Programming Assignment 2 - Freesbee Golf

JWJ CS UK (do not distribute)
CS315-002-Fall 2020
Final submission November 16, 2020

1 Project pa2 cs315-002: Requirements

Frisbee Golf - -a training session
Your task is to design an efficient solution for the *Frisbee Golf* problem: to implement your solution, to test it, and to describe and discuss the results. The main point of this assignment is in using known, nontrivial algorithms to develop an application. In our case, we will apply one of the algorithms (or perhaps techniques from) that we learned in our class.
This assignment contributes to the Student Learning Outcomes: *"Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the programs discipline"* and *"Apply computer science theory and software development fundamentals to produce computing-based solutions"*.

TASKS:

Your overall task is to develop an efficient algorithm for the *Freesbee Golf* problem (see below). The development is divided into three milestones (only the final program is to be submitted.)

Initial Step Understanding the problem: Check outputs for small inputs (calculations by hand). It is a paper-and-pencil exercise (no submissions).

Intermediate Step Design of the solution; pseudocode; initial implementation; testing; (no submission)

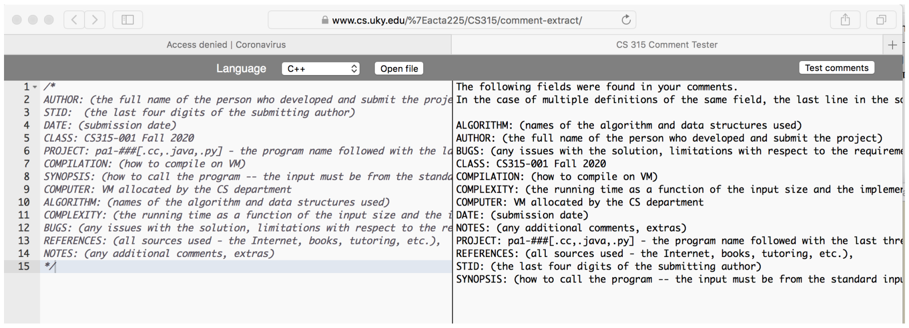
Final Stage – submit on November 16, 2020 Final Submission: the entire tested project.

Submit sources/program named pa1###[.cc,.java,.py] - the extension depends on the implementation language. ### should be replaced with the last three digits of your SID#. The source code must include comments in the header as described and applied for hw4-s.

The required fields in the comments are:

```
/*
AUTHOR: (the full name of the person who developed and submit the project)
STID:  (the last four digits of the submitting author)
DATE: (submission date)
CLASS: CS315-002 Fall 2020
PROJECT: pa2-###[.cc,.java,.py] - the program name followed with the last three digits of your SID#
COMPILATION: (how to compile on VM)
SYNOPSIS: (how to call the program -- the input must be from the standard input, with the redirection '<')
COMPUTER: VM allocated by the CS department
```

```
ALGORITHM: (names of the algorithm and data structures used)
COMPLEXITY: (the running time as a function of the input size and the implemented algorithm and structures)
BUGS: (any issues with the solution, limitations with respect to the requirements, etc.)
REFERENCES: (all sources used - the Internet, books, tutoring, etc.),
NOTES: (any additional comments, extras)
*/
```


www.cs.uky.edu/%7Eacta225/CS315/comment-extract/
Access denied | Coronavirus
CS 315 Comment Tester
Language
C++
Open file
Test comments

```
/*
AUTHOR: (the full name of the person who developed and submit the proje
STID:  (the last four digits of the submitting author)
DATE: (submission date)
CLASS: CS315-001 Fall 2020
PROJECT: pa1-###[.cc,.java,.py] - the program name followed with the la
COMPILATION: (how to compile on VM)
SYNOPSIS: (how to call the program -- the input must be from the standa
COMPUTER: VM allocated by the CS department
ALGORITHM: (names of the algorithm and data structures used)
COMPLEXITY: (the running time as a function of the input size and the i
BUGS: (any issues with the solution, limitations with respect to the re
REFERENCES: (all sources used - the Internet, books, tutoring, etc.),
NOTES: (any additional comments, extras)
*/
```

```
The following fields were found in your comments.
In the case of multiple definitions of the same field, the last line in the sc

ALGORITHM: (names of the algorithm and data structures used)
AUTHOR: (the full name of the person who developed and submit the project)
BUGS: (any issues with the solution, limitations with respect to the requireme
CLASS: CS315-001 Fall 2020
COMPILATION: (how to compile on VM)
COMPLEXITY: (the running time as a function of the input size and the implemer
COMPUTER: VM allocated by the CS department
DATE: (submission date)
NOTES: (any additional comments, extras)
PROJECT: pa1-###[.cc,.java,.py] - the program name followed with the last thre
REFERENCES: (all sources used - the Internet, books, tutoring, etc.),
STID: (the last four digits of the submitting author)
SYNOPSIS: (how to call the program -- the input must be from the standard inpu
```

You can check that your comments are formatted correctly using an on-line tool `https://www.cs.uky.edu/~acta225/CS315/comment-extract/` developed for our class.

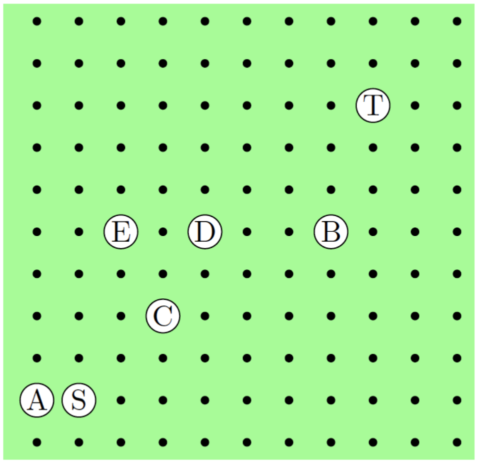
The submissions will be tested on VMs, on multiple input data, including large graphs. This project requires individual work: any sources, assistance, etc. must be credited in the comments as the comment in the "REFERENCES" field.

Grading In addition to inspecting the code, the submissions will be tested automatically: the output must be well-formatted, exactly as described above, and correct. The final score will depend on the completeness and correctness of the submission and results on the test data. The efficiency test will have a time limits for completing the calculations and outputting the results.

Problem statement:

Consider a Frisbee Golf course with a number of frisbee baskets. In teaching techniques of throwing a frisbee for a beginner, you are trying to identify a path of frisbee baskets from the starting to the target basket, that minimizes the maximal distance between two consecutive frisbee baskets on this path. More formally, for each path from S to T, we define the throw distance as the maximum distance between consecutive holes (baskets) on this path. The optimal throw distance is the minimum throw distance over all paths from S to T.

For example (see also additional sample inputs below), in the course depicted on this figure the optimal throw distance is 3.162. There is a path from S to T with the distances between consecutive holes on this path no larger than 3.162.



Coordinates of baskets	**The corresponding matrix of pairwise distances (not a part of the input)**
7	
1 1	S 0.000 9.899 1.000 7.211 2.828 5.000 4.123
8 8	T 9.899 0.000 10.630 3.162 7.071 5.000 6.708
0 1	A 1.000 10.630 0.000 8.062 3.606 5.657 4.472
7 5	B 7.211 3.162 8.062 0.000 4.472 3.000 5.000
3 3	C 2.828 7.071 3.606 4.472 0.000 2.236 2.236
4 5	D 5.000 5.000 5.657 3.000 2.236 0.000 2.000
2 5	E 4.123 6.708 4.472 5.000 2.236 2.000 0.000

Input/Output:

The structure of the input is:

- The first line contains one number n – the number of frisbee baskets: n satisfies $2 \leq n \leq 100$.
- The n following lines contains coordinates of the frisbee baskets: x and y coordinates per line, $0 \leq x, y \leq 1000$, separated with a space. The first pair of coordinates corresponds to the starting frisbee basket, the second pair of the coordinates corresponds to the target basket. The following lines describe coordinates of the remaining frisbee baskets.

Output consists of exactly one line with one number: the optimal throw distance in a training path from the starting (first pair of coordinates) to the target (the second pair of coordinates) basket.

See the following examples for valid input and output.

Input	**Output**

6
1 1
8 8
0 2
7 5
3 3
4 5
	3.162

Input	**Output**

9
1 1
8 8
4 2
4 7
5 4
4 5
7 7
2 0
9 3
	3.000