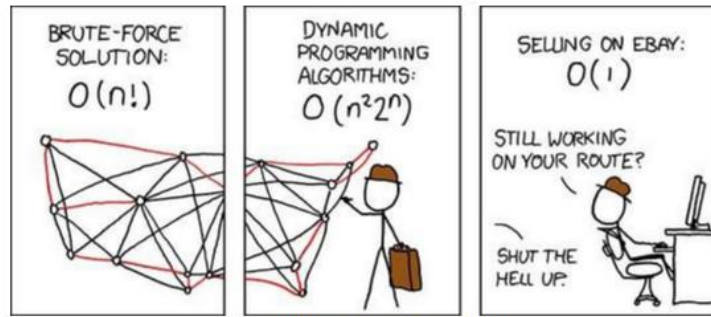


CS 325 Project 4: The Travelling Salesman Problem (TSP)



from: <http://xkcd.com/399>

This project you will have fun trying out ideas to solve a very hard problem: the Traveling Salesman Problem (TSP).

You are given a set of n cities and for each pair of cities c_1 and c_2 , the distances 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 you travel is the sum of the distances from the first city in the ordering to the second city, plus the distance second city to the third city, and so on until you reach the last city, and then adding the distance from the last city to the first city. For example if the cities are Seattle, Portland, Corvallis and Boise. The total distance traveled visiting the cities in this order is:

$$d(\text{tour}) = d(\text{Seattle}, \text{Portland}) + d(\text{Portland}, \text{Corvallis}) + d(\text{Corvallis}, \text{Boise}) + d(\text{Boise}, \text{Seattle})$$

In this project, you will only need to consider the special case where the cities are locations in a 2D grid (given by their x and y coordinates) and the distance between two cities $c_1 = (x_1, y_1)$ and $c_2 = (x_2, y_2)$ is given by their Euclidean distance. To avoid floating point precision problems in computing the square-root, we will always round the distance to the nearest integer. In other words you will compute the distance between cities c_1 and c_2 as:

$$d(c_1, c_2) = \text{nearestint}(\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2})$$

For example, if the three cities are given by the coordinates $c_1 = (0, 0)$, $c_2 = (1, 3)$, $c_3 = (6, 0)$, then a tour that visits the cities in order $c_1 \rightarrow c_2 \rightarrow c_3 \rightarrow c_1$ has the distance

$$d(\text{tour}) = d(c_1, c_2) + d(c_2, c_3) + d(c_3, c_1)$$

where

$$\begin{aligned} d(c_1, c_2) &= \text{nearestint}(\sqrt{(0 - 1)^2 + (0 - 3)^2}) \\ &= \text{nearestint}(\sqrt{(-1)^2 + (-3)^2}) \\ &= \text{nearestint}(\sqrt{1 + 9}) \\ &= \text{nearestint}(\sqrt{10}) \\ &= \text{nearestint}(3.1622 \dots) \\ &= 3 \end{aligned}$$

CS 325 Project 4: The Travelling Salesman Problem (TSP)

$$\begin{aligned}d(c_2, c_3) &= \text{nearestint}\left(\sqrt{(1-6)^2 + (3-0)^2}\right) \\&= \text{nearestint}\left(\sqrt{(-5)^2 + (3)^2}\right) \\&= \text{nearestint}(\sqrt{25+9}) \\&= \text{nearestint}(\sqrt{34}) \\&= \text{nearestint}(5.8309 \dots) \\&= 6\end{aligned}$$

$$\begin{aligned}d(c_3, c_1) &= \text{nearestint}\left(\sqrt{(6-0)^2 + (0-0)^2}\right) \\&= \text{nearestint}\left(\sqrt{(6)^2 + (0)^2}\right) \\&= \text{nearestint}(\sqrt{36+0}) \\&= \text{nearestint}(\sqrt{36}) \\&= \text{nearestint}(6) \\&= 6\end{aligned}$$

So that $d(\text{tour}) = 3 + 6 + 6 = 15$.

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 easily find optimal solutions. It is difficult. Your goal is to find the best solution you can in a certain time frame. You may want to start with some local search heuristics as described in DPV Section 9.3.1.

- You may read anything about how to solve TSP please cite any sources
- Use which ever programming language you want that runs on flip2.engr.oregonstate.edu as long as all group members are comfortable with it.

Your program must:

- Accept problem instances on the command line
- Name the the output file as the input file's name with .tour appended (for example input **tsp_example_1.txt** will output **tsp_example_1.txt.tour**)
- Compile/Execute correctly and without debugging on flip2.engr.oreognstate.edu according to specifications and any documentation you provide.

Input specifications:

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

CS 325 Project 4: The Travelling Salesman Problem (TSP)

Output specifications:

- You must output your solution into another text file with $n+1$ lines, where n is the number of cities.
- The first line is the length of the tour your program computes.
- 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 on Canvas and are provided according to the input specifications.

- **tsp_example_*.txt** Input files
- **tsp_example_*.txt.tour** Example outputs corresponding to these three input cases. The optimal tour lengths for test cases 1, 2, and 3 are 108159, 2579 and 1573084, respectively. Clearly these do not match the values in the tour files. You should use these values to judge how good your algorithm is.

Testing

A testing procedure **tsp-verifier.py** is given that we will use to verify your solutions. Usage to test example an instance is: (NOTE: requires TSPAllVisited.py)

```
python tsp-verifier.py inputfilename solutionfilename
```

You should test that your outputs are correct. By “correct” we mean that the distances have been computed correctly not that the solution is optimal.

Competition

We will hold a completion. The competition will require your program to find the best solution possible to one or more test instances within a fixed amount of time (e.g. 5 minutes). The competition instances will be available on Wednesday 3/11/15 at 8:00 am PST. You will not be told the optimal tour length for these instances.

Project Report

You will submit a project report containing the following:

- the ideas behind your algorithm as completely as possible
- your “best” tours for the three example instances and the time it took to obtain these tours.
- Your best tours for the competition test instance(s).

Submission

- All files, solutions and report must be zipped and submitted to both TEACH and Canvas by Friday 3/13/15 at 11:59pm PST,

CS 325 Project 4: The Travelling Salesman Problem (TSP)

Check List

- Does your program correctly compute tour lengths for simple cases?
- Does your program read input files and options from the command line?
- Does your program meet the output specifications?
- Did you check that you produce solutions that verify correctly?
- Did you find solutions to the example instances?
- Did you find solutions to the competition test instances?
- Does your code compile/run without issue according to your documentation?
- Have you submitted your report, your solutions to the test cases, your source and any comments you wish to include to TEACH and Canvas?