

Project 1

Spring 2024 CPSC 335 - Algorithm Engineering

Due: 02/25

Points: 100

Algorithm 1: Connecting Pairs of Persons

Assume that there are n individuals (possibly couples) sitting in $2n$ seats arranged in a row. They are able to hold hands. The people and the seats are presented by an integer row, where $row[i]$ is the ID of the person sitting in the i^{th} seat. The couples are arranged in order, the first couple is presented as $(0, 1)$, the second couple being $(2, 3)$, and so on, until the last couple. The last couple being $(2n - 2, 2n - 1)$.

Design an algorithm that takes in any available number of couple and return the minimum number of swaps, such that every couple is sitting side by side. A swap consists of choosing any two people, and ensuring that they switch seats

Below are some sample inputs and outcomes.

Sample 1:

Input: row = [0, 2, 1, 3]

Output: 1

Only the second (row[1]) needs to be swapped with the third (row[2]) individual

Sample 2:

Input: row = [3,2,0,1]

Output: 0

All couples are already seated side by side

Assume that n is less or equal to 30 and even. $2n$ represents the length of each row. All elements of the row are unique.

Algorithm 2: Greedy Approach to Hamiltonian Problem

You are given a set of cities that are laid out in a circle, connected by a circular road that runs clockwise. Each city has a gas station that provides gallons of fuel, and the distances between these

cities are known. You have a car that can drive some number of miles per gallon of fuel, and your goal is to pick a **starting city** such that you can fill up your car in that city (using that city's gas station). You can then drive to the next city, refill up your car with that city's fuel, drive to the next city, and so on and so forth until you return back to the starting city with 0 or more gallons of fuel left. This city is called a preferred starting city. In this problem, it is guaranteed that there will always be exactly one valid starting City

The problem set involves series of arrays. The first array is the distance between neighboring cities. Assume that city i is $distances[i]$ away from city $i + 1$. Since the cities are connected via a circular road, the last city is connected to the first city. In other words, the last distance in the distances array is equal to the distance from the last city to the first city. The second array is an array of gas available at each city, where $gas[i]$ is equal to the gas available at city i . The total amount of gas available (from all gas stations) is enough to travel to all cities. Your gas tank always starts out empty, and a positive integer value for the number of miles that your car can travel per gallon of fuel (miles per gallon, or MPG) is also given.

Write a function that returns the index of the preferred starting city.

Sample Input:

```
city_distances = [5, 25, 15, 10, 15]
fuel =          [1, 2, 1, 0, 3]
mpg =          10
```

The preferred starting city for the sample above is city 4

To Do:

1. Develop a complete and clear pseudocode for the two algorithms
2. Mathematically analyze your pseudocode and prove the efficiency class, using either induction, limits or step counts. State the method used.
3. Produce a readme file (can be in txt), describing how to execute your program.
4. Produce a brief written project report **in PDF format**. Your report should include:
 - a. Your name(s), CSUF-supplied email address(es), and an indication that the submission is for project 1.
5. Implement your algorithms in Python or C++. Your codes should be well commented, including your name(s).
6. Submit all files, including your PDF report and codes, separately.

Grading Rubric

The suggested grading rubric is given below:

Algorithm 1 (55 points)

- a. Clear and complete Pseudocode = 10 points
- b. Mathematical analysis and correct Big O efficiency class = 5 points
- c. Inclusion of a Readme file = 5 points
- d. Well commented codes = 5 points
- e. Successful compilation of codes = 25 points
- f. Produces accurate results = 5 points

Algorithm (45 points each)

- a. Clear and complete Pseudocode = 10 points
- b. Mathematical analysis and correct Big O efficiency class = 5 points
- c. Inclusion of a Readme file = 5 points
- d. Well commented codes = 5 points
- e. Successful compilation of codes = 15 points
- f. Produces accurate results = 10 points

Submitting your code

Submit your files to the Project 1 Assignment on Canvas. It allows for multiple submissions. You may submit your files **separately**. Do not zip or use .rar. You may work in a group or individually. Ensure that every page of your submissions and codes contain name(s) of everyone that participated. If you decide to work in a group, please use the Canvas-created groups.

Ensure your submissions are your own works. Be advised that your submissions may be checked for plagiarism using automated tools. Do not plagiarize. As stated in the syllabus, a submission that involves academic dishonesty will receive a 0% score on that assignment. A repeat offense will result in an "F" in the class and the incident will be reported to the Office of Student Conduct.

Deadline

The project deadline is **Sunday, February 25, by 11:59 pm** on Canvas.

Penalty for late submission (within 48 hours) is as stated in the syllabus. Projects submitted more than 48 hours after the deadline will not be accepted.