illustration on the toy "Nussinov-like" scoring model
  - our real systems have full scoring functions

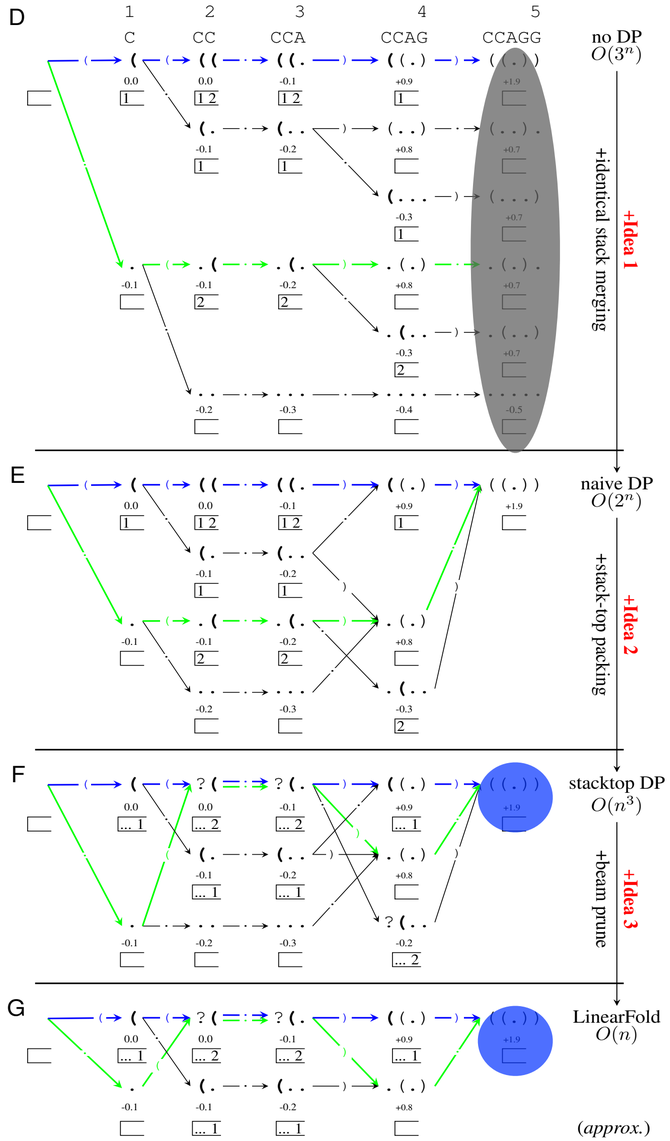


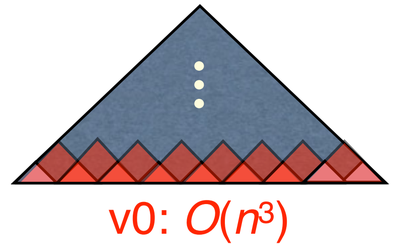


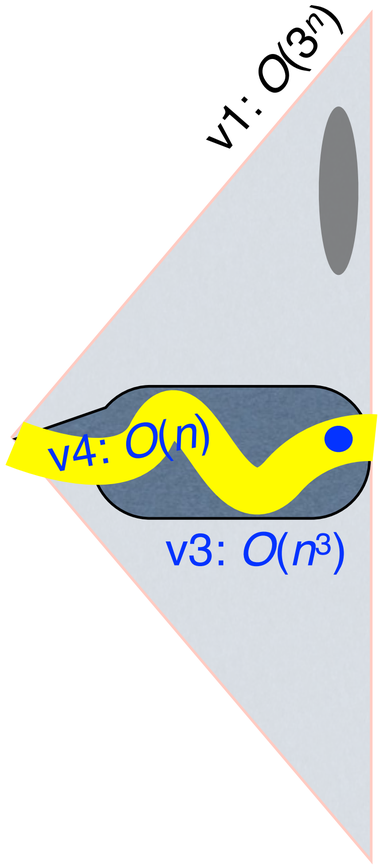


From $O(n^3)$ to $O(3^n)$ back to $O(n^3)$ finally to $O(n)$

- v0: conventional bottom-up DP: $O(n^3)$
- 4 versions of incremental (5'-to-3') search
  - v1: exhaustive: $O(3^n)$
  - v2: DP, merge by full stack: $O(2^n)$
  - v3: DP, pack by stack top: $O(n^3)$
  - v4: approx. DP via beam search: $O(n)$
- this is a simple illustration on the toy "Nussinov-like" scoring model
  - our real systems have full scoring functions



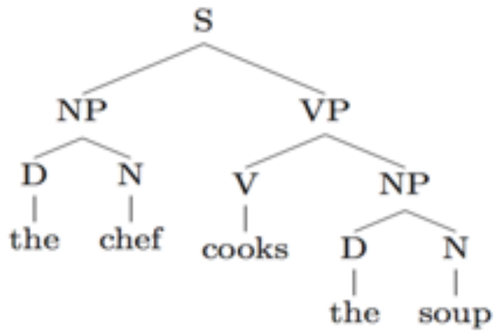




Computational Linguistics => Computational Biology

linguistics | **compiler theory** | **comp. linguistics** | **computational biology**

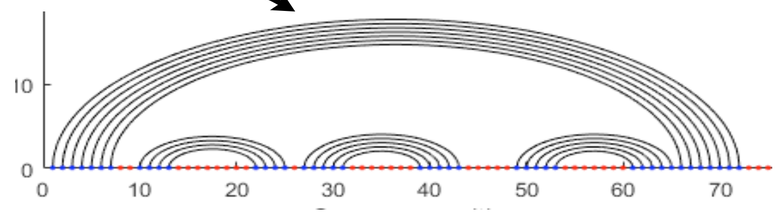
1955 Chomsky: context-free grammars (CFGs)



1958 Backus & Naur: CFGs for program, lang.

1964 **C**ocke \
1965 **K**asami - **CKY** $O(n^3)$
1967 **Y**ounger / for all CFGs

1965 Knuth: **LR** parsing for restricted CFGs: $O(n)$



1978: Nussinov $O(n^3)$ RNA folding
1981: Zuker & Siegler

1986 Tomita: **G**eneralized **LR** for all CFGs: $O(n^3)$

2019: 5'-to-3' $O(n^3)$ RNA folding (LinearFold with $+\infty$ beam)

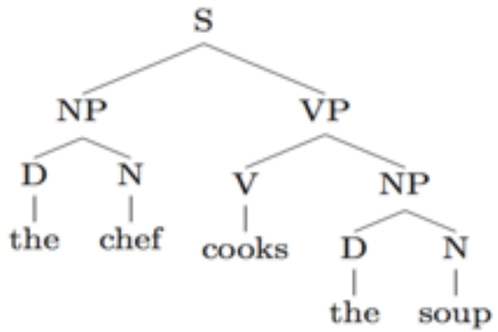
2010: Huang & Sagae: $O(n)$ (approx.) DP for all CFGs

2019: *LinearFold* $O(n)$ (approx.) RNA folding

Computational Linguistics => Computational Biology

linguistics

1955 Chomsky: context-free grammars (CFGs)



compiler theory

1958 Backus & Naur: CFGs for program, lang.

1965 Knuth: **LR** parsing for restricted CFGs: ***O(n)***

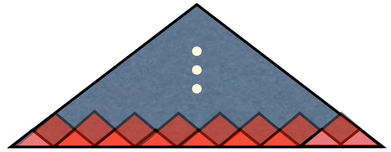
comp. linguistics

1964 **C**ocke \
1965 **K**asami - **CKY** ***O(n³)*** **v0**
1967 **Y**ounger / for all CFGs

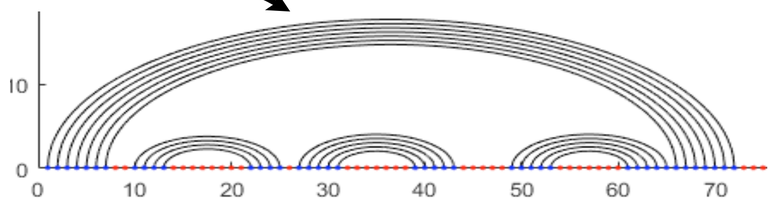
1986 Tomita: **G**eneralized **LR** for all CFGs: ***O(n³)***

2010: Huang & Sagae: ***O(n)*** (approx.) DP for all CFGs

computational biology



v0: $O(n^3)$



1978: Nussinov ***O(n³)*** RNA folding
1981: Zuker & Siegler **v0**

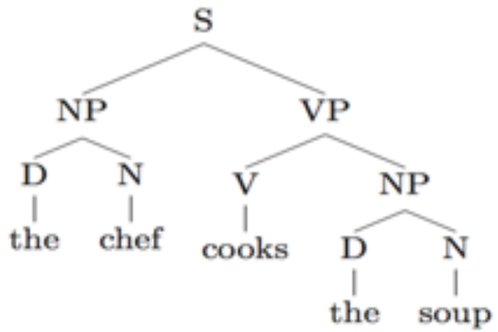
2019: 5'-to-3' ***O(n³)*** RNA folding (LinearFold with +∞ beam)

2019: *LinearFold O(n)* (approx.) RNA folding

Computational Linguistics => Computational Biology

linguistics

1955 Chomsky: context-free grammars (CFGs)



compiler theory

1958 Backus & Naur: CFGs for program, lang.

1965 Knuth: **LR** parsing for restricted CFGs: ***O(n)***

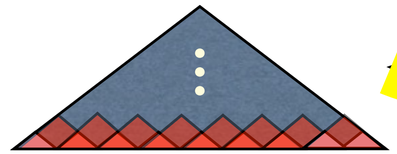
comp. linguistics

1964 **C**ocke \
1965 **K**asami - **CKY** ***O(n³)*** **v0**
1967 **Y**ounger / for all CFGs

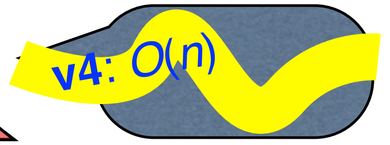
1986 Tomita: **G**eneralized **LR** for all CFGs: ***O(n³)*** **v3**

2010: Huang & Sagae: ***O(n)*** **v4** (approx.) DP for all CFGs

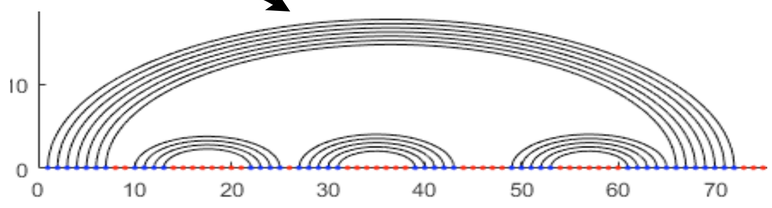
computational biology



v0: $O(n^3)$



v3: $O(n^3)$



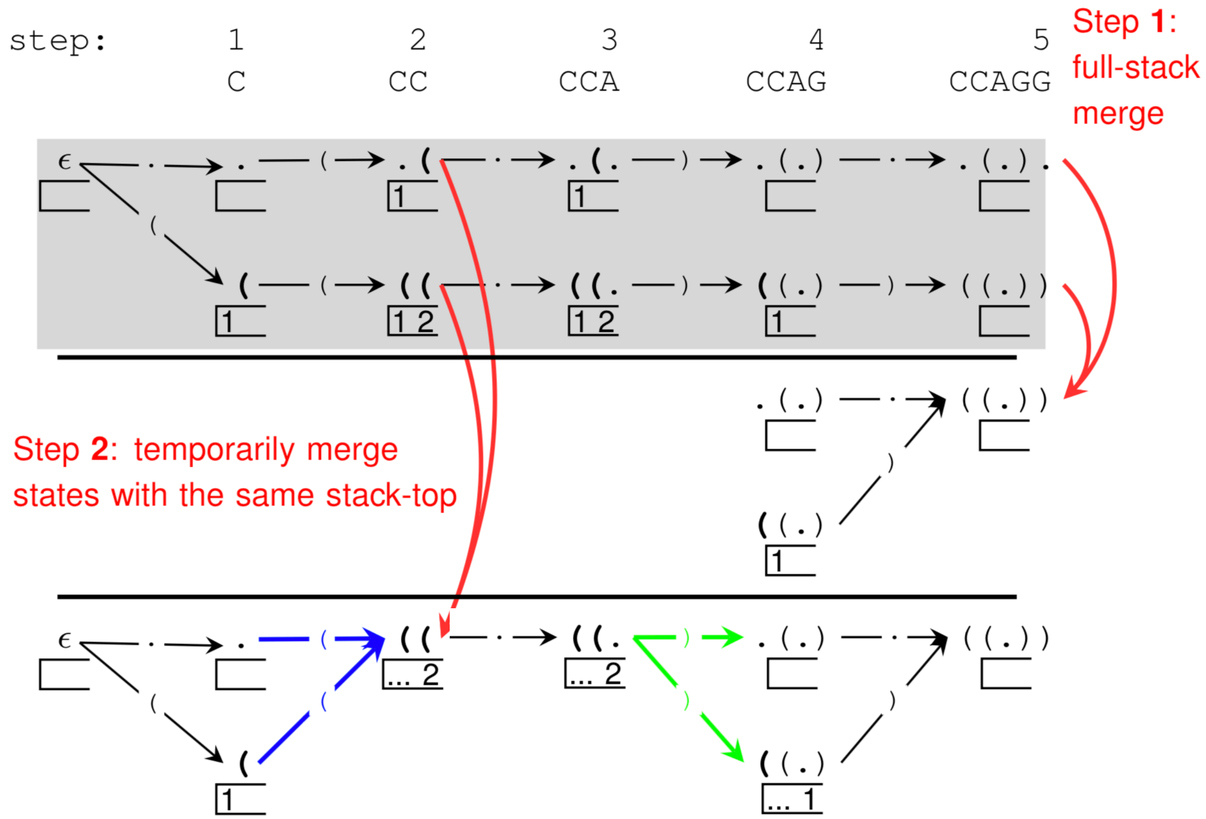
1978: Nussinov ***O(n³)*** RNA folding
1981: Zuker & Siegler **v0**

2019: 5'-to-3' ***O(n³)*** RNA folding **v3** (LinearFold with +∞ beam)

2019: *LinearFold O(n)* **v4** (approx.) RNA folding

Connections to Incremental Parsing

- shared key observation: local ambiguity packing
  - pack non-crucial local ambiguities along the way
  - unpack (in a reduce action) only when needed



psycholinguistic evidence
(eye-tracking experiments):

delayed disambiguation

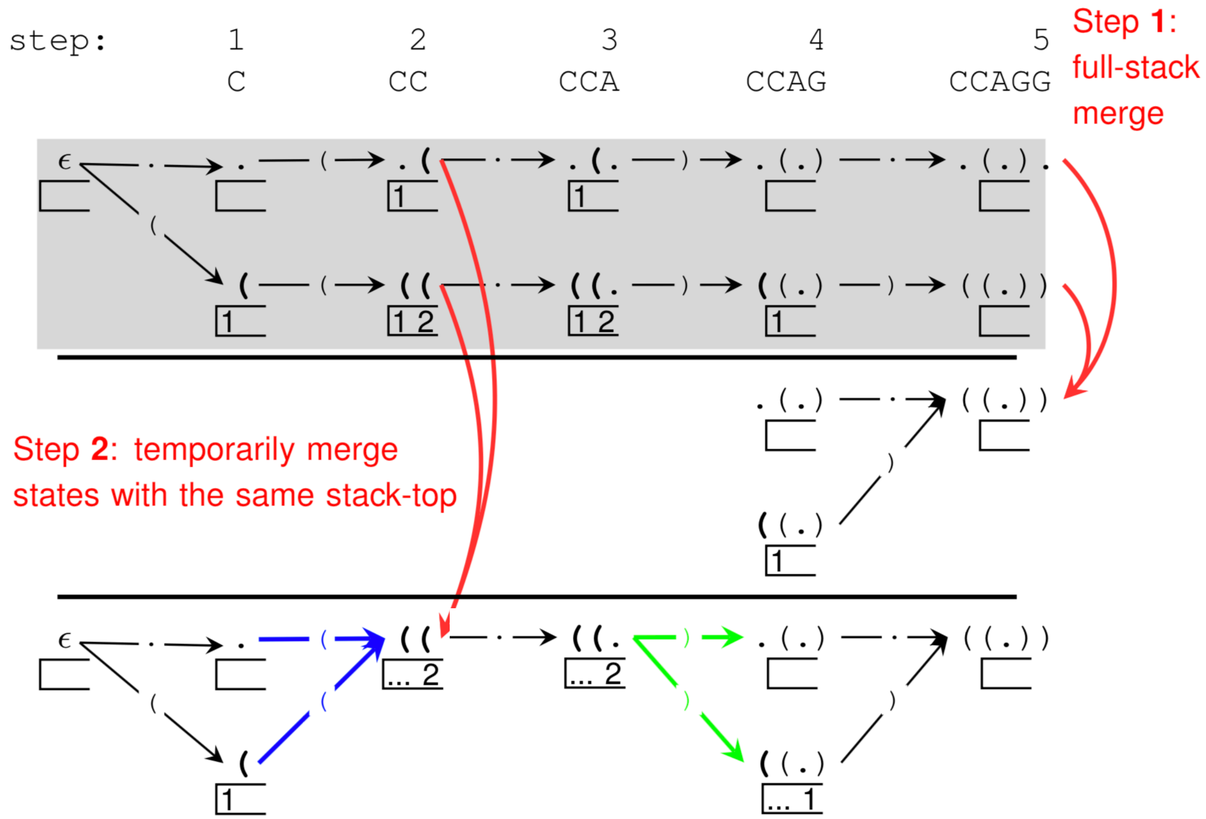
John and Mary had 2 papers
John and Mary had 2 papers

Frazier and Rayner (1990), Frazier (1999)

(Huang and Sagae, 2010)

Connections to Incremental Parsing

- shared key observation: local ambiguity packing
  - pack non-crucial local ambiguities along the way
  - unpack (in a reduce action) only when needed



psycholinguistic evidence
(eye-tracking experiments):

delayed disambiguation

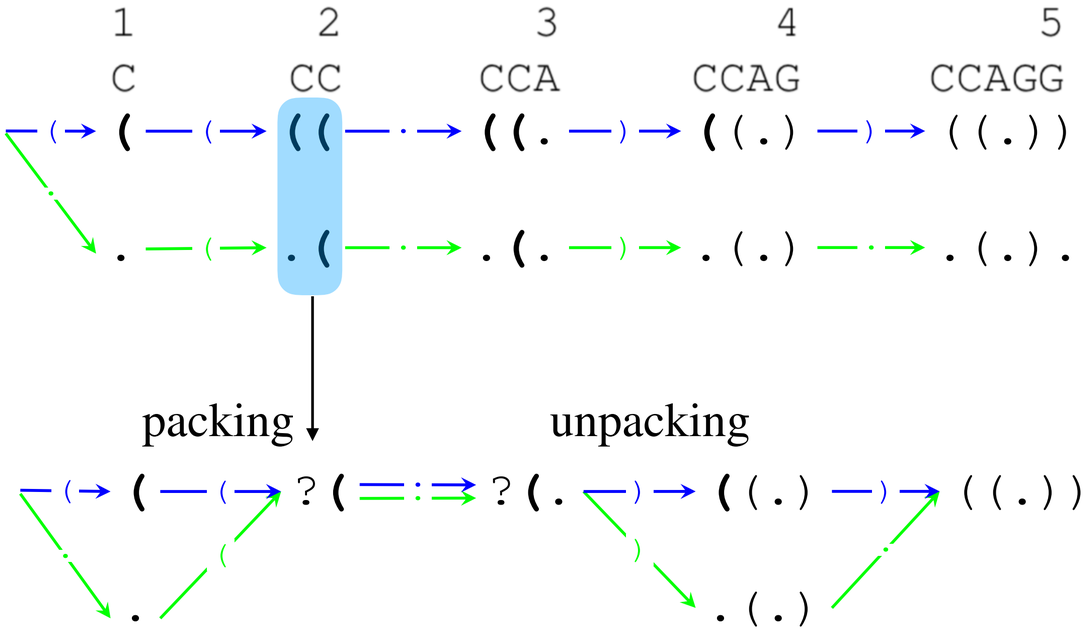
John and Mary had 2 papers each
John and Mary had 2 papers together

Frazier and Rayner (1990), Frazier (1999)

(Huang and Sagae, 2010)

Connections to Incremental Parsing

- shared key observation: local ambiguity packing
  - pack non-crucial local ambiguities along the way
  - unpack (in a reduce action) only when needed



psycholinguistic evidence
(eye-tracking experiments):

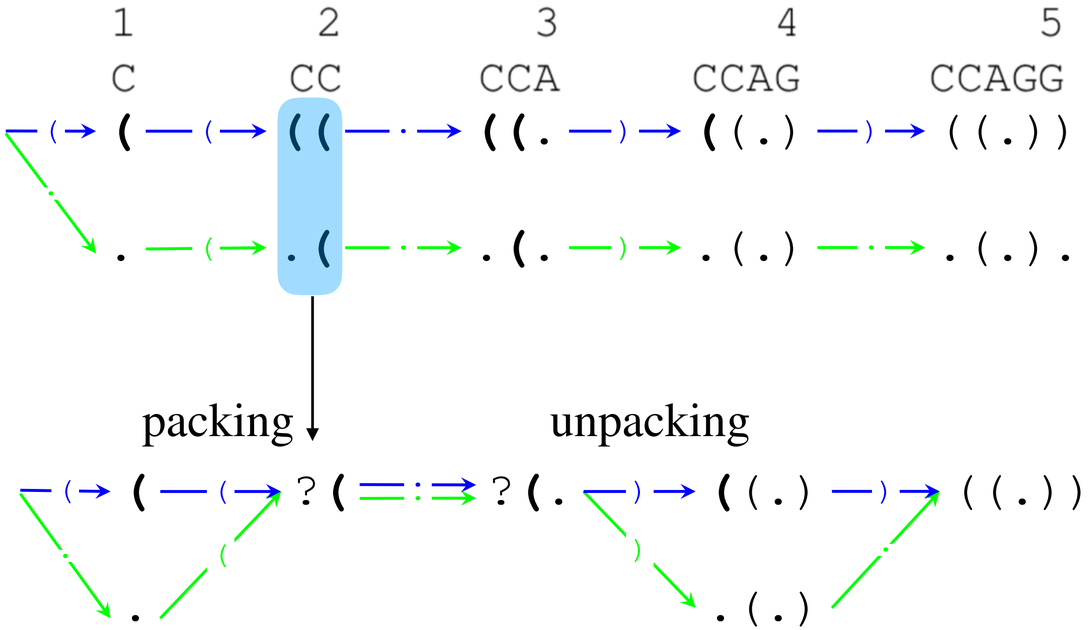
delayed disambiguation

John and Mary had 2 papers
John and Mary had 2 papers

Frazier and Rayner (1990), Frazier (1999)

Connections to Incremental Parsing

- shared key observation: local ambiguity packing
  - pack non-crucial local ambiguities along the way
  - unpack (in a reduce action) only when needed



psycholinguistic evidence
(eye-tracking experiments):

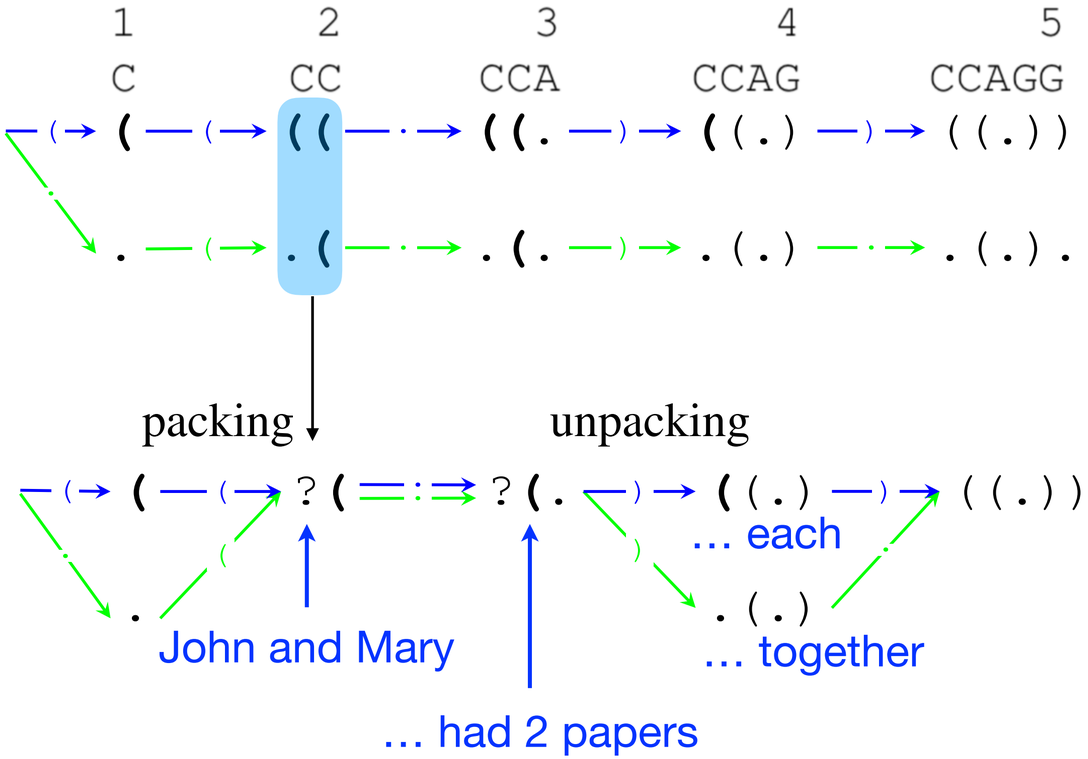
delayed disambiguation

John and Mary had 2 papers each
John and Mary had 2 papers together

Frazier and Rayner (1990), Frazier (1999)

Connections to Incremental Parsing

- shared key observation: local ambiguity packing
  - pack non-crucial local ambiguities along the way
  - unpack (in a reduce action) only when needed



psycholinguistic evidence
(eye-tracking experiments):

delayed disambiguation

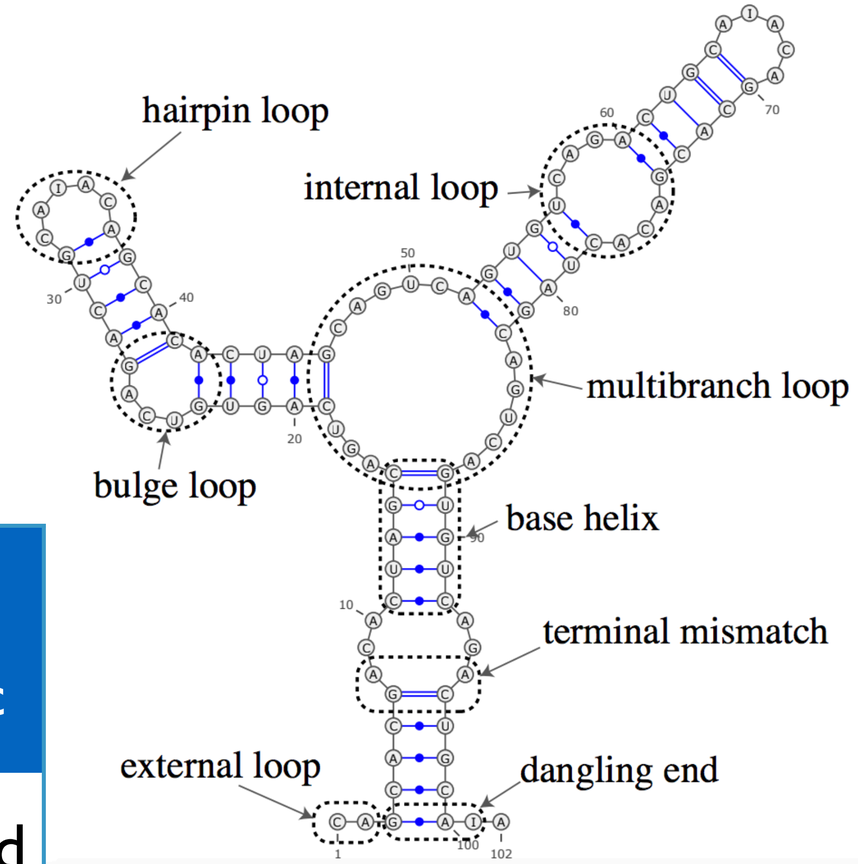
John and Mary had 2 papers each
John and Mary had 2 papers together

Frazier and Rayner (1990), Frazier (1999)

Results

LinearFold with SOTA Prediction Models

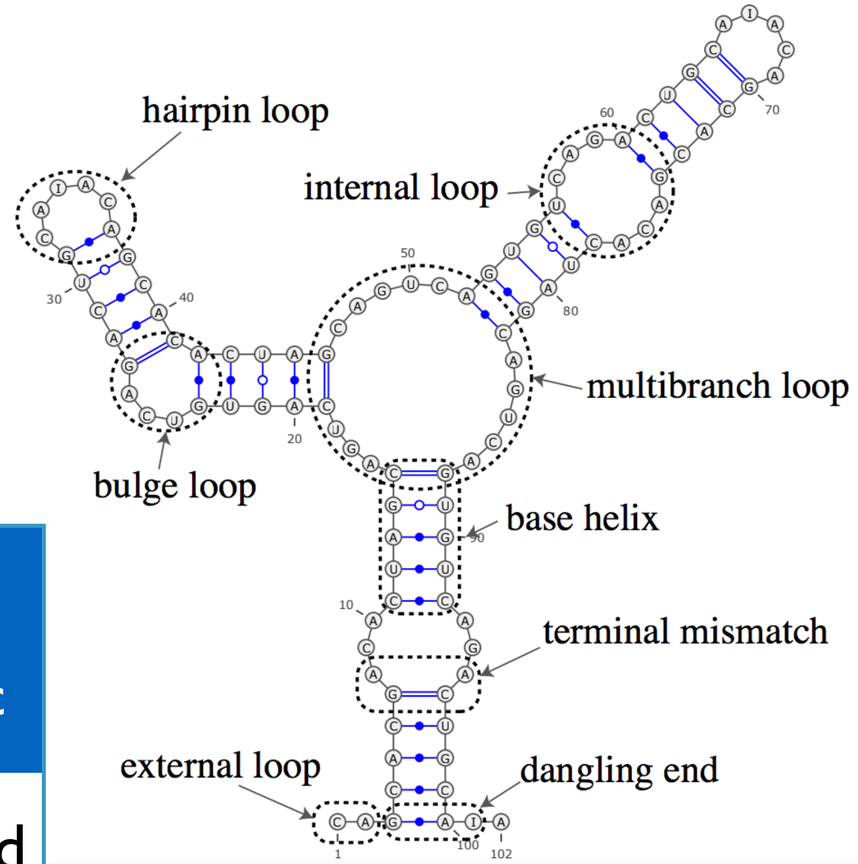
- models from two widely-used folding engines
  - CONTRAfold MFE (machine-learned)
  - Vienna RNAfold (thermodynamic)
- we linearized both systems from $O(n^3)$ to $O(n)$



	efficiency		*systems*	
	time	space	machine-learned	thermo-dynamic
baselines	$O(n^3)$	$O(n^2)$	**C**ONTRAfold	**V**ienna RNAfold
our work	$O(n)$	$O(n)$	LinearFold-**C**	LinearFold-**V**

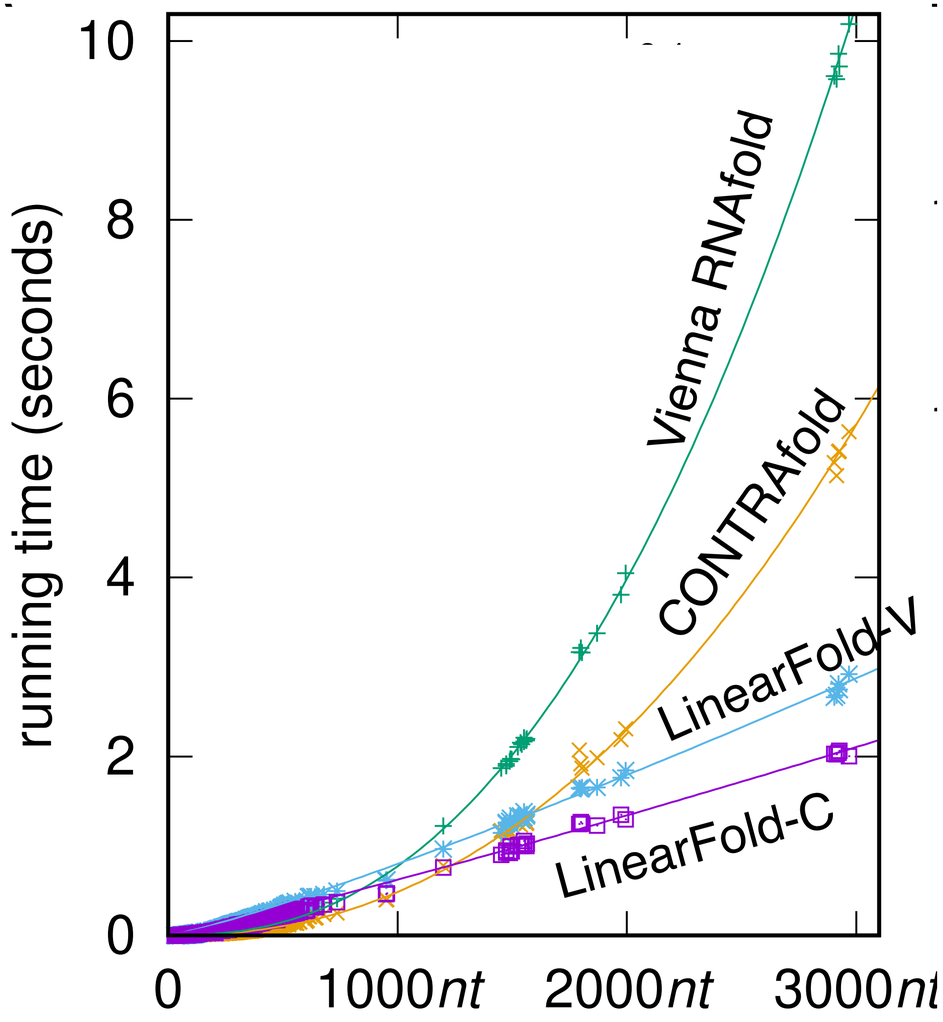
LinearFold with SOTA Prediction Models

- models from two widely-used folding engines
  - CONTRAfold MFE (machine-learned)
  - Vienna RNAfold (thermodynamic)
- we linearized both systems from $O(n^3)$ to $O(n)$

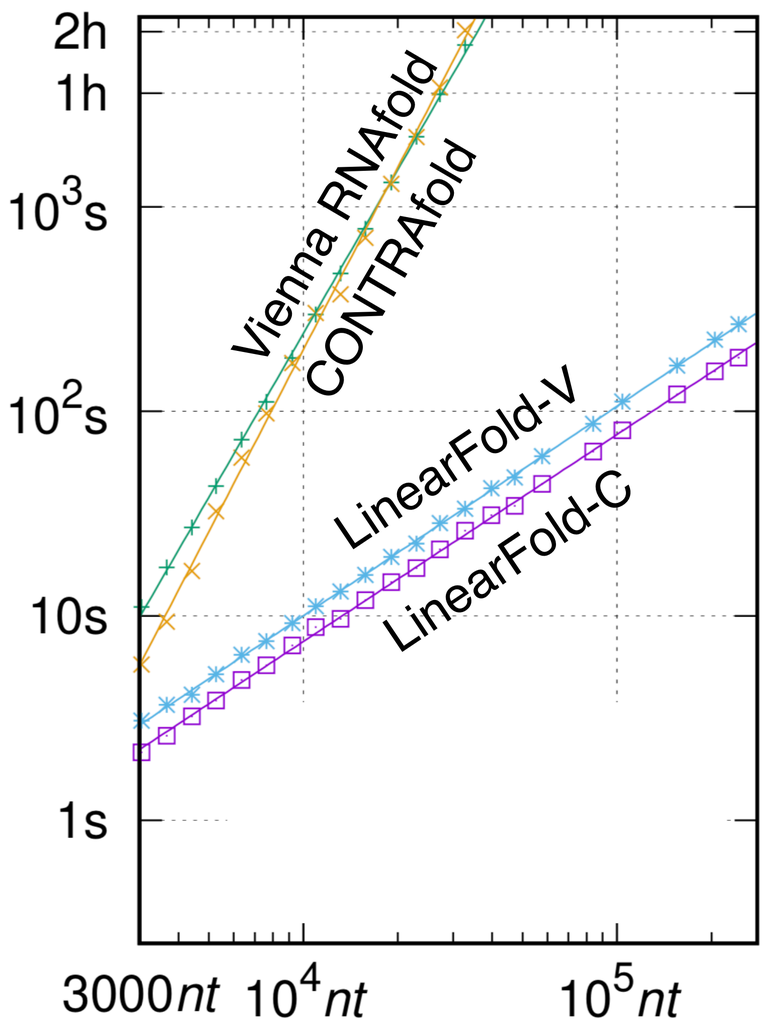


	efficiency		*systems*	
	time	space	machine-learned	thermo-dynamic
baselines	$O(n^3)$	$O(n^2)$	**C**ONTRAfold	**V**ienna RNAfold
our work	$O(n)$	$O(n)$	LinearFold-**C**	LinearFold-**V**

Efficiency & Scalability: *O*(*n*) time, *O*(*n*) memory



Archive II data set
(~3,000 seqs, max len: ~3,000 *nt*)



RNAcentral data set
(sampled, max len: ~250,000 *nt*)

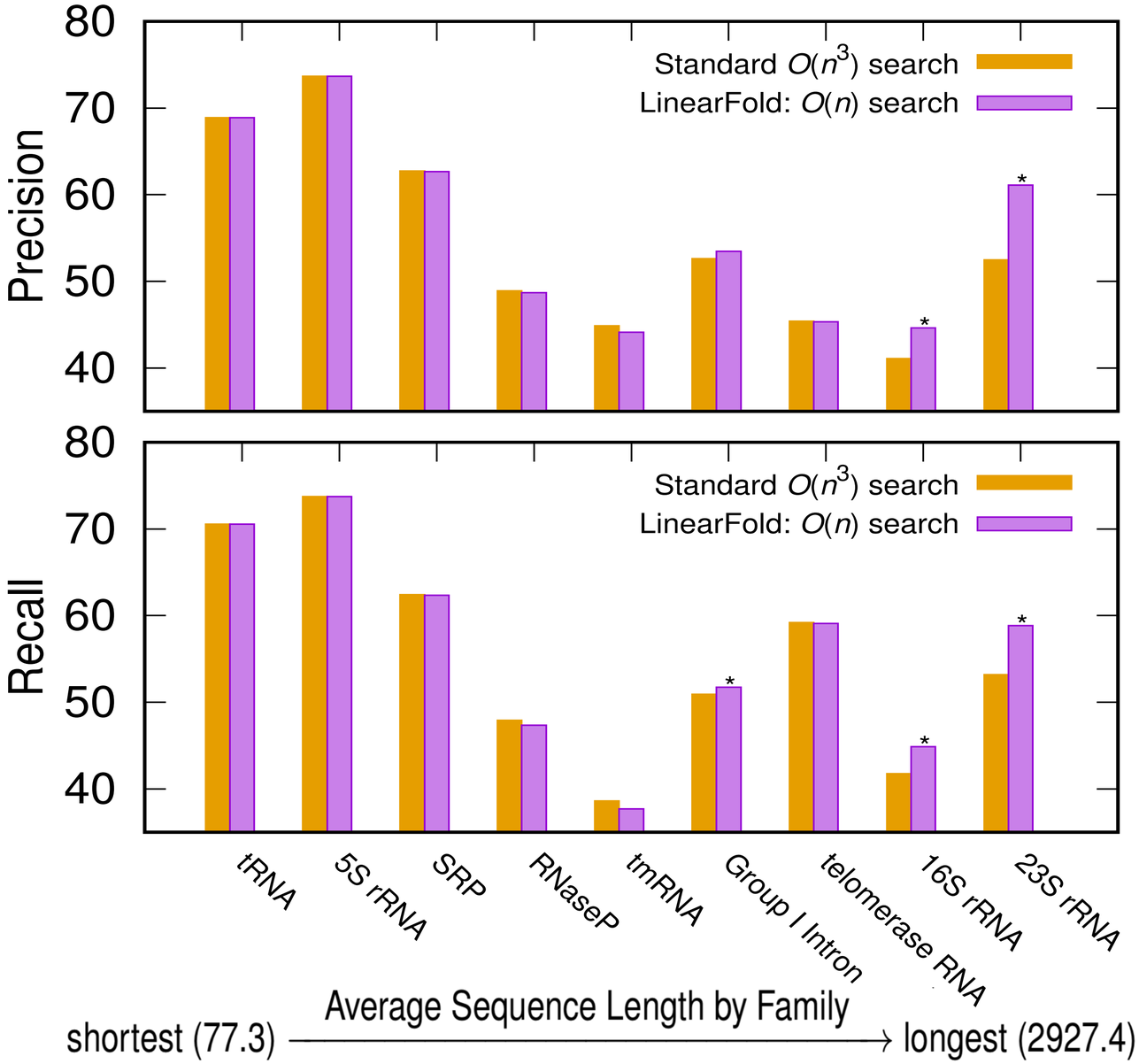
10,000*nt* (~HIV)
4min → 7s

244,296*nt*
(longest in RNAcentral)
~200hrs → 120s

Accuracy

- Tested on Archive II dataset (on a family-by-family basis)
  - significantly better on 3 long families
  - biggest boost on the longest families: 16S/23S rRNAs
- LinearFold-V vs. Vienna is similar

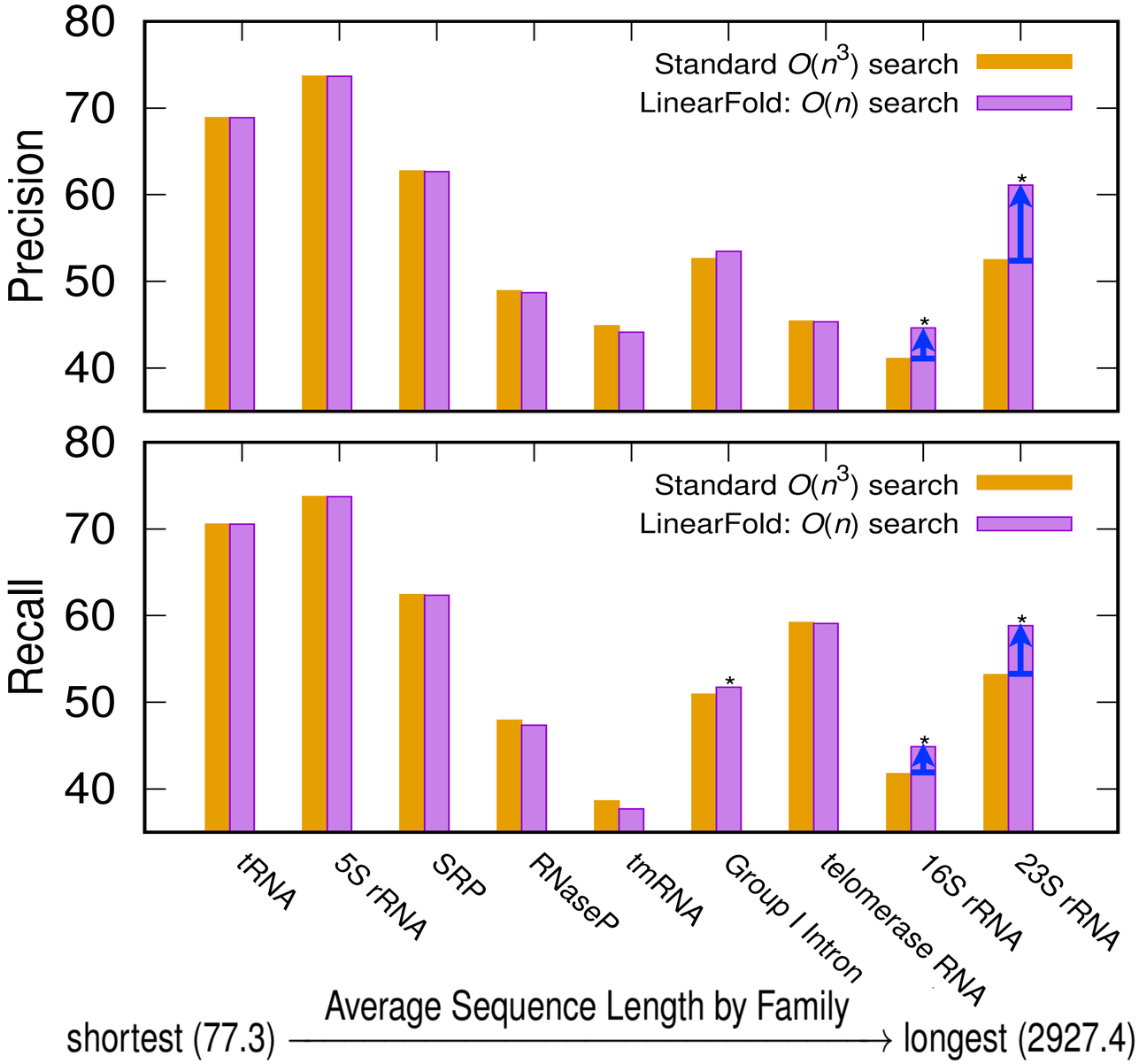
Overall	(precision) PPV		(recall) Sensitivity	
CONTRAfold MFE	54.51		55.36	
LinearFold-C	55.84	(+1.3)	56.24	(+0.9)
Vienna RNAfold	50.22		58.74	
LinearFold-V	50.51	(+0.3)	58.97	(+0.2)



Accuracy

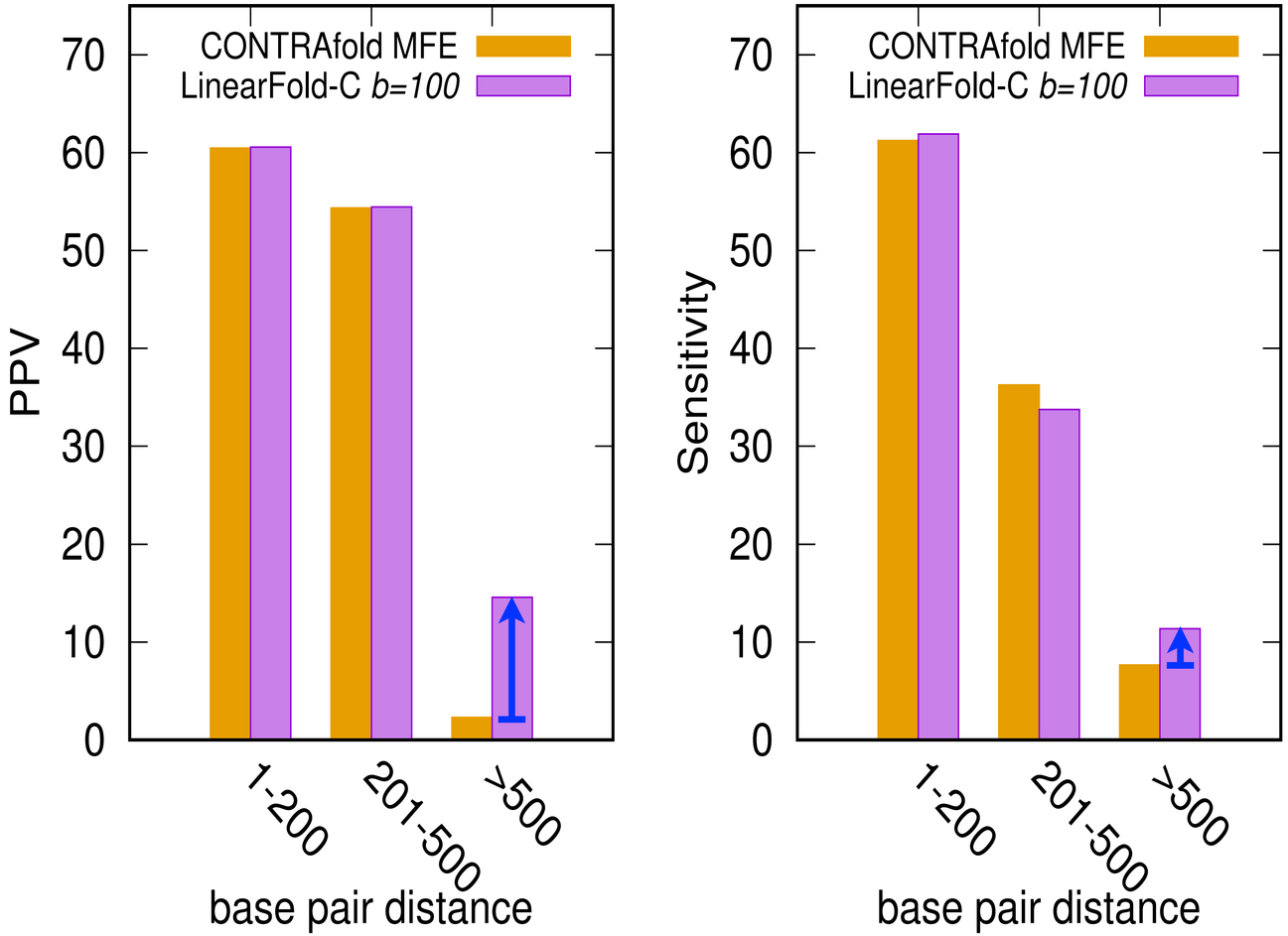
- Tested on Archive II dataset (on a family-by-family basis)
  - significantly better on 3 long families
  - biggest boost on the longest families: 16S/23S rRNAs
- LinearFold-V vs. Vienna is similar

	(precision)		(recall)	
Overall	PPV		Sensitivity	
CONTRAfold MFE	54.51		55.36	
LinearFold-C	55.84	(+1.3)	56.24	(+0.9)
Vienna RNAfold	50.22		58.74	
LinearFold-V	50.51	(+0.3)	58.97	(+0.2)



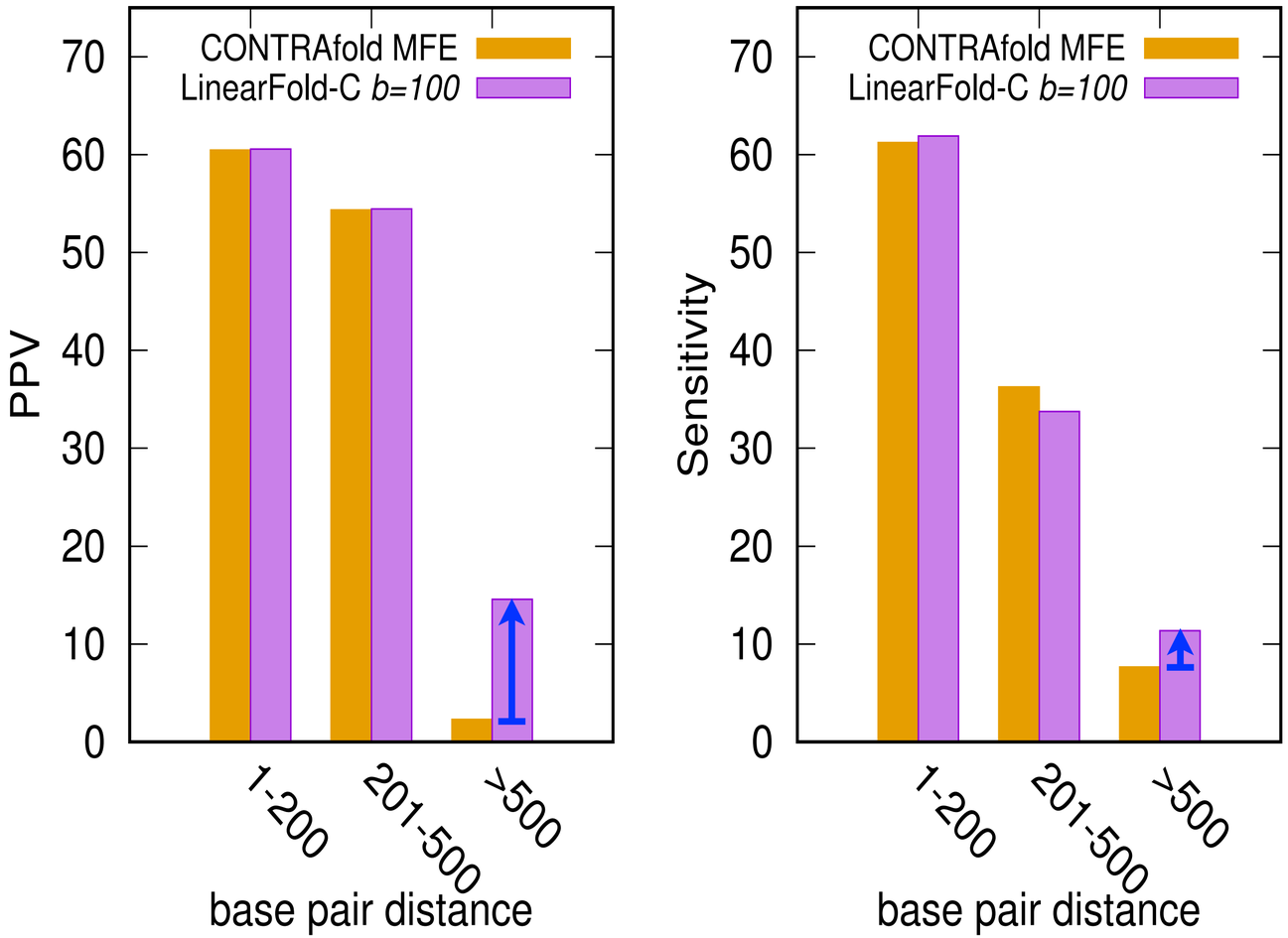
Improvements on Long-Range Base Pairs

- long-distance pairs are well-known to be hard to predict
  - LinearFold outputs less long-range base pairs, but more correct ones

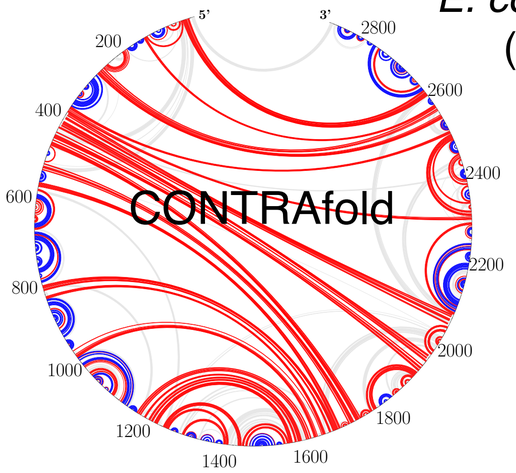


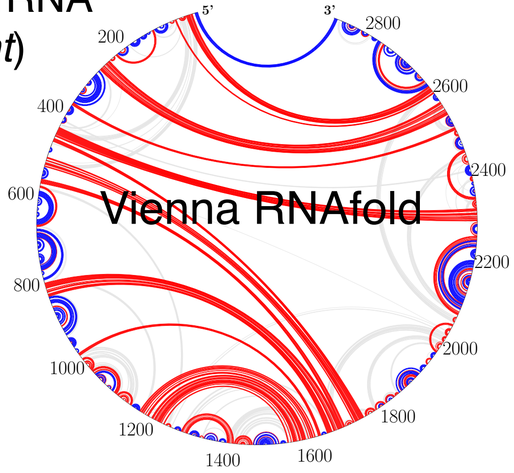
Improvements on Long-Range Base Pairs

- long-distance pairs are well-known to be hard to predict
  - LinearFold outputs less long-range base pairs, but more correct ones



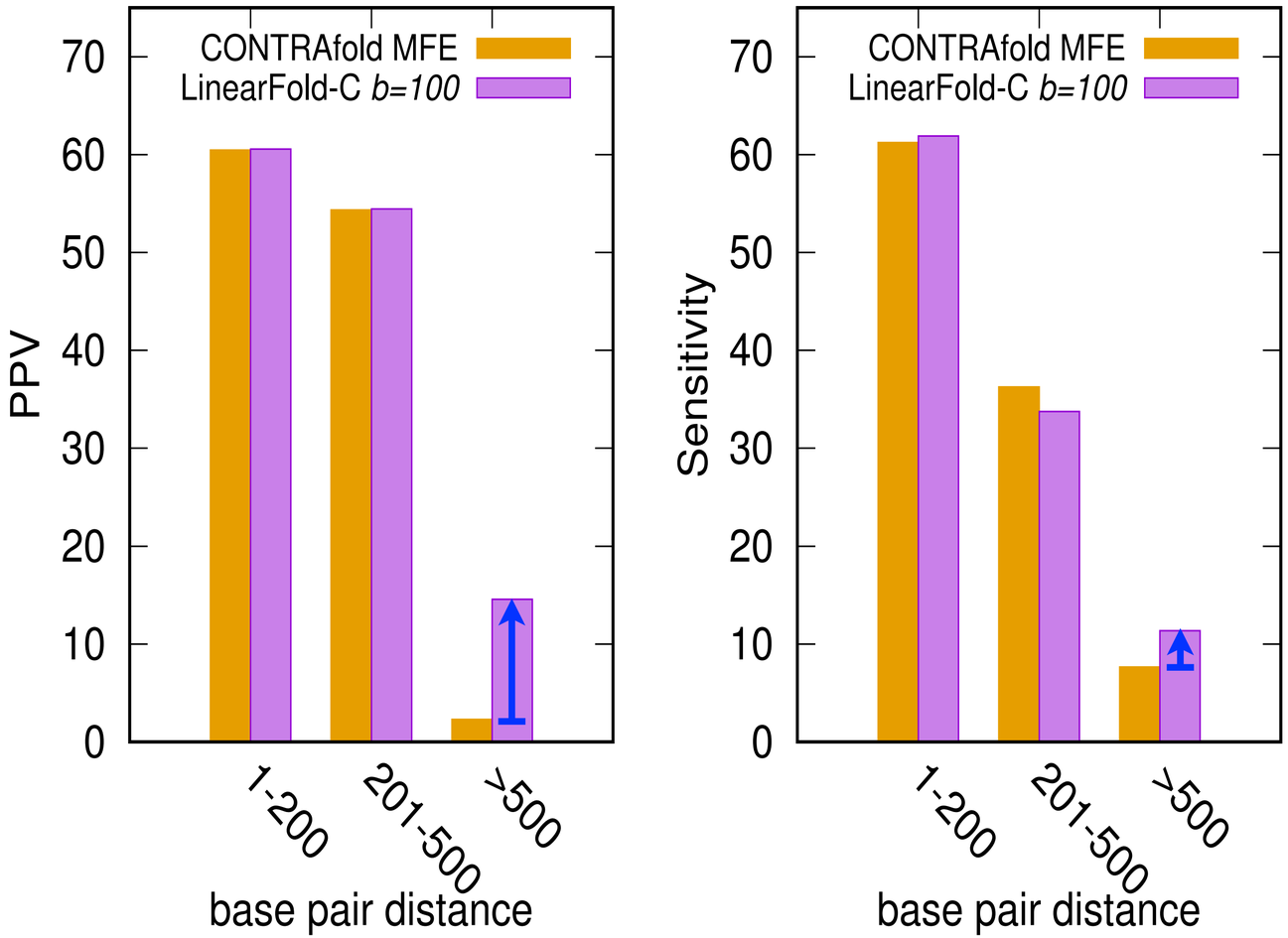
E. coli 23S rRNA
(2,904 *nt*)



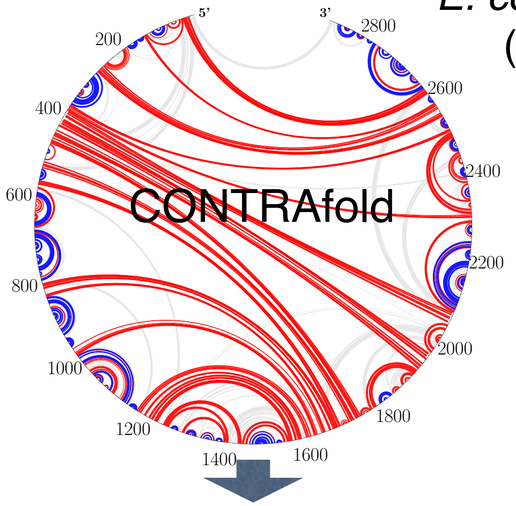


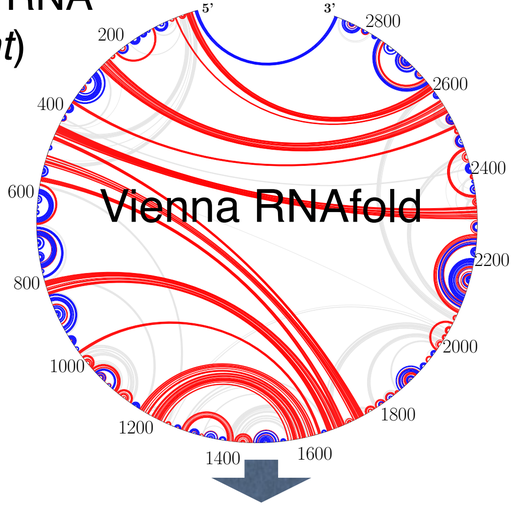
Improvements on Long-Range Base Pairs

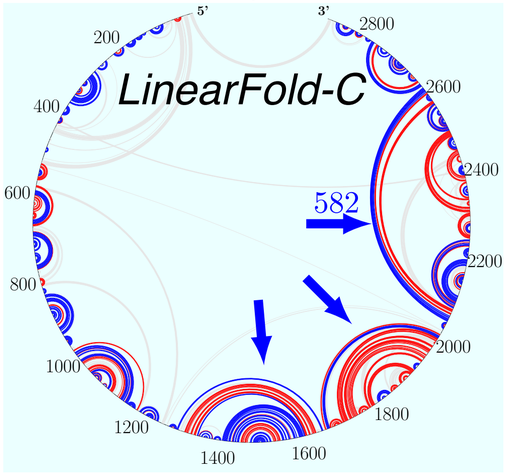
- long-distance pairs are well-known to be hard to predict
  - LinearFold outputs less long-range base pairs, but more correct ones

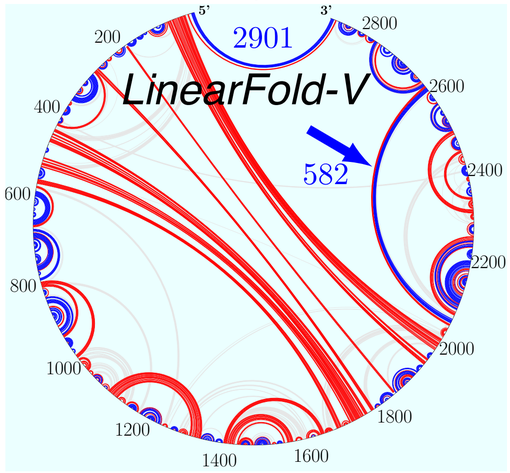


E. coli 23S rRNA
(2,904 *nt*)



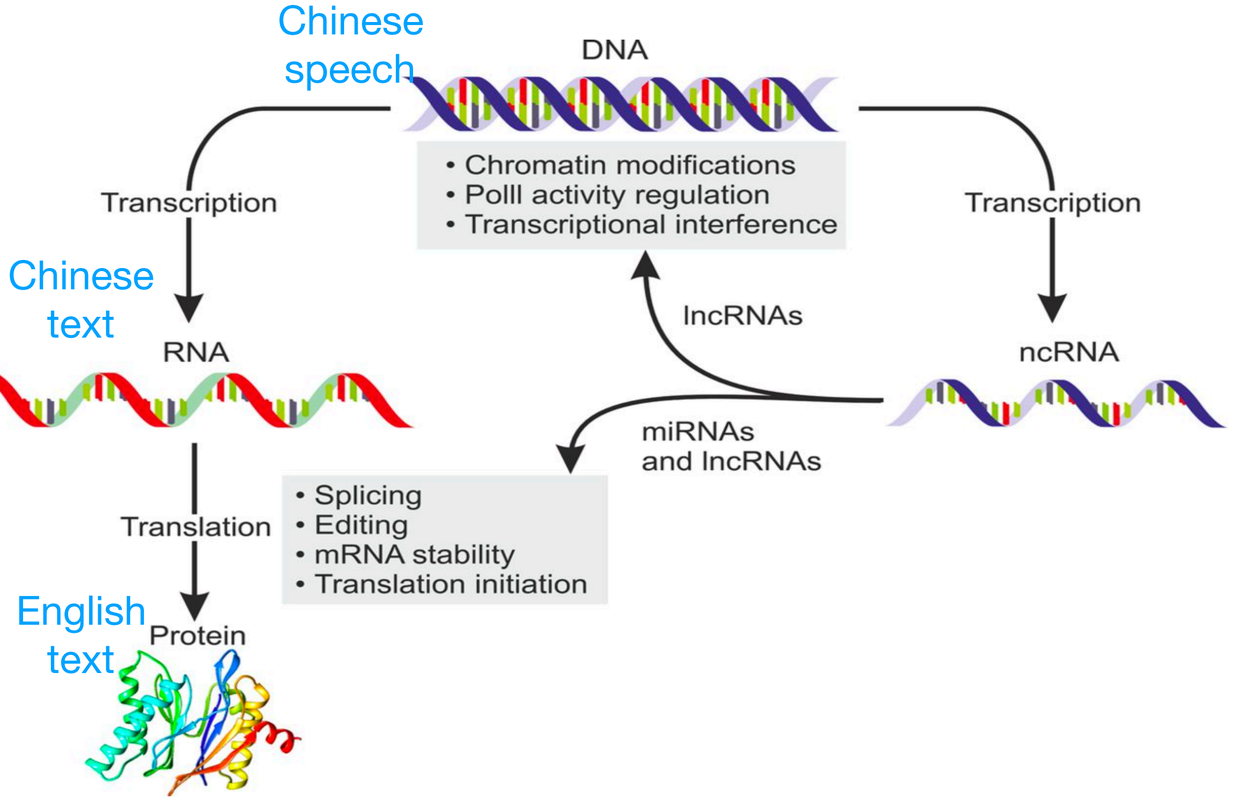


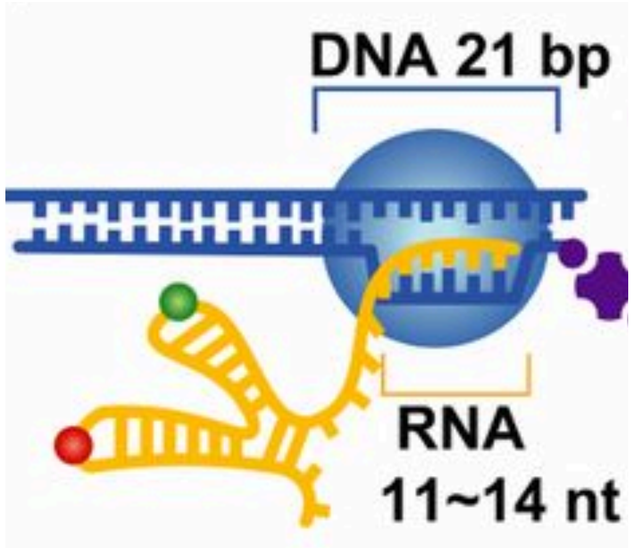


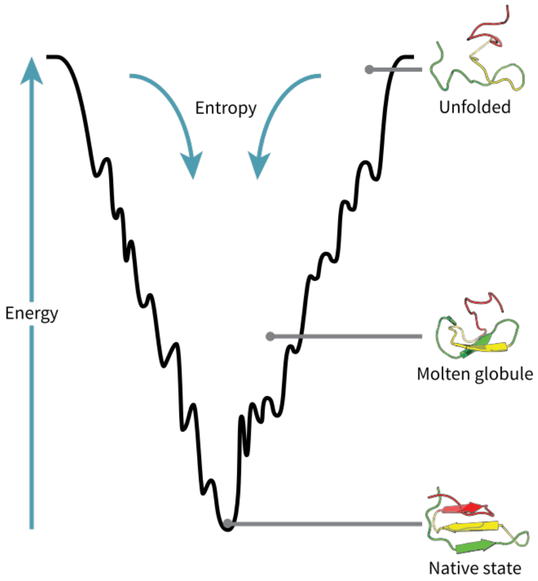


Incremental Parsing <=> Incremental Folding

- humans process sentences incrementally
- human language sentences *evolve* to be incrementally parsable
- RNAs & proteins fold while being assembled
- RNA & protein sequences evolve to be incrementally foldable
- these might explain why linear-time search performs better than exact search

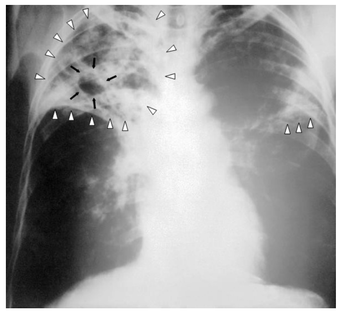






Fast Structure Prediction Enables RNA Design

detecting active TB using RNA design
which needs our fast RNA folding



Professor **Rhiju Das**
Stanford Medical School

EteRNA game
(RNA design)

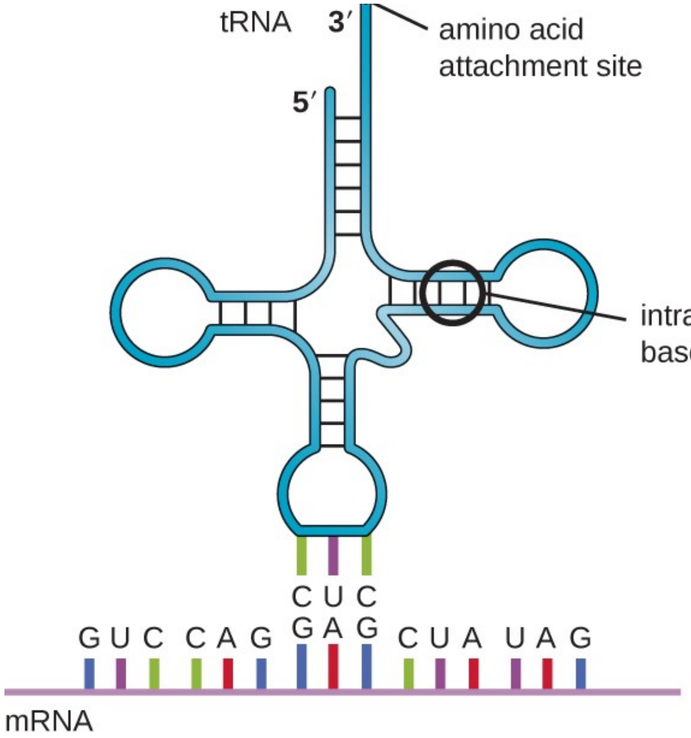
RNA sequence

GCGGGAAUAGCUCAGUUGGUAGAGCACGACCUUGCCAAGGUCGGGGUCGCGAGUUCGAGUCUCGUUUCCCGCUCCA

structure prediction ("folding")

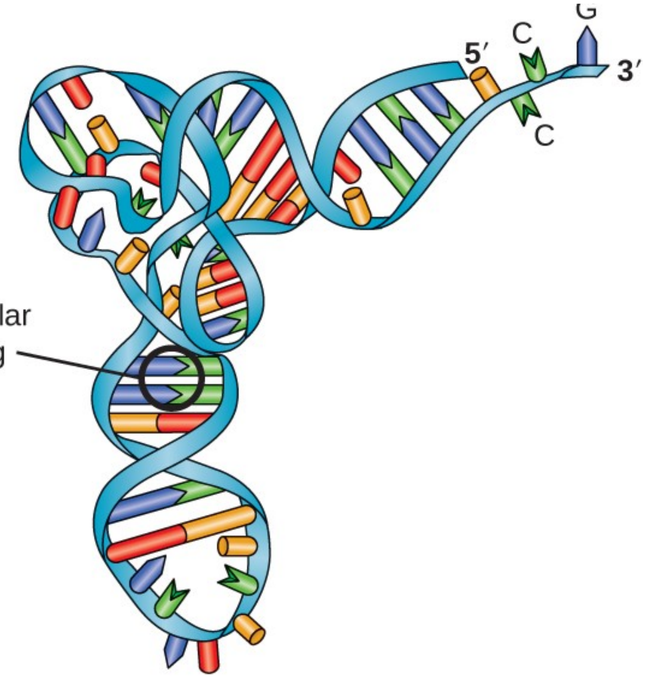
design

RNA secondary structure



tRNA 2-dimensional structure

RNA 3D structure

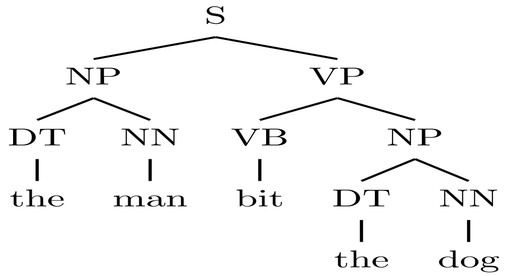


tRNA 3-dimensional structure

Computational Linguistics => Computational Biology

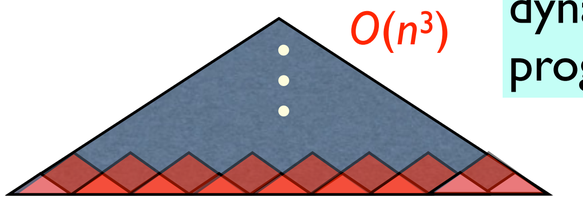
linguistics | **compiler theory** | **comp. linguistics** | **computational biology**

1955 Chomsky: context-free grammars (CFGs)



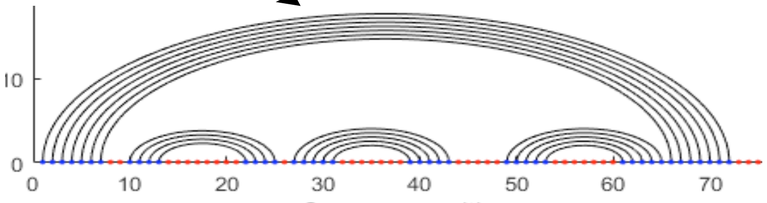
1958 Backus & Naur: CFGs for program, lang.

1964 **C**ocke \ bottom-up
1965 **K**asami - **CKY** $O(n^3)$
1967 **Y**ounger / for all CFGs



dynamic programming

1965 Knuth: **LR** parsing for restricted CFGs: $O(n)$



1978: Nussinov $O(n^3)$ RNA folding
1981: Zuker & Siegler

1986 Tomita: **G**eneralized **LR** for all CFGs: $O(n^3)$

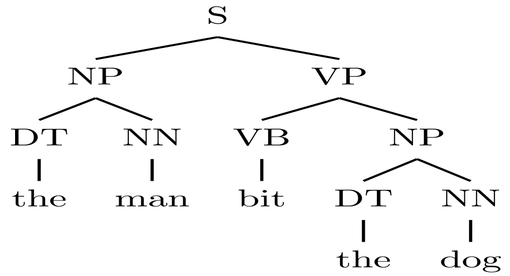
2010: Huang & Sagae: $O(n)$ (approx.) DP for all CFGs

2019: *LinearFold* $O(n)$ (approx.) RNA folding

Computational Linguistics => Computational Biology

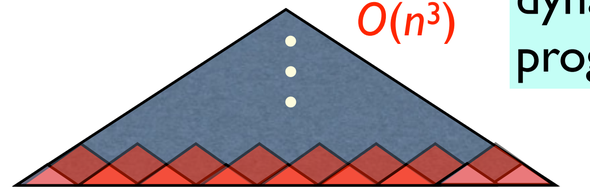
linguistics | **compiler theory** | **comp. linguistics** | **computational biology**

1955 Chomsky: context-free grammars (CFGs)



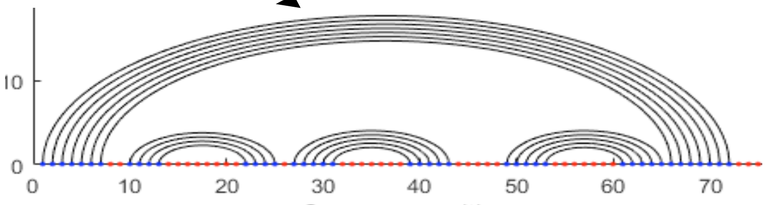
1958 Backus & Naur: CFGs for program, lang.

1964 **C**ocke \ bottom-up
1965 **K**asami - **CKY** $O(n^3)$
1967 **Y**ounger / for all CFGs



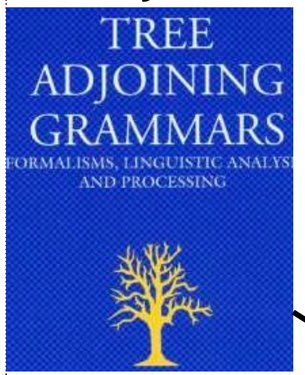
dynamic programming

1965 Knuth: **LR** parsing for restricted CFGs: $O(n)$



1978: Nussinov $O(n^3)$ RNA folding
1981: Zuker & Siegler

1970s Joshi:



1985 Shieber: NL is not CF

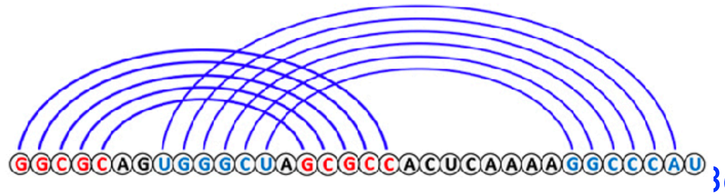
Swiss-German:
...mer em Hans es huss hälfed aastriiche

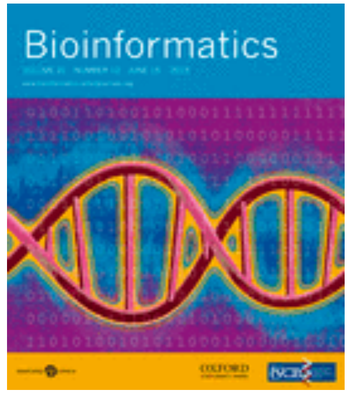
English:
...we helped Hans paint the house

1986 Tomita: **G**eneralized **LR** for all CFGs: $O(n^3)$

2010: Huang & Sagae: $O(n)$ (approx.) DP for all CFGs

1990s TAGs for RNA pseudoknots



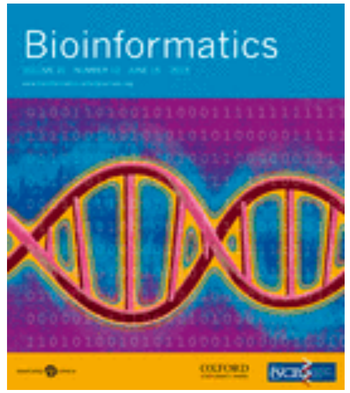


first linear-time (approximate) RNA folding algorithm

better in accuracy (esp. long seqs and long-range pairs)

http://linearfold.org

Thank you very much !



first linear-time (approximate) RNA folding algorithm

better in accuracy (esp. long seqs and long-range pairs)

http://linearfold.org

Oregon State University

Baidu Research

UNIVERSITY of ROCHESTER