CS 514 Algorithms

Spring 2025

Dynamic Programming 101

Tianshuo Zhou

May 1, 2025

Dynamic Programming 101

Dynamic Programming 101

- DP = recursion (divide-n-conquer) + caching (multiple divides, overlapping subproblems)

Dynamic Programming 101

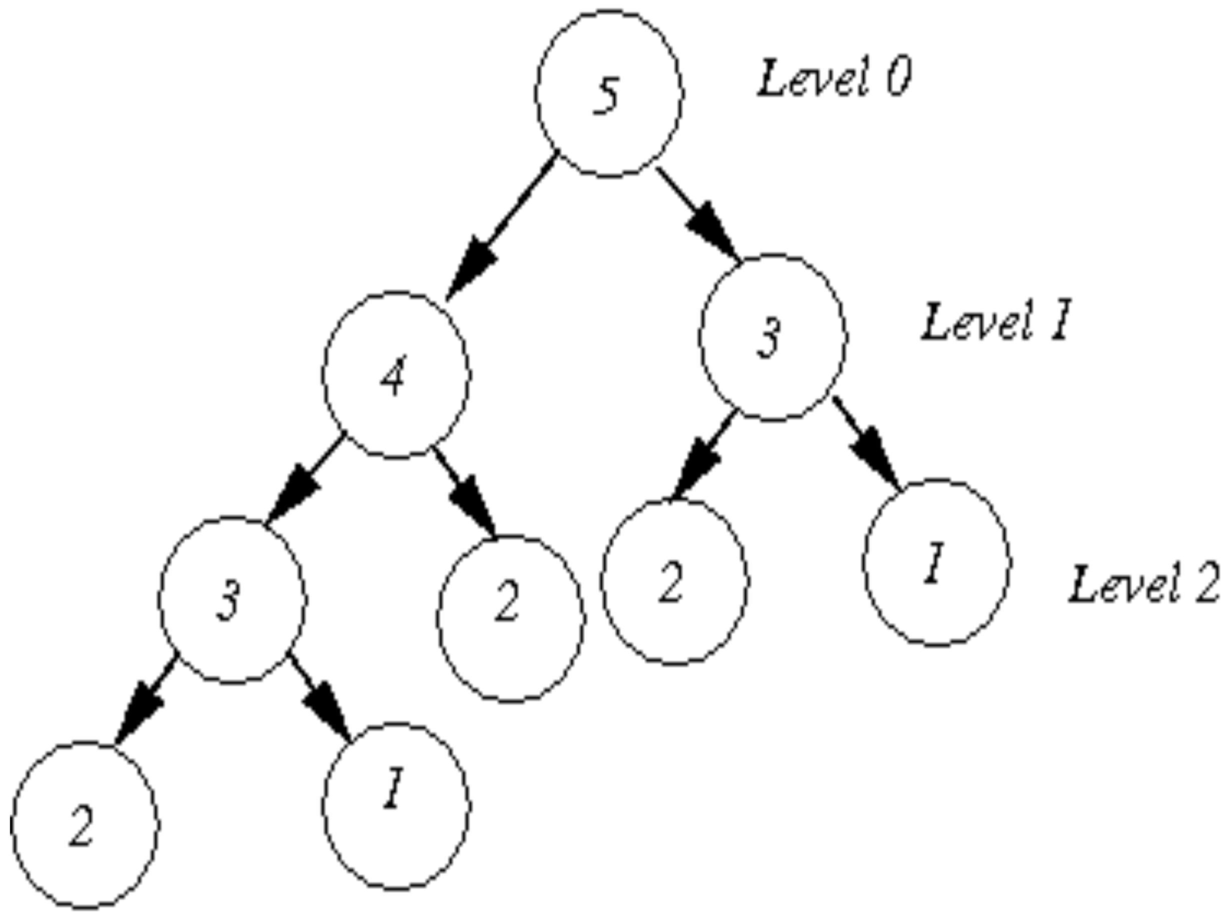
- DP = recursion (divide-n-conquer) + caching (multiple divides, overlapping subproblems)
- Fibonacci

$$f(n) = f(n-1) + f(n-2)$$
$$f(1) = f(2) = 1$$

Dynamic Programming 101

- DP = recursion (divide-n-conquer) + caching (multiple divides, overlapping subproblems)
- Fibonacci

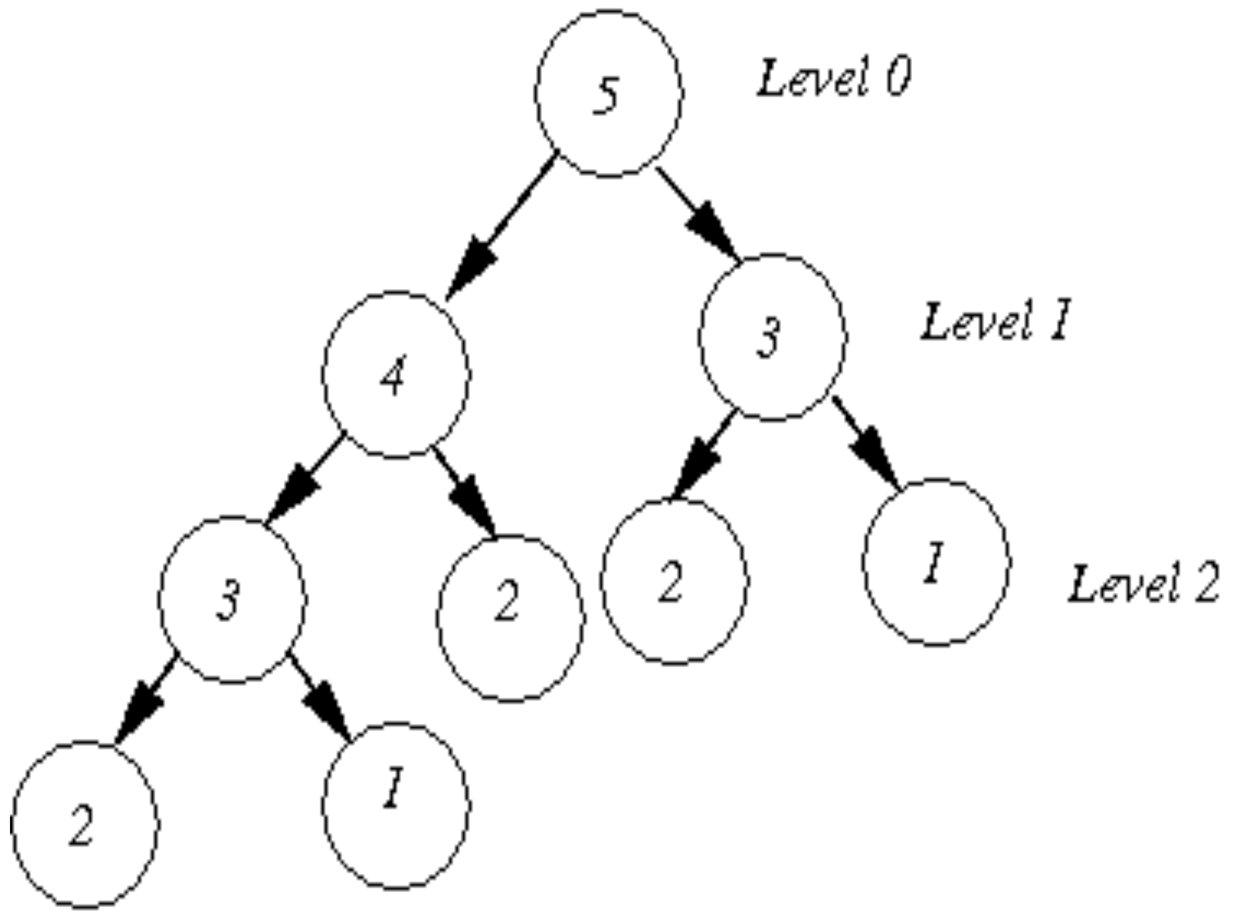
$$f(n) = f(n-1) + f(n-2)$$
$$f(1) = f(2) = 1$$



Dynamic Programming 101

- DP = recursion (divide-n-conquer) + caching (multiple divides, overlapping subproblems)
- Fibonacci

$$f(n) = f(n-1) + f(n-2)$$
$$f(1) = f(2) = 1$$



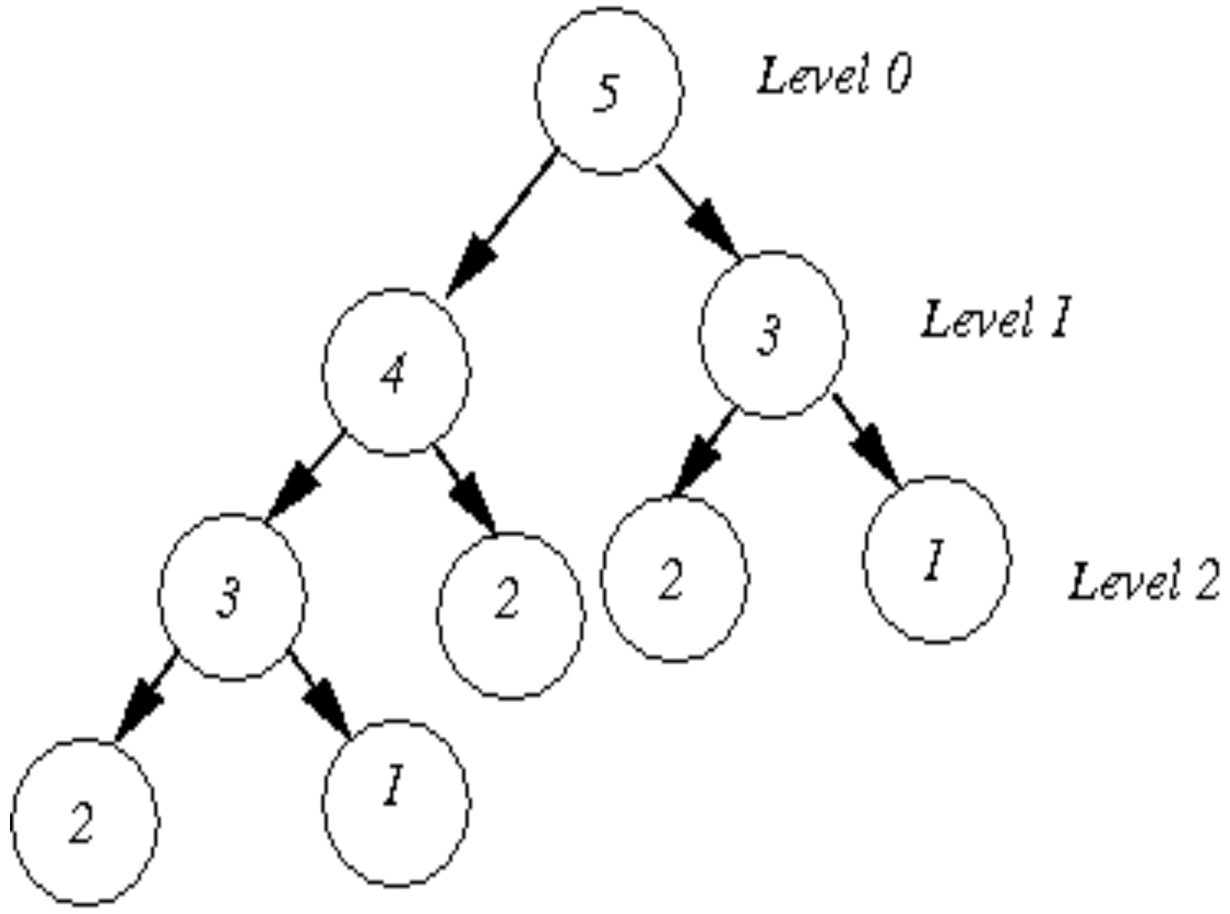
top-down with recursion: $O(1.618...^n)$

```
def fib(n):
    if n <= 2:
        return 1
    return fib(n-1) + fib(n-2)
```

Dynamic Programming 101

- DP = recursion (divide-n-conquer) + caching (multiple divides, overlapping subproblems)
- Fibonacci

$$f(n) = f(n-1) + f(n-2)$$
$$f(1) = f(2) = 1$$



i	1	2	3	4	5	…
$f(i)$	1	1	2	3	5	…

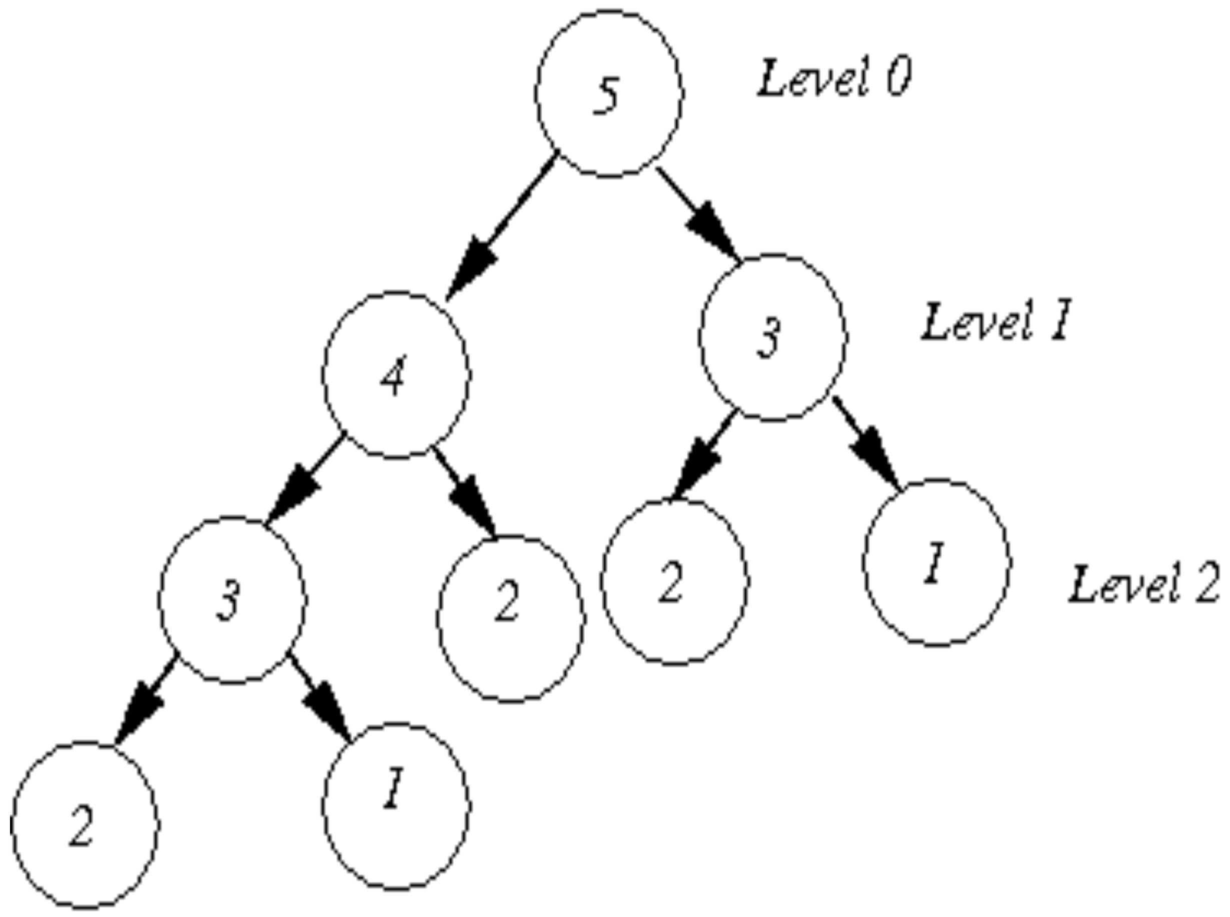
top-down with recursion: $O(1.618...^n)$

```
def fib(n):
    if n <= 2:
        return 1
    return fib(n-1) + fib(n-2)
```

Dynamic Programming 101

- DP = recursion (divide-n-conquer) + caching (multiple divides, overlapping subproblems)
- Fibonacci

$$f(n) = f(n-1) + f(n-2)$$
$$f(1) = f(2) = 1$$



i	1	2	3	4	5	...
$f(i)$	1	1	2	3	5	...

top-down with recursion: $O(1.618...^n)$

```
def fib(n):
    if n <= 2:
        return 1
    return fib(n-1) + fib(n-2)
```

bottom-up with memoization: $O(n)$

```
def fib0(n):
  f = [1, 1]
  for i in range(3, n+1):
    fibs.append(f[-1]+f[-2])
  return f[-1]
```

Number of Bitstrings

Number of Bitstrings

- number of n-bit strings that do not have 00 as a substring

Number of Bitstrings

- number of *n*-bit strings that do not have 00 as a substring
  - n=0: “”;

Number of Bitstrings

- number of n-bit strings that do not have 00 as a substring
  - n=0: “”;
  - n=1: 0, 1;

Number of Bitstrings

- number of n-bit strings that do not have 00 as a substring
  - n=0: “”;
  - n=1: 0, 1;
  - n=2: 01, 10, 11;

Number of Bitstrings

- number of n-bit strings that do not have 00 as a substring
  - n=0: “”;
  - n=1: 0, 1;
  - n=2: 01, 10, 11;
  - n=3: 010, 011, 101, 110, 111

Number of Bitstrings

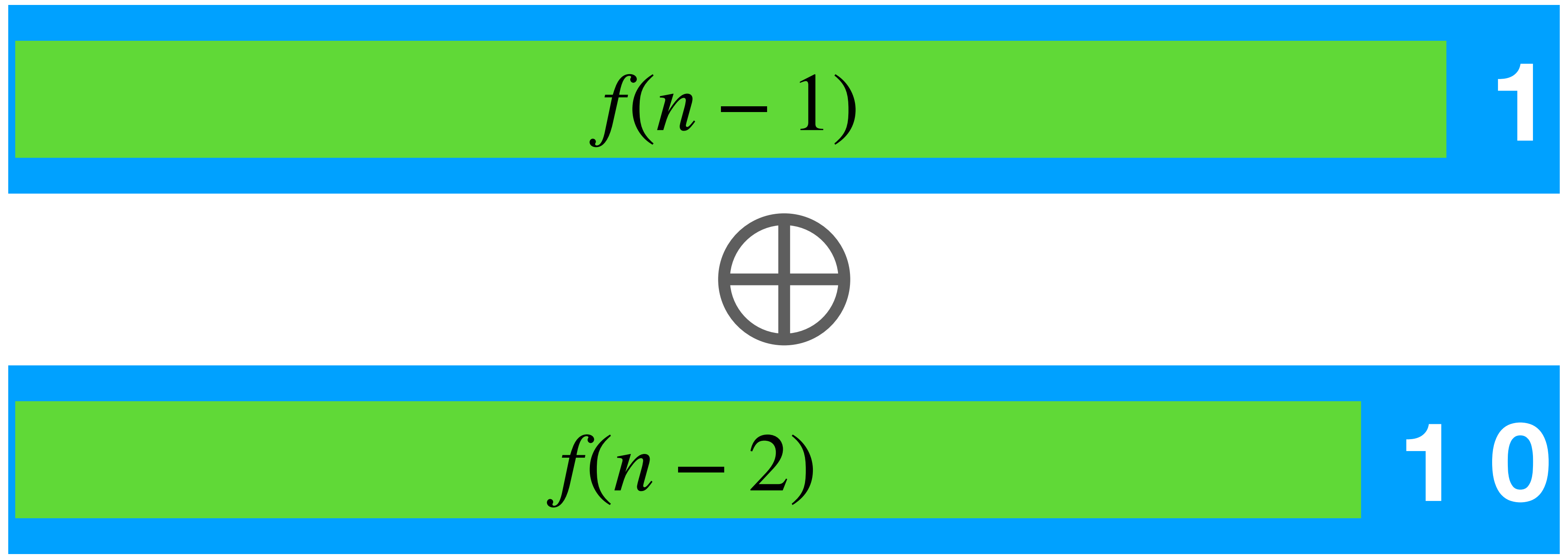
- number of *n*-bit strings that do not have 00 as a substring
  - n=0: “”;
  - n=1: 0, 1;
  - n=2: 01, 10, 11;
  - n=3: 010, 011, 101, 110, 111
  - last bit “1” followed by $f(n\text{-}1)$ substrings

Number of Bitstrings

- number of n-bit strings that do not have 00 as a substring
  - n=0: “”;
  - n=1: 0, 1;
  - n=2: 01, 10, 11;
  - n=3: 010, 011, 101, 110, 111
  - last bit “1” followed by $f(n\text{-}1)$ substrings
  - last two bits “01” followed by $f(n\text{-}2)$ substrings

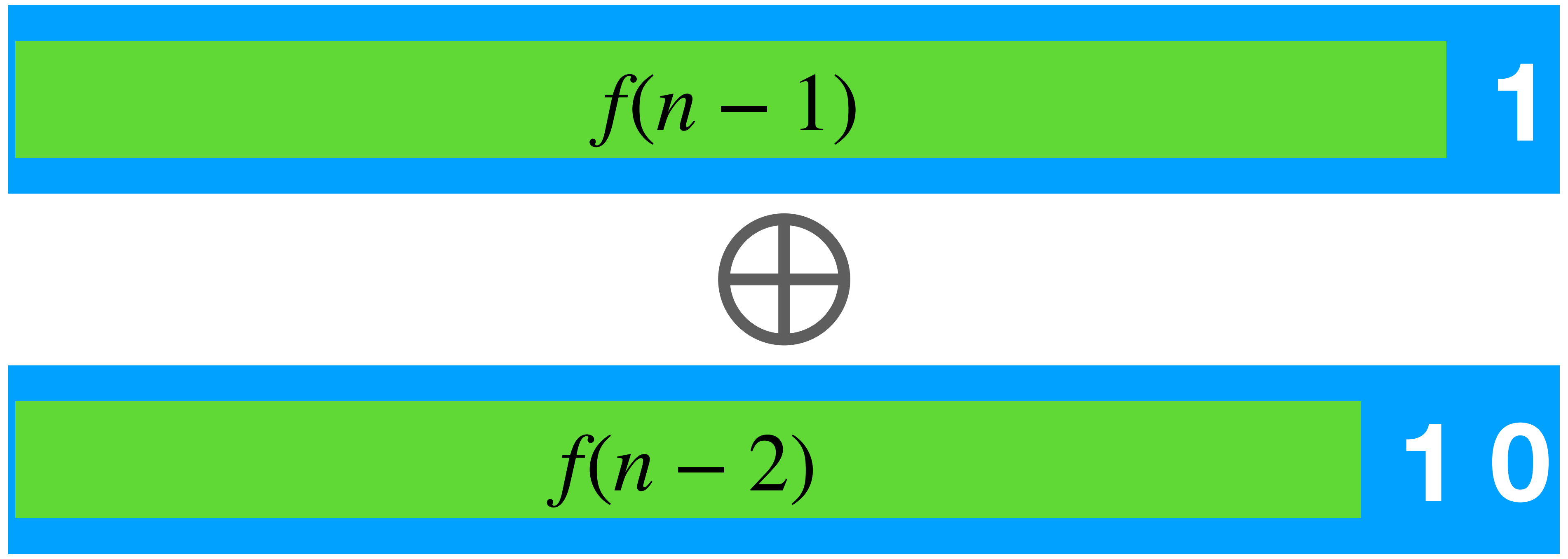
Number of Bitstrings

- number of n-bit strings that do not have 00 as a substring
  - n=0: “”;
  - n=1: 0, 1;
  - n=2: 01, 10, 11;
  - n=3: 010, 011, 101, 110, 111
  - last bit “1” followed by $f(n\text{-}1)$ substrings
  - last two bits “01” followed by $f(n\text{-}2)$ substrings



Number of Bitstrings

- number of *n*-bit strings that do not have 00 as a substring
  - n=0: “”;
  - *n*=1: 0, 1;
  - *n*=2: 01, 10, 11;
  - *n*=3: 010, 011, 101, 110, 111
  - last bit “1” followed by *f*(*n*-1) substrings
  - last two bits “01” followed by *f*(*n*-2) substrings



$$f(n) = f(n-1) + f(n-2)$$

$$f(1) = 2, \quad f(0) = 1$$

Max Independent Set (MIS)

Max Independent Set (MIS)

- max weighted independent set on a linear-chain graph
  - e.g. **9** — 10 — **8** — 5 — 2 — **4** ; best MIS: [9, 8, 4] = 21 (vs. greedy: [10, 5, 4] = 19)

Max Independent Set (MIS)

- max weighted independent set on a linear-chain graph
  - e.g. **9** — 10 — **8** — 5 — 2 — **4** ; best MIS: [9, 8, 4] = 21 (vs. greedy: [10, 5, 4] = 19)
- brute-force / exhaustive search
  - enumerate all possible (sub)sets, and evaluate independent ones
  - complexity: $O(2^n)$

Max Independent Set (MIS)

- max weighted independent set on a linear-chain graph
  - e.g. **9** — 10 — **8** — 5 — 2 — **4** ; best MIS: [9, 8, 4] = 21 (vs. greedy: [10, 5, 4] = 19)

Max Independent Set (MIS)

- max weighted independent set on a linear-chain graph
  - e.g. **9** — 10 — **8** — 5 — 2 — **4** ; best MIS: [9, 8, 4] = 21 (vs. greedy: [10, 5, 4] = 19)
- subproblem: $f(i)$ -- max independent set for a[1]..a[i] (1-based index)

$$f(i) = \max\{f(i-1),\ f(i-2) + a[i]\}$$

$f(0) = 0; f(1) = a[1]$?

No! $f(1) = \max\{a[1], 0\}$

Max Independent Set (MIS)

- max weighted independent set on a linear-chain graph
  - e.g. **9** — 10 — **8** — 5 — 2 — **4** ; best MIS: [9, 8, 4] = 21 (vs. greedy: [10, 5, 4] = 19)
- subproblem: *f*(*i*) -- max independent set for a[1]..a[*i*] (1-based index)

$$f(i) = \max\{f(i-1),\ f(i-2) + a[i]\}$$

$f(0) = 0; f(1) = a[1]?$

No! $f(1) = \max\{a[1], 0\}$

or even better: $f(0) = 0;\ f(-1) = 0$

Max Independent Set (MIS)

- max weighted independent set on a linear-chain graph
  - e.g. **9** — 10 — **8** — 5 — 2 — **4** ; best MIS: [9, 8, 4] = 21 (vs. greedy: [10, 5, 4] = 19)
- subproblem: *f*(*i*) -- max independent set for a[1]..a[*i*] (1-based index)

$$f(i) = \max\{f(i-1),\ f(i-2) + a[i]\}$$

$f(0) = 0; f(1) = a[1]$?

No! $f(1) = \max\{a[1], 0\}$

or even better: $f(0) = 0;\ f(-1) = 0$

  - top-down recursion without memoization
  - complexity: $O(1.618\ldots^n)$

Max Independent Set (MIS)

- max weighted independent set on a linear-chain graph
  - e.g. **9** — 10 — **8** — 5 — 2 — **4** ; best MIS: [9, 8, 4] = 21 (vs. greedy: [10, 5, 4] = 19)
- subproblem: *f*(*i*) -- max independent set for a[1]..a[*i*] (1-based index)

$$f(i) = \max\{f(i-1),\ f(i-2) + a[i]\}$$

$f(0) = 0;\ f(1) = a[1]$?

No! $f(1) = \max\{a[1], 0\}$

or even better: $f(0) = 0;\ f(-1) = 0$

Max Independent Set (MIS)

- max weighted independent set on a linear-chain graph
  - e.g. **9** — 10 — **8** — 5 — 2 — **4** ; best MIS: [9, 8, 4] = 21 (vs. greedy: [10, 5, 4] = 19)
- subproblem: *f*(*i*) -- max independent set for a[1]..a[*i*] (1-based index)

$$f(i) = \max\{f(i-1),\ f(i-2) + a[i]\}$$

$f(0) = 0;\ f(1) = a[1]$?

No! $f(1) = \max\{a[1], 0\}$

or even better: $f(0) = 0;\ f(-1) = 0$

i	-1	0	1	2	3	4	5	6
a[*i*]			9	10	8	5	2	4
f(*i*)								

best value

Max Independent Set (MIS)

- max weighted independent set on a linear-chain graph
  - e.g. **9** — 10 — **8** — 5 — 2 — **4** ; best MIS: [9, 8, 4] = 21 (vs. greedy: [10, 5, 4] = 19)
- subproblem: *f*(*i*) -- max independent set for a[1]..a[*i*] (1-based index)

$$f(i) = \max\{f(i-1),\ f(i-2) + a[i]\}$$

$f(0) = 0;\ f(1) = a[1]?$

No! $f(1) = \max\{a[1], 0\}$

or even better: $f(0) = 0;\ f(-1) = 0$

i	-1	0	1	2	3	4	5	6
a[*i*]			9	10	8	5	2	4
f(*i*)	0							

best value

Max Independent Set (MIS)

- max weighted independent set on a linear-chain graph
  - e.g. **9** — 10 — **8** — 5 — 2 — **4** ; best MIS: [9, 8, 4] = 21 (vs. greedy: [10, 5, 4] = 19)
- subproblem: *f*(*i*) -- max independent set for a[1]..a[*i*] (1-based index)

$$f(i) = \max\{f(i-1),\ f(i-2) + a[i]\}$$

$f(0) = 0;\ f(1) = a[1]$?

No! $f(1) = \max\{a[1], 0\}$

or even better: $f(0) = 0;\ f(-1) = 0$

i	-1	0	1	2	3	4	5	6
a[i]			9	10	8	5	2	4
f(i)	0	0						

best value

Max Independent Set (MIS)

- max weighted independent set on a linear-chain graph
  - e.g. **9** — 10 — **8** — 5 — 2 — **4** ; best MIS: [9, 8, 4] = 21 (vs. greedy: [10, 5, 4] = 19)
- subproblem: *f*(*i*) -- max independent set for a[1]..a[*i*] (1-based index)

$$f(i) = \max\{f(i-1),\ f(i-2) + a[i]\}$$

$f(0) = 0;\ f(1) = a[1]$?

No! $f(1) = \max\{a[1], 0\}$

or even better: $f(0) = 0;\ f(-1) = 0$

i	-1	0	1	2	3	4	5	6
a[*i*]			9	10	8	5	2	4
f(*i*)	0	0	9					

best value

Max Independent Set (MIS)

- max weighted independent set on a linear-chain graph
  - e.g. **9** — 10 — **8** — 5 — 2 — **4** ; best MIS: [9, 8, 4] = 21 (vs. greedy: [10, 5, 4] = 19)
- subproblem: *f*(*i*) -- max independent set for a[1]..a[*i*] (1-based index)

$$f(i) = \max\{f(i-1),\ f(i-2) + a[i]\}$$

$f(0) = 0; f(1) = a[1]$?

No! $f(1) = \max\{a[1], 0\}$

or even better: $f(0) = 0;\ f(-1) = 0$

i	-1	0	1	2	3	4	5	6
a[*i*]			9	10	8	5	2	4
f(*i*)	0	0	9	10				

best value

Max Independent Set (MIS)

- max weighted independent set on a linear-chain graph
  - e.g. **9** — 10 — **8** — 5 — 2 — **4** ; best MIS: [9, 8, 4] = 21 (vs. greedy: [10, 5, 4] = 19)
- subproblem: *f*(*i*) -- max independent set for a[1]..a[*i*] (1-based index)

$$f(i) = \max\{f(i-1),\ f(i-2) + a[i]\}$$

$f(0) = 0; f(1) = a[1]$?

No! $f(1) = \max\{a[1], 0\}$

or even better: $f(0) = 0;\ f(-1) = 0$

i	-1	0	1	2	3	4	5	6
a[i]			9	10	8	5	2	4
f(i)	0	0	9	10	17			

best value

Max Independent Set (MIS)

- max weighted independent set on a linear-chain graph
  - e.g. **9** — 10 — **8** — 5 — 2 — **4** ; best MIS: [9, 8, 4] = 21 (vs. greedy: [10, 5, 4] = 19)
- subproblem: $f(i)$ -- max independent set for a[1]..a[i] (1-based index)

$$f(i) = \max\{f(i-1),\ f(i-2) + a[i]\}$$

$f(0) = 0;\ f(1) = a[1]$?

No! $f(1) = \max\{a[1], 0\}$

or even better: $f(0) = 0;\ f(-1) = 0$

i	-1	0	1	2	3	4	5	6
a[i]			9	10	8	5	2	4
f(i)	0	0	9	10	17	17		

best value

Max Independent Set (MIS)

- max weighted independent set on a linear-chain graph
  - e.g. **9** — 10 — **8** — 5 — 2 — **4** ; best MIS: [9, 8, 4] = 21 (vs. greedy: [10, 5, 4] = 19)
- subproblem: $f(i)$ -- max independent set for a[1]..a[i] (1-based index)

$$f(i) = \max\{f(i-1),\ f(i-2) + a[i]\}$$

$f(0) = 0; f(1) = a[1]$?

No! $f(1) = \max\{a[1], 0\}$

or even better: $f(0) = 0;\ f(-1) = 0$

i	-1	0	1	2	3	4	5	6
a[i]			9	10	8	5	2	4
f(i)	0	0	9	10	17	17	19	

best value

Max Independent Set (MIS)

- max weighted independent set on a linear-chain graph
  - e.g. **9** — 10 — **8** — 5 — 2 — **4** ; best MIS: [9, 8, 4] = 21 (vs. greedy: [10, 5, 4] = 19)
- subproblem: *f*(*i*) -- max independent set for a[1]..a[*i*] (1-based index)

$$f(i) = \max\{f(i-1),\ f(i-2) + a[i]\}$$

$f(0) = 0;\ f(1) = a[1]$?

No! $f(1) = \max\{a[1], 0\}$

or even better: $f(0) = 0;\ f(-1) = 0$

i	-1	0	1	2	3	4	5	6
a[i]			9	10	8	5	2	4
f(i)	0	0	9	10	17	17	19	21

best value

Max Independent Set (MIS)

- max weighted independent set on a linear-chain graph
  - e.g. **9** — 10 — **8** — 5 — 2 — **4** ; best MIS: [9, 8, 4] = 21 (vs. greedy: [10, 5, 4] = 19)
- subproblem: *f*(*i*) -- max independent set for a[1]..a[*i*] (1-based index)

$$f(i) = \max\{f(i-1),\ f(i-2) + a[i]\}$$

$f(0) = 0;\ f(1) = a[1]$?

No! $f(1) = \max\{a[1], 0\}$

or even better: $f(0) = 0;\ f(-1) = 0$

i	-1	0	1	2	3	4	5	6
a[*i*]			9	10	8	5	2	4
f(*i*)	0	0	9	10	17	17	19	21

best value

- bottom-up with memoization
- complexity: $O(n)$

Max Independent Set (MIS)

- max weighted independent set on a linear-chain graph
  - e.g. **9** — 10 — **8** — 5 — 2 — **4** ; best MIS: [9, 8, 4] = 21 (vs. greedy: [10, 5, 4] = 19)
- subproblem: *f*(*i*) -- max independent set for a[1]..a[*i*] (1-based index)

$$f(i) = \max\{f(i-1),\ f(i-2) + a[i]\}$$

$f(0) = 0;\ f(1) = a[1]?$

No! $f(1) = \max\{a[1], 0\}$

or even better: $f(0) = 0;\ f(-1) = 0$

i	-1	0	1	2	3	4	5	6	
a[i]			9	10	8	5	2	4	
f(i)	0	0	9	10	17	17	19	21	*best value*
S(i)	{}	{}	{9}	{10}	{9,8}	{9,8}	{9,8,2}	{9,8,4}	*best set*

Max Independent Set (MIS)

- max weighted independent set on a linear-chain graph
  - e.g. **9** — 10 — **8** — 5 — 2 — **4** ; best MIS: [9, 8, 4] = 21 (vs. greedy: [10, 5, 4] = 19)
- subproblem: $f(i)$ -- max independent set for a[1]..a[i] (1-based index)

$$f(i) = \max\{f(i-1),\ f(i-2) + a[i]\}$$

$f(0) = 0;\ f(1) = a[1]$?

No! $f(1) = \max\{a[1], 0\}$

or even better: $f(0) = 0;\ f(-1) = 0$

Max Independent Set (MIS)

- max weighted independent set on a linear-chain graph
  - e.g. **9** — 10 — **8** — 5 — 2 — **4** ; best MIS: [9, 8, 4] = 21 (vs. greedy: [10, 5, 4] = 19)
- subproblem: *f*(*i*) -- max independent set for a[1]..a[*i*] (1-based index)

$$f(i) = \max\{f(i-1),\ f(i-2) + a[i]\}$$

$b(i) = [f(i) \neq f(i-1)]$: take $a[i]$ for $f(i)$?

$f(0) = 0;\ f(1) = a[1]$?

No! $f(1) = \max\{a[1], 0\}$

or even better: $f(0) = 0;\ f(-1) = 0$

Max Independent Set (MIS)

- max weighted independent set on a linear-chain graph
  - e.g. **9** — 10 — **8** — 5 — 2 — **4** ; best MIS: [9, 8, 4] = 21 (vs. greedy: [10, 5, 4] = 19)
- subproblem: $f(i)$ -- max independent set for a[1]..a[i] (1-based index)

$$f(i) = \max\{f(i-1),\ f(i-2) + a[i]\}$$

$b(i) = [f(i) \neq f(i-1)]$: take $a[i]$ for $f(i)$?

$f(0) = 0; f(1) = a[1]$?

No! $f(1) = \max\{a[1], 0\}$

or even better: $f(0) = 0;\ f(-1) = 0$

i	-1	0	1	2	3	4	5	6
a[i]			9	10	8	5	2	4
f(i)								

best value
backpointer

Max Independent Set (MIS)

- max weighted independent set on a linear-chain graph
  - e.g. **9** — 10 — **8** — 5 — 2 — **4** ; best MIS: [9, 8, 4] = 21 (vs. greedy: [10, 5, 4] = 19)
- subproblem: *f*(*i*) -- max independent set for a[1]..a[*i*] (1-based index)

$$f(i) = \max\{f(i-1),\ f(i-2) + a[i]\}$$

$$b(i) = [f(i) \neq f(i-1)] : \text{ take } a[i] \text{ for } f(i)?$$

$f(0) = 0; f(1) = a[1]$?
No! $f(1) = \max\{a[1], 0\}$
or even better: $f(0) = 0;\ f(-1) = 0$

i	-1	0	1	2	3	4	5	6
a[i]			9	10	8	5	2	4
f(i)	0							

best value
backpointer

Max Independent Set (MIS)

- max weighted independent set on a linear-chain graph
  - e.g. **9** — 10 — **8** — 5 — 2 — **4** ; best MIS: [9, 8, 4] = 21 (vs. greedy: [10, 5, 4] = 19)
- subproblem: $f(i)$ -- max independent set for a[1]..a[i] (1-based index)

$$f(i) = \max\{f(i-1),\ f(i-2) + a[i]\}$$

$$b(i) = [f(i) \neq f(i-1)] : \text{ take } a[i] \text{ for } f(i)?$$

$f(0) = 0;\ f(1) = a[1]$?

No! $f(1) = \max\{a[1], 0\}$

or even better: $f(0) = 0;\ f(-1) = 0$

i	-1	0	1	2	3	4	5	6
$a[i]$			9	10	8	5	2	4
$f(i)$	0	0						

best value

backpointer

Max Independent Set (MIS)

- max weighted independent set on a linear-chain graph
  - e.g. **9** — 10 — **8** — 5 — 2 — **4** ; best MIS: [9, 8, 4] = 21 (vs. greedy: [10, 5, 4] = 19)
- subproblem: *f*(*i*) -- max independent set for a[1]..a[*i*] (1-based index)

$$f(i) = \max\{f(i-1),\ f(i-2) + a[i]\}$$

$$b(i) = [f(i) \neq f(i-1)] : \text{ take } a[i] \text{ for } f(i)?$$

$f(0) = 0; f(1) = a[1]$?

No! $f(1) = \max\{a[1], 0\}$

or even better: $f(0) = 0;\ f(-1) = 0$

i	-1	0	1	2	3	4	5	6
a[i]			9	10	8	5	2	4
f(i)	0	0	9					

best value

backpointer

Max Independent Set (MIS)

- max weighted independent set on a linear-chain graph
  - e.g. **9** — 10 — **8** — 5 — 2 — **4** ; best MIS: [9, 8, 4] = 21 (vs. greedy: [10, 5, 4] = 19)
- subproblem: *f*(*i*) -- max independent set for a[1]..a[*i*] (1-based index)

$$f(i) = \max\{f(i-1),\ f(i-2) + a[i]\}$$

$$b(i) = [f(i) \neq f(i-1)]: \text{ take } a[i] \text{ for } f(i)?$$

$f(0) = 0; f(1) = a[1]$?

No! $f(1) = \max\{a[1], 0\}$

or even better: $f(0) = 0;\ f(-1) = 0$

i	-1	0	1	2	3	4	5	6
a[*i*]			9	10	8	5	2	4
f(*i*)	0	0	9	10				

best value

backpointer

Max Independent Set (MIS)

- max weighted independent set on a linear-chain graph
  - e.g. **9** — 10 — **8** — 5 — 2 — **4** ; best MIS: [9, 8, 4] = 21 (vs. greedy: [10, 5, 4] = 19)
- subproblem: *f*(*i*) -- max independent set for a[1]..a[*i*] (1-based index)

$$f(i) = \max\{f(i-1),\ f(i-2) + a[i]\}$$

$$b(i) = [f(i) \neq f(i-1)] :\ \text{take } a[i] \text{ for } f(i)?$$

$f(0) = 0;\ f(1) = a[1]$?

No! $f(1) = \max\{a[1], 0\}$

or even better: $f(0) = 0;\ f(-1) = 0$

i	-1	0	1	2	3	4	5	6
a[i]			9	10	8	5	2	4
f(i)	0	0	9	10	17			

best value

backpointer

Max Independent Set (MIS)

- max weighted independent set on a linear-chain graph
  - e.g. **9** — 10 — **8** — 5 — 2 — **4** ; best MIS: [9, 8, 4] = 21 (vs. greedy: [10, 5, 4] = 19)
- subproblem: *f*(*i*) -- max independent set for a[1]..a[*i*] (1-based index)

$$f(i) = \max\{f(i-1),\ f(i-2) + a[i]\}$$

$$b(i) = [f(i) \neq f(i-1)] : \text{ take } a[i] \text{ for } f(i)?$$

$f(0) = 0;\ f(1) = a[1]$?

No! $f(1) = \max\{a[1], 0\}$

or even better: $f(0) = 0;\ f(-1) = 0$

i	-1	0	1	2	3	4	5	6
a[i]			9	10	8	5	2	4
f(i)	0	0	9	10	17	17		

best value
backpointer

Max Independent Set (MIS)

- max weighted independent set on a linear-chain graph
  - e.g. **9** — 10 — **8** — 5 — 2 — **4** ; best MIS: [9, 8, 4] = 21 (vs. greedy: [10, 5, 4] = 19)
- subproblem: *f*(*i*) -- max independent set for a[1]..a[*i*] (1-based index)

$$f(i) = \max\{f(i-1),\ f(i-2) + a[i]\}$$

$$b(i) = [f(i) \neq f(i-1)] :\ \text{take } a[i] \text{ for } f(i)?$$

$f(0) = 0;\ f(1) = a[1]$?

No! $f(1) = \max\{a[1], 0\}$

or even better: $f(0) = 0;\ f(-1) = 0$

i	-1	0	1	2	3	4	5	6
a[i]			9	10	8	5	2	4
f(i)	0	0	9	10	17	17	19	

best value

backpointer

Max Independent Set (MIS)

- max weighted independent set on a linear-chain graph
  - e.g. **9** — 10 — **8** — 5 — 2 — **4** ; best MIS: [9, 8, 4] = 21 (vs. greedy: [10, 5, 4] = 19)
- subproblem: *f*(*i*) -- max independent set for a[1]..a[*i*] (1-based index)

$$f(i) = \max\{f(i-1),\ f(i-2) + a[i]\}$$

$$b(i) = [f(i) \neq f(i-1)] : \text{ take } a[i] \text{ for } f(i)?$$

$f(0) = 0; f(1) = a[1]$?

No! $f(1) = \max\{a[1], 0\}$

or even better: $f(0) = 0;\ f(-1) = 0$

i	-1	0	1	2	3	4	5	6	
a[*i*]			9	10	8	5	2	4	
f(*i*)	0	0	9	10	17	17	19	21	*best value*
b(*i*)									*backpointer*

Max Independent Set (MIS)

- max weighted independent set on a linear-chain graph
  - e.g. **9** — 10 — **8** — 5 — 2 — **4** ; best MIS: [9, 8, 4] = 21 (vs. greedy: [10, 5, 4] = 19)
- subproblem: *f*(*i*) -- max independent set for a[1]..a[*i*] (1-based index)

$$f(i) = \max\{f(i-1),\ f(i-2) + a[i]\}$$

$b(i) = [f(i) \neq f(i-1)]$: take $a[i]$ for $f(i)$?

$f(0) = 0;\ f(1) = a[1]$?

No! $f(1) = \max\{a[1], 0\}$

or even better: $f(0) = 0;\ f(-1) = 0$

i	-1	0	1	2	3	4	5	6
a[*i*]			9	10	8	5	2	4
f(*i*)	0	0	9	10	17	17	19	21
b(*i*)			T					

best value
backpointer

Max Independent Set (MIS)

- max weighted independent set on a linear-chain graph
  - e.g. **9** — 10 — **8** — 5 — 2 — **4** ; best MIS: [9, 8, 4] = 21 (vs. greedy: [10, 5, 4] = 19)
- subproblem: *f*(*i*) -- max independent set for a[1]..a[*i*] (1-based index)

$$f(i) = \max\{f(i-1),\ f(i-2) + a[i]\}$$

$$b(i) = [f(i) \neq f(i-1)] : \text{ take } a[i] \text{ for } f(i)?$$

$f(0) = 0;\ f(1) = a[1]$?

No! $f(1) = \max\{a[1], 0\}$

or even better: $f(0) = 0;\ f(-1) = 0$

i	-1	0	1	2	3	4	5	6
a[i]			9	10	8	5	2	4
f(i)	0	0	9	10	17	17	19	21
b(i)			T	T				

best value

backpointer

Max Independent Set (MIS)

- max weighted independent set on a linear-chain graph
  - e.g. **9** — 10 — **8** — 5 — 2 — **4** ; best MIS: [9, 8, 4] = 21 (vs. greedy: [10, 5, 4] = 19)
- subproblem: $f(i)$ -- max independent set for a[1]..a[i] (1-based index)

$$f(i) = \max\{f(i-1),\ f(i-2) + a[i]\}$$

$$b(i) = [f(i) \neq f(i-1)] : \text{ take } a[i] \text{ for } f(i)?$$

$f(0) = 0;\ f(1) = a[1]$?

No! $f(1) = \max\{a[1], 0\}$

or even better: $f(0) = 0;\ f(-1) = 0$

i	-1	0	1	2	3	4	5	6
$a[i]$			9	10	8	5	2	4
$f(i)$	0	0	9	10	17	17	19	21
$b(i)$			T	T	T			

best value

backpointer

Max Independent Set (MIS)

- max weighted independent set on a linear-chain graph
  - e.g. **9** — 10 — **8** — 5 — 2 — **4** ; best MIS: [9, 8, 4] = 21 (vs. greedy: [10, 5, 4] = 19)
- subproblem: *f*(*i*) -- max independent set for a[1]..a[*i*] (1-based index)

$$f(i) = \max\{f(i-1),\ f(i-2) + a[i]\}$$

$$b(i) = [f(i) \neq f(i-1)] : \text{ take } a[i] \text{ for } f(i)?$$

$f(0) = 0;\ f(1) = a[1]$?

No! $f(1) = \max\{a[1], 0\}$

or even better: $f(0) = 0;\ f(-1) = 0$

i	-1	0	1	2	3	4	5	6	
a[*i*]			9	10	8	5	2	4	
f(*i*)	0	0	9	10	17	17	19	21	*best value*
b(*i*)			T	T	T	F	T	T	*backpointer*

Max Independent Set (MIS)

- max weighted independent set on a linear-chain graph
  - e.g. **9** — 10 — **8** — 5 — 2 — **4** ; best MIS: [9, 8, 4] = 21 (vs. greedy: [10, 5, 4] = 19)
- subproblem: *f*(*i*) -- max independent set for a[1]..a[*i*] (1-based index)

$$f(i) = \max\{f(i-1),\ f(i-2) + a[i]\}$$

$$b(i) = [f(i) \neq f(i-1)] : \text{ take } a[i] \text{ for } f(i)?$$

$f(0) = 0; f(1) = a[1]$?
No! $f(1) = \max\{a[1], 0\}$
or even better: $f(0) = 0;\ f(-1) = 0$

i	-1	0	1	2	3	4	5	6
a[i]			9	10	8	5	2	4
f(i)	0	0	9	10	17	17	19	21
b(i)			T	T	T	F	T	T

best value
backpointer
start here

recursively backtrack
the optimal solution

Max Independent Set (MIS)

- max weighted independent set on a linear-chain graph
  - e.g. **9** — 10 — **8** — 5 — 2 — **4** ; best MIS: [9, 8, 4] = 21 (vs. greedy: [10, 5, 4] = 19)
- subproblem: *f*(*i*) -- max independent set for a[1]..a[*i*] (1-based index)

$$f(i) = \max\{f(i-1),\ f(i-2) + a[i]\}$$

$b(i) = [f(i) \neq f(i-1)]$: take $a[i]$ for $f(i)$?

$f(0) = 0; f(1) = a[1]$?

No! $f(1) = \max\{a[1], 0\}$

or even better: $f(0) = 0;\ f(-1) = 0$

i	-1	0	1	2	3	4	5	6	
a[i]			9	10	8	5	2	4	
f(i)	0	0	9	10	17	17	19	21	*best value*
b(i)			T	T	T	F	T	T	*backpointer*
									← *start here*

recursively backtrack
the optimal solution

Max Independent Set (MIS)

- max weighted independent set on a linear-chain graph
  - e.g. **9** — 10 — **8** — 5 — 2 — **4** ; best MIS: [9, 8, 4] = 21 (vs. greedy: [10, 5, 4] = 19)
- subproblem: *f*(*i*) -- max independent set for a[1]..a[*i*] (1-based index)

$$f(i) = \max\{f(i-1),\ f(i-2) + a[i]\}$$

$b(i) = [f(i) \neq f(i-1)]$: take $a[i]$ for $f(i)$?

$f(0) = 0;\ f(1) = a[1]$?
No! $f(1) = \max\{a[1], 0\}$
or even better: $f(0) = 0;\ f(-1) = 0$

i	-1	0	1	2	3	4	5	6	
a[i]			9	10	8	5	2	4	
f(i)	0	0	9	10	17	17	19	21	*best value*
b(i)			T	T	T	F	T	T	*backpointer*
back track	*		* take		* take	* not		* take	← *start here*

recursively backtrack
the optimal solution

Max Independent Set (MIS)

- max weighted independent set on a linear-chain graph
  - e.g. **9** — 10 — **8** — 5 — 2 — **4** ; best MIS: [9, 8, 4] = 21 (vs. greedy: [10, 5, 4] = 19)
- subproblem: *f*(*i*) -- max independent set for a[1]..a[*i*] (1-based index)

$$f(i) = \max\{f(i-1),\ f(i-2) + a[i]\}$$

$$b(i) = [f(i) \neq f(i-1)] : \text{ take } a[i] \text{ for } f(i)?$$

$f(0) = 0; f(1) = a[1]$?

No! $f(1) = \max\{a[1], 0\}$

or even better: $f(0) = 0;\ f(-1) = 0$

i	-1	0	1	2	3	4	5	6
a[i]			9	10	8	5	2	4
f(i)	0	0	9	10	17	17	19	21
b(i)			T	T	T	F	T	T
back track	*		* take		* take	* not		* take

best value

backpointer

start here

recursively backtrack the optimal solution

MIS

$$f(n) = \max \begin{cases} f(n-1) + 0 \\ f(n-2) + a[n] \end{cases}$$

bitstrings

$$f(n) = + \begin{cases} f(n-1) \times 1 \\ f(n-2) \times 1 \end{cases}$$

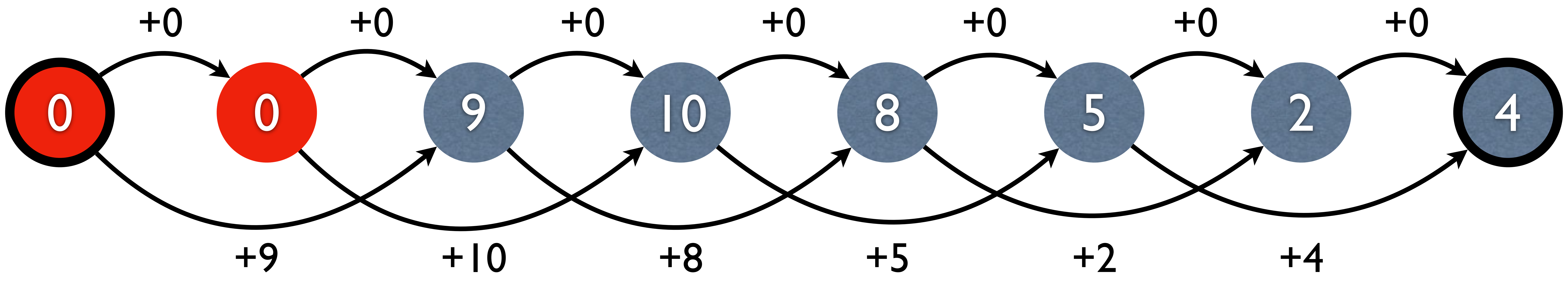
cost

reward

summary operator ⊕ (across divides)

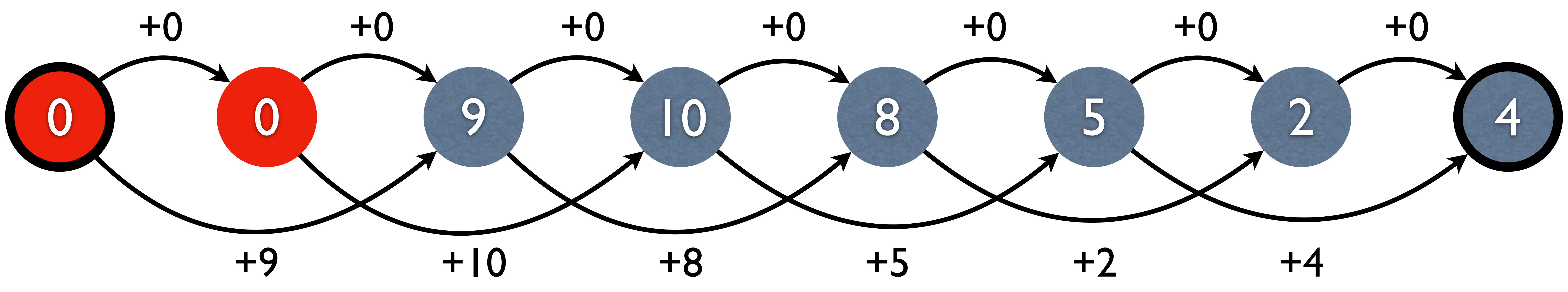
combination operator ⊗ (within a divide)

Graph Interpretation of DP



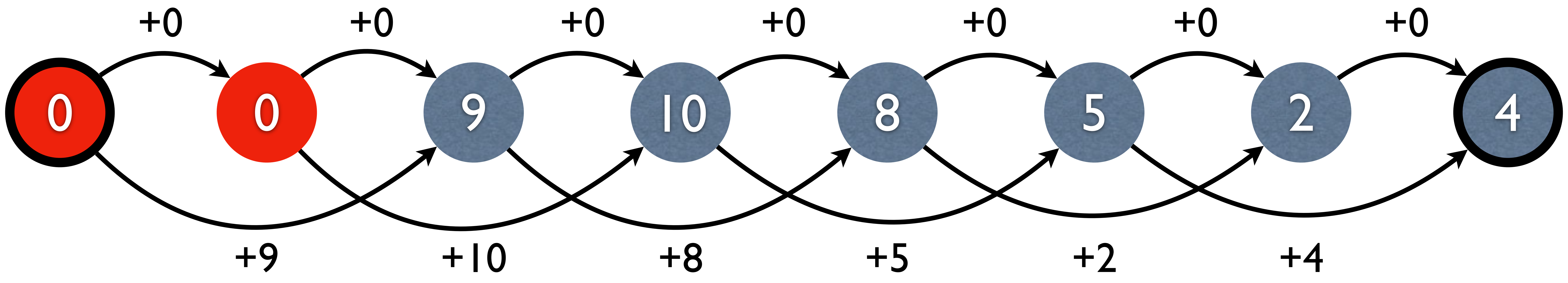
Graph Interpretation of DP

- MIS: longest path between source and target (see lecture video)
  - each node i has two incoming edges: $(i-2) \xrightarrow{a[i]} i$ (take) and $(i-1) \xrightarrow{0} i$ (not take)



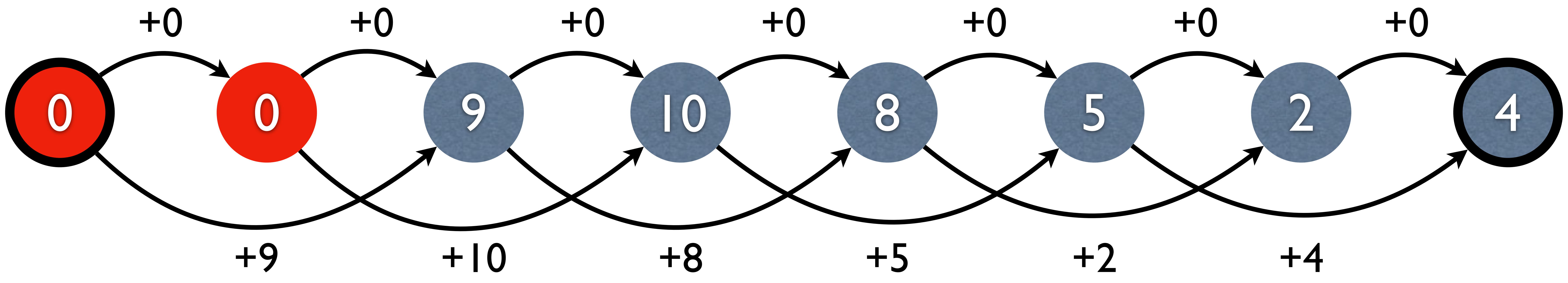
Graph Interpretation of DP

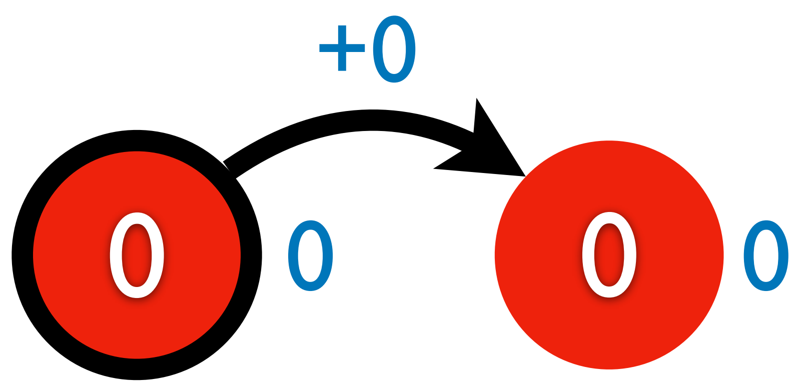
- MIS: longest path between source and target (see lecture video)
  - each node i has two incoming edges: $(i-2) \xrightarrow{a[i]} i$ (take) and $(i-1) \xrightarrow{0} i$ (not take)
  - $f(i)$: longest path between source and node i



Graph Interpretation of DP

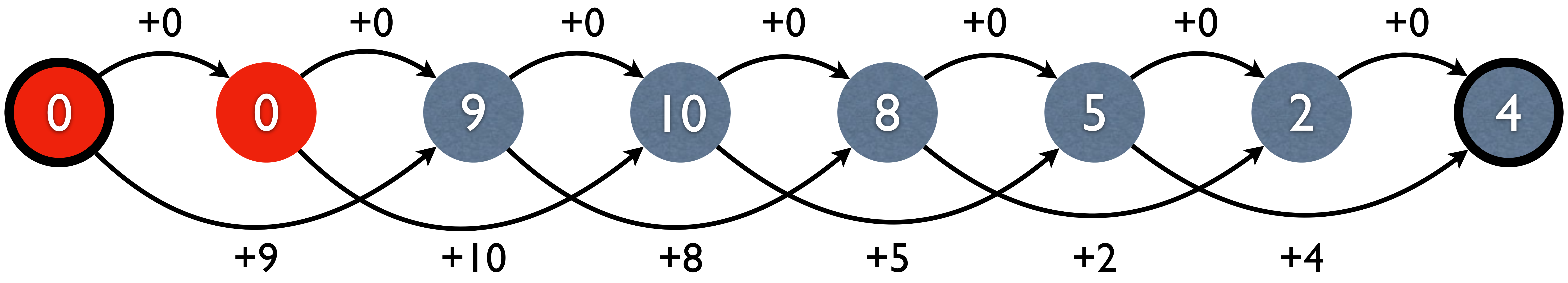
- MIS: longest path between source and target (see lecture video)
  - each node i has two incoming edges: $(i-2) \xrightarrow{a[i]} i$ (take) and $(i-1) \xrightarrow{0} i$ (not take)
  - $f(i)$: longest path between source and node i

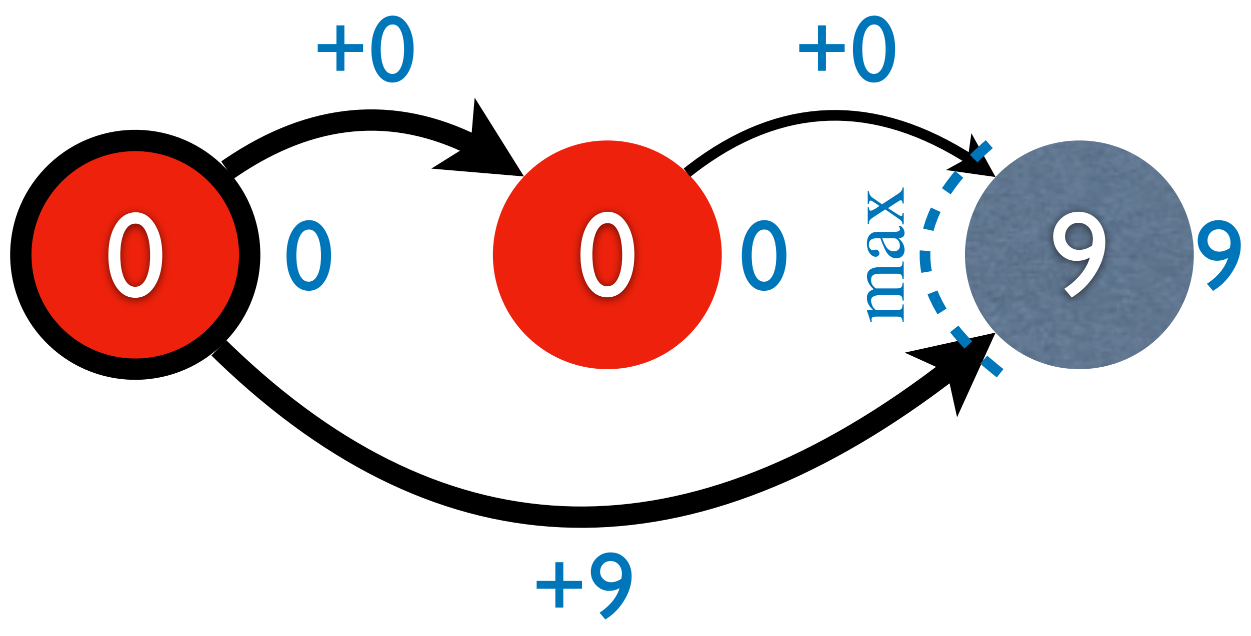




Graph Interpretation of DP

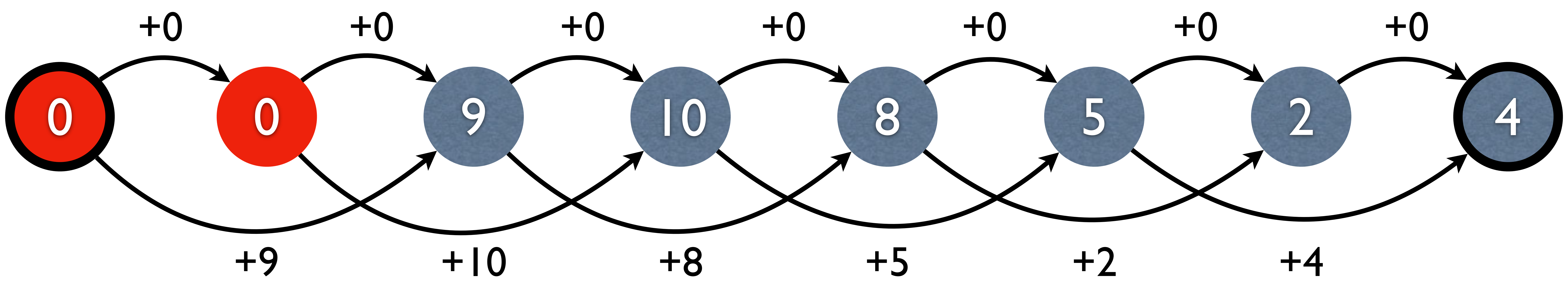
- MIS: longest path between source and target (see lecture video)
  - each node i has two incoming edges: $(i-2) \xrightarrow{a[i]} i$ (take) and $(i-1) \xrightarrow{0} i$ (not take)
  - $f(i)$: longest path between source and node i

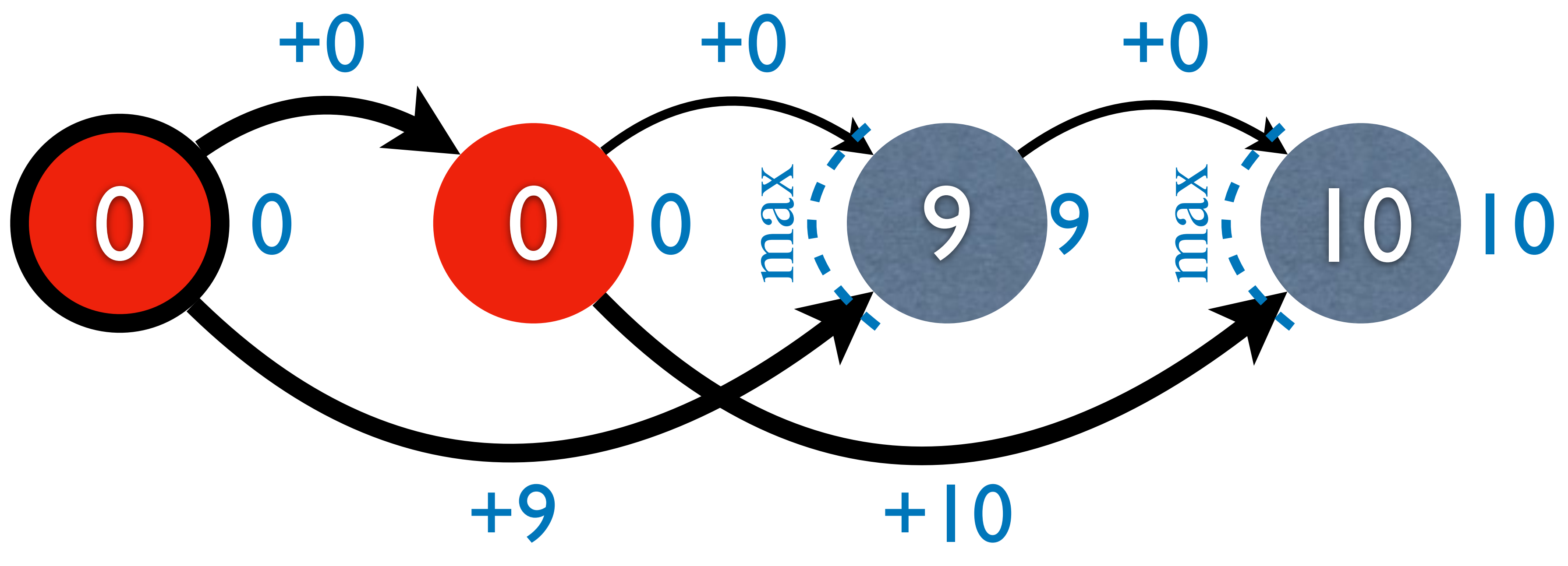




Graph Interpretation of DP

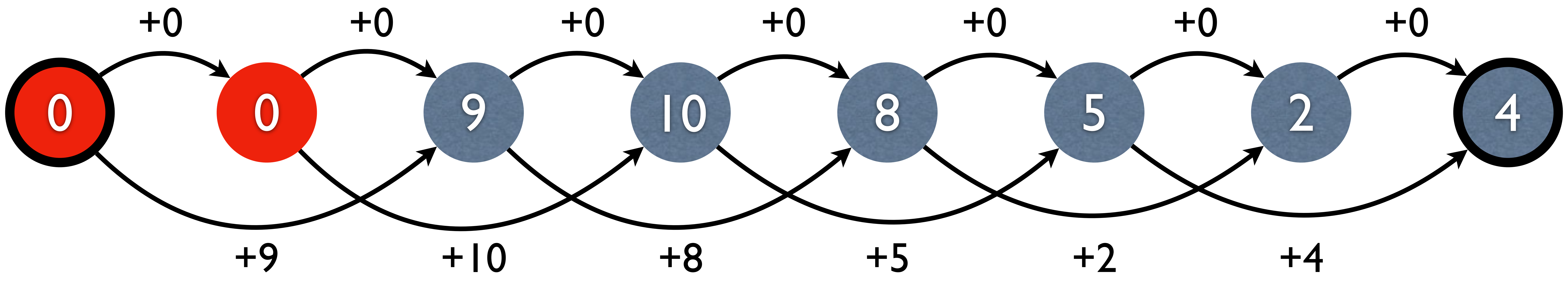
- MIS: longest path between source and target (see lecture video)
  - each node i has two incoming edges: $(i-2) \xrightarrow{a[i]} i$ (take) and $(i-1) \xrightarrow{0} i$ (not take)
  - $f(i)$: longest path between source and node i

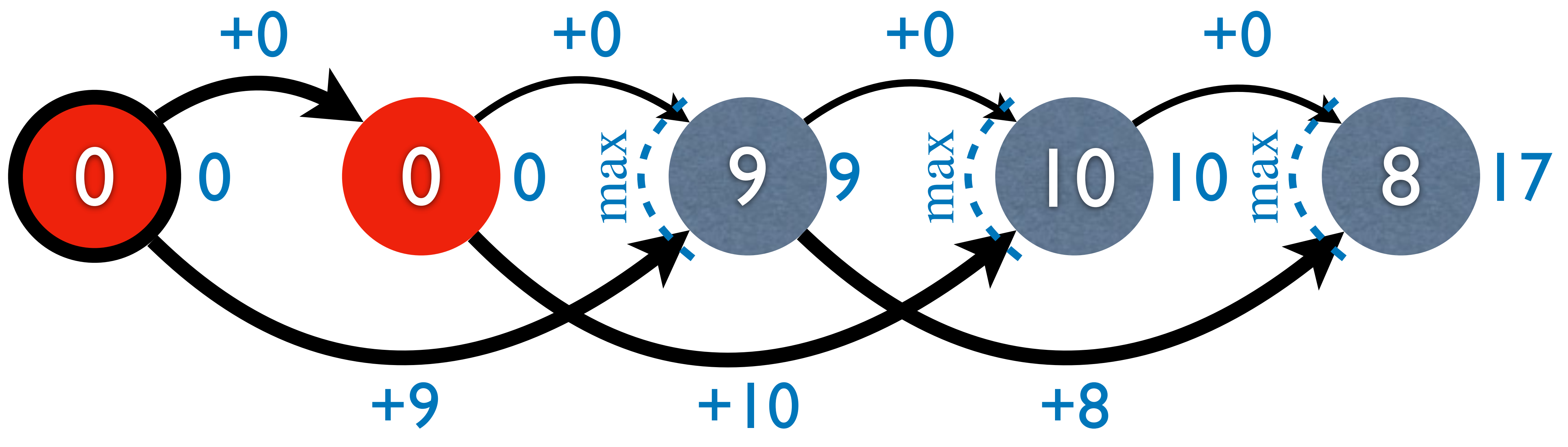




Graph Interpretation of DP

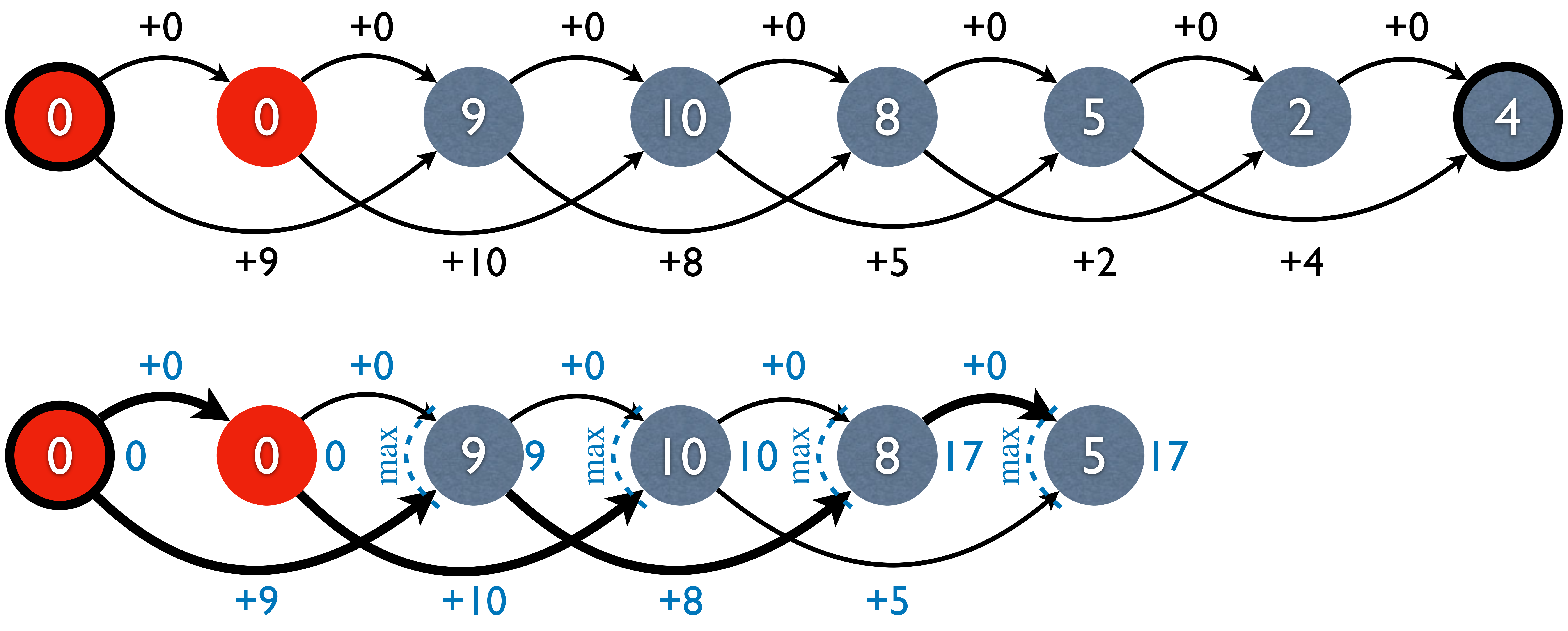
- MIS: longest path between source and target (see lecture video)
  - each node i has two incoming edges: $(i-2) \xrightarrow{a[i]} i$ (take) and $(i-1) \xrightarrow{0} i$ (not take)
  - $f(i)$: longest path between source and node i





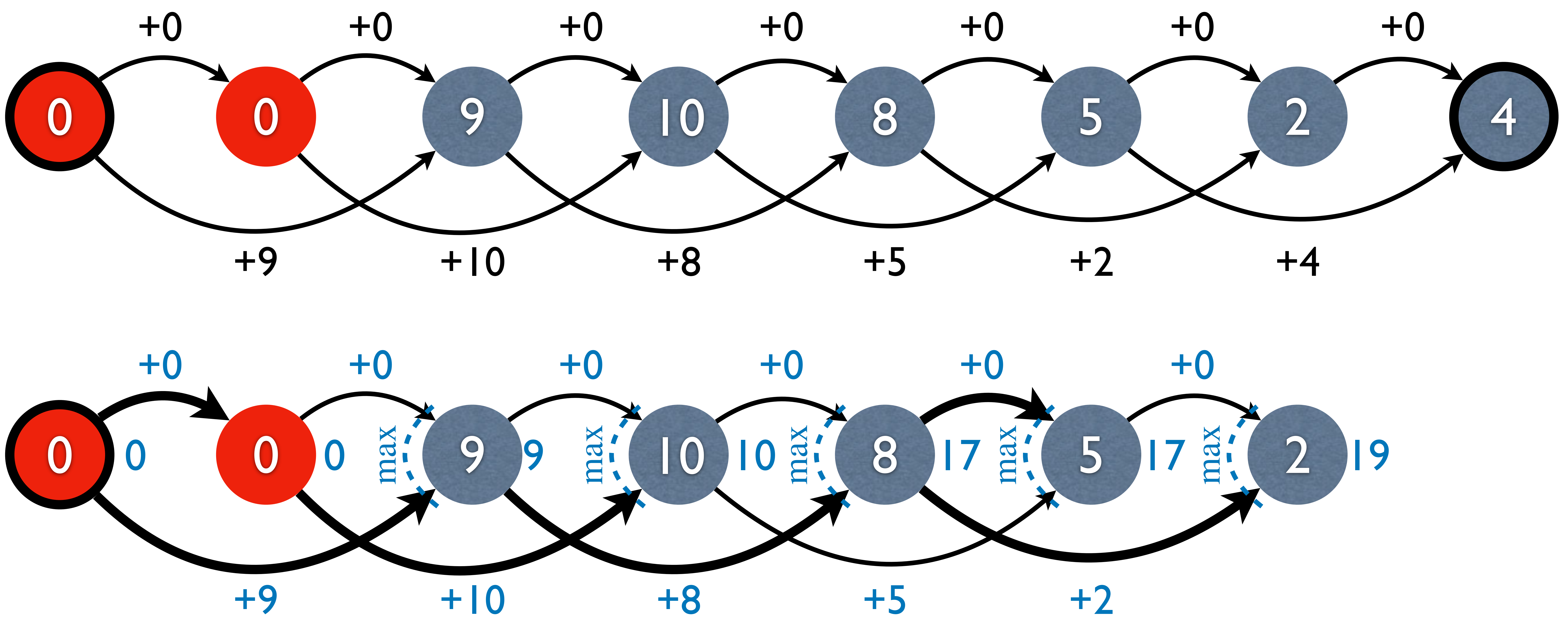
Graph Interpretation of DP

- MIS: longest path between source and target (see lecture video)
  - each node i has two incoming edges: $(i-2) \xrightarrow{a[i]} i$ (take) and $(i-1) \xrightarrow{0} i$ (not take)
  - $f(i)$: longest path between source and node i



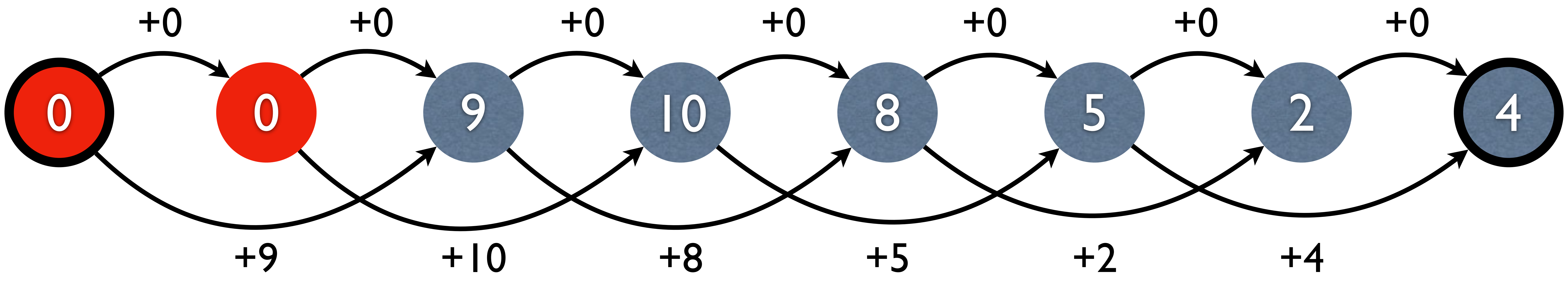
Graph Interpretation of DP

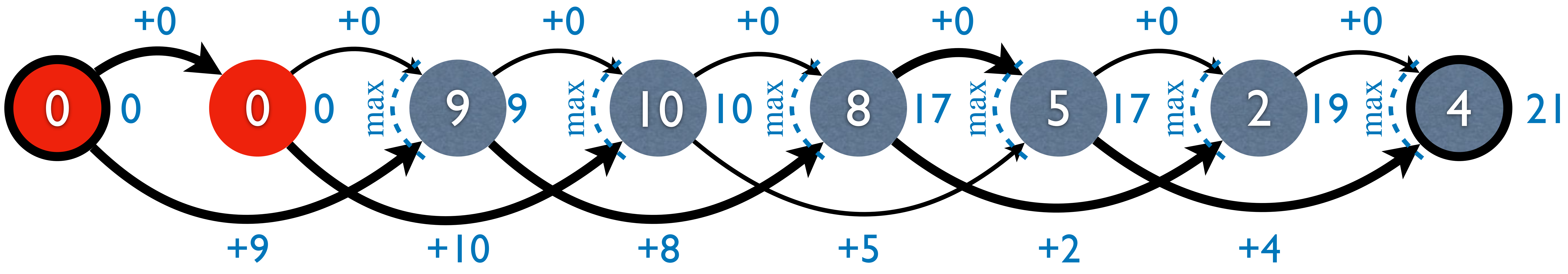
- MIS: longest path between source and target (see lecture video)
  - each node i has two incoming edges: $(i-2) \xrightarrow{a[i]} i$ (take) and $(i-1) \xrightarrow{0} i$ (not take)
  - $f(i)$: longest path between source and node i



Graph Interpretation of DP

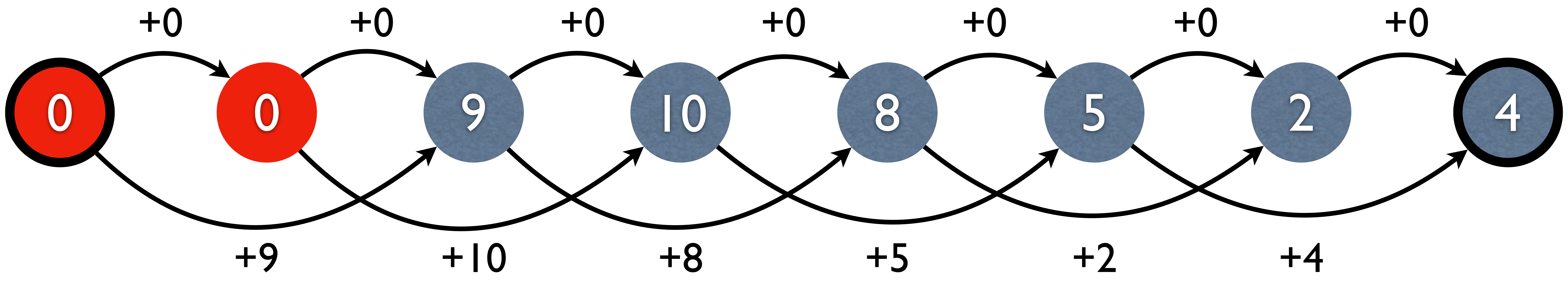
- MIS: longest path between source and target (see lecture video)
  - each node i has two incoming edges: $(i-2) \xrightarrow{a[i]} i$ (take) and $(i-1) \xrightarrow{0} i$ (not take)
  - $f(i)$: longest path between source and node i

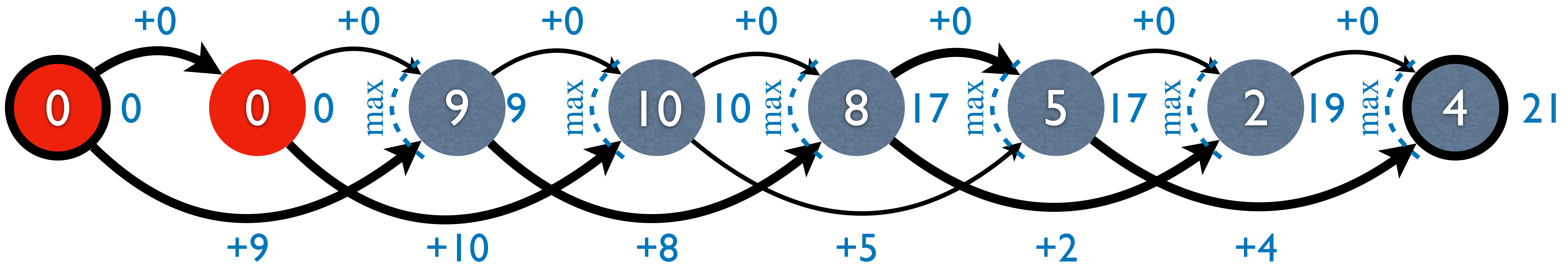




Graph Interpretation of DP

- MIS: longest path between source and target (see lecture video)
  - each node i has two incoming edges: $(i-2) \xrightarrow{a[i]} i$ (take) and $(i-1) \xrightarrow{0} i$ (not take)
  - $f(i)$: longest path between source and node i
- fibonacci & bitstrings: number of paths between source and target





Summary

- Divide-and-Conquer = single divide + disjoint conquer + single combine
- Dynamic Programming = multiple divides + memoized conquer + summarized combine
- overlapping subproblems due to multiple divides (still disjoint subproblems within each divide)
- two implementation styles
  - 1. recursive top-down + memoization
  - 2. bottom-up
- backtracking to recover best solution for optimization problems
  - 1. backpointers (recommended); 2. store subsolutions (not recommended — often slows down); 3. recompute on-the-fly
- two operators: $\oplus$ for summary (across multiple divides) and $\otimes$ for combine (within a divide)
- counting problems vs. optimization problems ("cost-reward model")
- three steps in solving a DP problem
  - define the subproblem
  - recursive formula
  - base cases

Summary

- Divide-and-Conquer = single divide + disjoint conquer + single combine
- Dynamic Programming = multiple divides + memoized conquer + summarized combine
- overlapping subproblems due to multiple divides (still disjoint subproblems within each divide)
- two implementation styles
  - 1. recursive top-down + memoization
  - 2. bottom-up
- backtracking to recover best solution for optimization problems
  - 1. backpointers (recommended); 2. store subsolutions (not recommended — often slows down); 3. recompute on-the-fly
- two operators: ⊕ for summary (across multiple divides) and ⊗ for combine (within a divide)
- counting problems vs. optimization problems ("cost-reward model")
- three steps in solving a DP problem
  - define the subproblem
  - recursive formula
  - base cases

cost reward

$$f(n) = \max \begin{cases} f(n-1) + 0 \\ f(n-2) + a[n] \end{cases}$$

summary operator ⊕ (across divides)

combination operator ⊗ (within a divide)