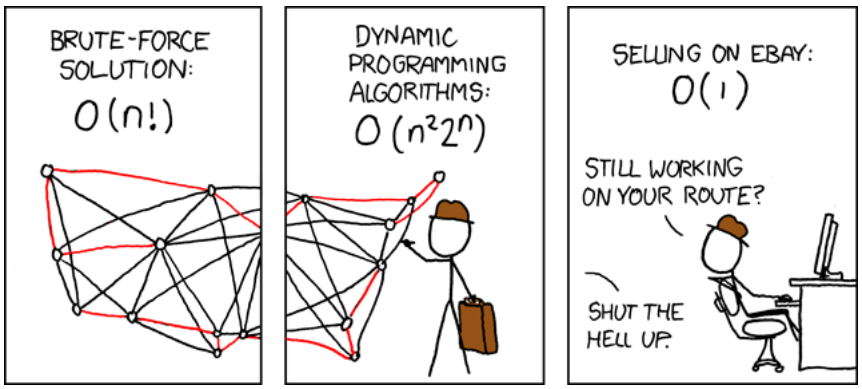
CS325: TSP Challenge

Prof. Glencora Borradaile

Due dates:
Presentations on Tuesday, November 18 in class
Project report due on Thursday, December 4 at 9:59 AM
Code and solutions to test cases uploaded to TEACH by Friday, December 5 at 9:59 AM



from: `http://xkcd.com/399/`

This project is rather open-ended, and I hope you will have fun trying out ideas to solve a very hard problem: the *travelling salesperson problem* or TSP.

> You are given a set of n cities and, for each pair of cities c_1 and c_2, the distance between them $d(c_1, c_2)$. Your goal is to find an ordering (called a *tour*) of the cities so that the distance you travel is minimized. The distance your travel is the sum of the distance from the first city in your ordering to the second plus the distance from the second city in your ordering to the third and so on until you reach the last city and finally add the distance from the last city to the first city. For example, say the cities are Chicago, New York, New Orleans and San Francisco. The total distance travelled visiting the cities in this order is:

d(Chicago, New York)+d(New York, New Orleans)+d(New Orleans, San Francisco)+d(San Francisco, Chicago)

In this project, you will only need to consider the special case where the cities are locations in a 2D plane (given by integral x and y coordinates) and the distances between two cities $c_1 = (x_1, y_1)$ and $c_2 = (x_2, y_2)$ is given by their Euclidean distance. So that we need not worry about floating point precision problems in computing the square-root (and so that everyone will get the same answer), we will always round this distance **to the nearest integer**. In other words, you will compute the distance between city c_1 and c_2 as:

$$d(c_1, c_2) = \texttt{nearestinteger}\left(\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}\right)$$

For example, if three cities are given by the coordinates $(0, 0), (1, 3), (3, 1)$, then a tour that visits these cities in this order travels a total distance of, or has *length*

$$\texttt{nint}(\sqrt{10}) + \texttt{nint}(\sqrt{8}) + \texttt{nint}(\sqrt{10}) = \texttt{nint}(3.16\ldots) + \texttt{nint}(2.82\ldots) + \texttt{nint}(3.16\ldots) = 9$$

Project specification

Your group is to design and implement a method for finding the best tour you can. TSP is not a problem for which you will be able to reliably find optimal solutions. It is *that* difficult. But your goal is to find the best solution you can. You may want to start with some *heuristics* as described here: `http://tinyurl.com/lqeqtgm`. Beyond that you may:

- read as much as you want about how to solve the travelling salesperson problem (so long as you cite any resources you use)
- use which ever programming language you want (so long as all group members are comfortable with it)

You may **not**:

- use existing implementations or subroutines
- extensive libraries (if you are unsure, email to ask)
- use other people's code (other than that of your group members)

Input specification A problem *instance* (a particular input) will always be given to you as a text file. Each line defines a city and each line has three numbers separated by a white space. The first number is the city identifier, the second number is the city's x-coordinate and the third number is the city's y-coordinate.

Output specification You must output your solution into another text file with $n + 1$ lines where n is the number of cities. The first line is to be the length of the tour you find. The next n lines should contain the city identifiers in the order they are visited by your tour. Each city must be listed exactly once in this list. This is the *certificate* for your solution and your solutions will be checked. If they are not valid, you will not receive credit for them.

Example instances We have provided you with three example instances. They are available in the directory `http://eecs.orst.edu/~glencora/cs325/tsp`. `example-input-*` are provided according to the input specifications. `example-output-*` are example outputs corresponding to these three example cases. You should use the output instances to test that you are computing distances correctly. The **optimal** tour lengths for test cases 1, 2 and 3 are 108159, 2579, and 1573084, respectively. You should use these values to judge how good your algorithm is.

Testing A testing procedure `tsp-verifier.py` is given in `http://eecs.orst.edu/~glencora/cs325/tsp` that we will use to verify your solutions. Usage to test example instance 1 is:
`python tsp-verifier.py example-input-1.txt example-output-1.txt`
You should test that your outputs are correct.

Project Components

Presentations on Tuesday, November 18 Each group will have up to 5 minutes to tell us about their algorithmic idea: how it works, why it is the best idea ever. Based on these presentations, any group may adopt or incorporate any other groups ideas. If your group adopts or incorporates any other group's idea in your final algorithm, you will receive a bonus of 1% *on your final grade*; you must describe **how and why** you adopted any other group's ideas in your report. You may adopt the ideas of more than one group, but you will only earn an additional 1% regardless of the number of adopted ideas. If another group adopts or incorportates their idea in their final report, they will receive a bonus of 1% on their final grade *for each group that adopts their idea*. That is, it pays to use other people's ideas and it pays (more) to convince others to use your idea.

Presenters may use the white board or document camera, but **may not** use the overhead projector connected to a computer.

Project report due on Thursday, December 4 at 9:59 AM You will submit a project report on December 4 at 9:59 AM. You may submit this up to 24hrs late without penalty. The project report may only be up to **2 pages in length** in no less than **11pt font**. In this report you must describe the ideas behind your algorithm as completely as is possible.

Test instances due on Friday, December 5 at 9:59 AM On **December 4 at noon**, we will make available 3 test instances. The location will be emailed to the class email list. By **Friday, December 5 at 9:59 AM**, you will be required to submit 3 separate text files according to the output specification and corresponding to each of these test instances. These files should be called `test1`, `test2`, `test3` and will be submitted to the TEACH subdirectory of **one** team member. You should also submit your code at this time.

The deadline for this portion of the project is hard. No exceptions will be made without prior approval from Prof. Borradaile.

Grading rubric Unlike previous projects, which had a project report and quiz, this project will not have a quiz. The 10% of your final grade determined from this TSP project will be broken down as follows:

30% for presentation You will be judged on clarity, creativity, effort, thoroughness, etc.

40% for 2-page report You will be judged on clarity and creativity.

30% for test instances You will be judged on how close your tour length is to that of the best possible solution. However, you will not be told what the optimal tour length is for the test instances.

Check list

1. **Does your program correctly compute tour lengths?**
2. **Does your program meet the output specification?**
3. **Did you check that you produce solutions that verify correctly?**
4. **Did you submit your solutions to the test instances by December 5 at 9:59 AM?**
5. **Did you submit your code to TEACH?**