Algorithms Homework #3

Due: 107/12/12 (Wed.) 09:10 (Hand-Written)
107/12/13 (Thu.) 03:00 (Programming)

Contact TAs: alg2018ta@gmail.com

Please read the instructions & submission rules very carefully!!
Any loss of credits due to not following the instructions is NOT negotiable.

Instructions

- There are two sections in this homework, including the hand-written part and the programming part.
- **Academic Honesty**

  Cheating is not allowed and is a violation of our school regulations. Plagiarism is a form of cheating. If cheating is discovered, all students involved will receive an F grade for the course (**NOT** negotiable).
- **Collaboration Policy**

  Self-learning by researching on the Internet or discussing with fellow classmates is highly encouraged. However, you must obtain and write the final solution by yourself. Please specify, if any, the references for each of your answers (e.g. the name and student ID of your collaborators and/or the Internet URL you consult with). If you solve the problems by yourself, you must also specify "*no collaborators*".

  You should specify the references and/or collaborators of all problems (including the programming part) on your hand-written answer sheets or in the report of your programming problems. **Homework without reference specification will not be graded.**
- **Delay Policy**

  For hand-written problems, you must submit your answer sheets to the instructor at the beginning of the class. TAs will collect them during the first break. **No late submission is allowed for hand-written problems**.

  For programming problems, you have a *six-hour* delay quota for the whole semester. Once you have exceeded your quota, **any late submission will receive no credits**.

Hand-Written Problems

Problem 1 (10%)

Consider the maze in Figure 1. We want to find a shortest path from A to B which requires the smallest number of steps.

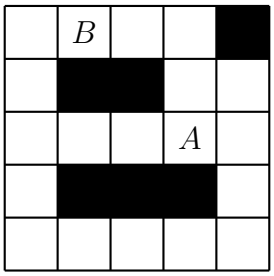

Figure 1: Maze for Problem 1

1. (4%) Model the maze into a graph problem, and draw the resulting graph.
2. (4%) Give an efficient algorithm that solves the problem.
3. (2%) Analyze the time and space complexity of your answer.

Problem 2 (15%)

Judy and Nick are taking a road trip from Zootopia to Bunnyburrow. Since it is a 46-hour drive, Judy and Nick decide to take turns driving at each rest area they visit. Since Nick has a better sense of direction than Judy, he should be driving when they depart from Zootopia and when they arrive at Bunnyburrow, so that Judy can avoid downtown traffic.

Judy and Nick want to find a *shortest* route from Zootopia to Bunnyburrow (if exists) such that Judy and Nick could drive alternately while Nick takes the first and the last turns (note that this means they have to stop at at least two rest areas).

Suppose Judy and Nick has a highway map from Zootopia to Bunnyburrow, where there are m highway segments and n rest areas (a highway segment connects two distinct rest areas). The lengths of each highway segment $l_1, \ldots, l_m$ are also given.

1. (5%) Model the scenario into a graph problem, and explain the elements of your graph.
2. (8%) Give an efficient algorithm that solves the problem. Your answer should return either the shortest path from Zootopia to Bunnyburrow that fits their needs, or "None" if no such path exists.
3. (2%) Analyze the time and space complexity of your answer.

Problem 3 (15%)

After the road trip, Nick and Judy stay at a local hotel in Bunnyburrow and plan to visit Judy's parents the next day. Since it is the first time that Nick meets Judy's parents, he would like to stop by the pâtisserie to buy a carrot cake on his way to her house, *if it is possible* to do so without increasing the length of his path by more than a factor of α.

Nick wants to determine a *shortest* path from the hotel to Judy's house, given Nick's preference for stopping at the pâtisserie along the way.

Suppose Nick has a street map of Bunnyburrow, where there are m road segments and n intersections (a road segment connects two distinct intersections), and the lengths of all road segment are equal. Also, the hotel and the pâtisserie and Judy's house are located at intersections s, u and t, respectively.

1. (5%) Model the scenario into a graph problem, and explain the elements of your graph.
2. (8%) Give an efficient algorithm that solves the problem. Your answer should return either the shortest path from s to t or the shortest path from s to t containing u, depending on the situation.
3. (2%) Analyze the time and space complexity of your answer.

Problem 4 (25%)

Mark is a student of Harvard College. He is currently building a social network website, and wants to implement a feature that helps users find potential friends.

First, denote the users of the website as $V = \{v_0, v_1, \ldots, v_n\}$. Define the ***interest*** between any two distinct users v_i, v_j as $I(v_i, v_j)$, where $0 < I(v_i, v_j) < 1$ for all $0 \le i, j \le n$. This value can be computed based on how frequently v_i visits v_j's profile page or views v_j's posts, etc. Note that $I(v_i, v_j)$ is not necessarily equal to $I(v_j, v_i)$. Suppose that all interest values are precalculated and can be obtained in $O(1)$ time.

Next, we say that a user u is ***connected*** to another user v if they are direct friends or are connected via some mutual friends. Denote the ***set of connections*** between u and v as $P(u, v)$. For example, if $u = u_0$ has a friend u_1, who has a friend u_2, ...who has a friend $u_m = v$, then $p = \langle u_0, u_1, u_2, \ldots, u_m \rangle \in P(u, v)$ is a valid ***connection*** between u and v. Note that given two users u and v, there may be multiple connections between them.

For each connection $p \in P(u, v)$, we denote $L(p) = |p| - 2$ as the ***length*** of this connection, which is the number of intermediate friends between u and v. Also, define the ***strength*** of this connection as

$$S(p) = I(u_0, u_1) \times \cdots \times I(u_{m-1}, u_m) = \prod_{i=0}^{L(p)} I(u_i, u_{i+1}),$$

where $L(p) = m - 1$. Note that when u and v are direct friends, there exists a connection $p_d = \langle u, v \rangle \in P(u, v)$ between them such that $L(p_d) = 0$ and $S(p_d) = I(u, v)$.

In this problem, we want to compute the ***strength*** of the **strongest** connection between a given user u and **every other** user v with a ***length*** of **at most** k. That is, for every pair of u and $v \in V \backslash \{u\}$, find

$$S(p^*) = S(\arg\max_{p \in \widetilde{P}(u,v)} \{S(p)\}),$$

where $\widetilde{P} = P \backslash \{p | p \in P, L(p) > k\}$, that is, we ignore all connections with more than k intermediate friends in between.

1. (10%) Model the scenario into a graph problem, and explain the elements of your graph.
2. (10%) Give an efficient algorithm that solves the problem.
3. (5%) Suppose there are a total of $|V|$ users and $|E|$ friend pairs, analyze the time and space complexity of your algorithm.

Programming Problems

Problem 1 (35%)

In the spring of 2016, Nick Wilde entered the Zootopia Police Academy as a fresh cadet. Before Nick can join the ZPD as a police officer, he has to take a minimum number of courses and graduate from the academy.

The Zootopia Police Academy provides many courses, but some of them have prerequisites. For example, Nick must pass the course "Defensive Driving" before he can take the course "Vehicle Pursuit". The academy adopts the semester system where there are two school terms in each academic year, and some of the courses may not be available in every term.

Given the minimum number of courses to take and the list of all courses with their prerequisites (if exists) and availability, Nick wants to determine an optimal ordering of courses he should take so that he can graduate from the academy as soon as possible.

Input

The input of this problem is a text file of numbers.

- The first row contains two integers n and p, where n is the minimum number of courses a cadet should take to graduate, and p is the number of prerequisite pairs.
- The following n rows contain two integers i and j, where i is the course ID ($0 \leq i \leq n-1$) and $j \in \{\texttt{0}, \texttt{1}, \texttt{2}\}$ indicates in which terms the course is available (`0` for the *Spring* term, `1` for the *Fall* term, and `2` for both terms).
- The following p rows contain two distinct integers x and y ($0 \leq x, y \leq n-1$), meaning one has to pass the course y before taking the course x.

Output

The output should also be a text file of numbers. If there is no legal solution, output a single `-1` for the entire text file. Otherwise, follow the rules below:

- The first row should contain a single integer t indicating the minimum number of terms it takes for Nick to graduate.
- The following t rows should contain the IDs of the courses Nick should take in each term. You should separate each course ID with a space. If Nick doesn't have to take any course in a term, output a `-1` in the corresponding row.

Refer to Table 1 for an example list of courses, where $n = 4$.

Course ID	**Course Title**	**Prerequisites**	**Availability**	
			Spring	*Fall*
0	Psychology	—		✓
1	Criminology	Psychology	✓	
2	Constitutional Law	Psychology	✓	
3	Criminal Law	Criminology, Constitutional Law	✓	✓

Table 1: Example list of courses in the Zootopia Police Academy

A resulting optimal ordering of courses Nick should take is provided in Table 2, where the corresponding `input.txt` and `output.txt` are shown in Table 3.

	Courses
Spring, 2016	—
Fall, 2016	Psychology
Spring, 2017	Criminology, Constitutional Law
Fall, 2017	Criminal Law

Table 2: An optimal ordering of courses Nick should take for the example in Table 1

input.txt	**output.txt**
4 4 0 1 1 0 2 0 3 2 1 0 2 0 3 1 3 2	4 -1 0 1 2 3

Table 3: Example output for Problem 1

For this problem, your `.py` file should take two arguments of type `string`, which are the paths to the input and output text files, respectively. We will test your code in the following manner, and verify the correctness of `output.txt`:

```
$ python p1_b05901000.py input.txt output.txt
```

1. (30%) Give an efficient algorithm for solving this problem. Write your code in a file named "`p1_<StudentID>.py`".
2. (5%) Briefly explain your work in "`report_<StudentID>.pdf`".

Problem 2 (20%)

In this problem, we will implement the maze problem from the Hand-Written Problem 1. Given a rectangular maze with the starting and ending points A and B, we want to find a shortest path from A to B which requires the smallest number of steps.

Refer to Figure 2 for an example maze, where an optimal solution is shown with the blue arrow. Note that for a given maze, the optimal solution may not be unique.

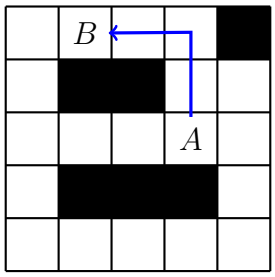

Figure 2: Maze for Problem 2

Input

The input of this problem is a text file containing m rows of numbers, where each row contains n numbers separated by spaces. All numbers are either `0`, `1`, `2` or `3`, which indicates the walls, paths, starting point, and ending point of the maze, respectively. Note that there is exactly one starting/ending point in the maze.

Output

The output should be a text file containing two rows of numbers. If there is no legal solution, output a single `-1` for the entire text file. Otherwise, follow the rules below:

- The first row should describe your optimal path for the input maze, where the numbers indicate the direction of each step. You should output `1`, `2`, `3`, `4` for directions north, east, south and west, respectively.
- The second row should contain only one number, which is the minimized number of steps.

For the example in Figure 2 where $m = n = 5$, its corresponding input/output is provided in Table 4.

input.txt	**output.txt**

`1 3 1 1 0`
`1 0 0 1 1`
`1 1 1 2 1`
`1 0 0 0 1`
`1 1 1 1 1`
	`1 1 4 4`
	`4`

Table 4: Example output for Problem 2

For this problem, your `.py` file should take two arguments of type `string`, which are the paths to the input and output text files, respectively. We will test your code in the following manner, and verify the correctness of `output.txt`:

```
$ python p2_b05901000.py input.txt output.txt
```

(20%) Give an efficient algorithm for solving this problem. Write your code in a file named "`p2_<StudentID>.py`".

Submission Rules

- Please write your code using **Python 3.x** only. You will lose scores if TAs fail to run your codes due to compatibility issues.
- You may import the following modules in this homework. You are **NOT** allowed to import any other modules unless otherwise listed below:
  - `import sys`
- Please pack your source codes into a single `.tar` or `.zip` file named "`hw3_<StudentID>`" (e.g. `hw3_b05901000.zip`) and upload it to CEIBA before the deadline. You should include in it only the following files/folders:

```
hw3_b05901000.zip
├── p1/
│   └── p1_b05901000.py
├── p2/
│   └── p2_b05901000.py
└── report_b05901000.pdf
```

- We include a file-checking script (`selfCheck.py`) for you in the attachment. Before uploading your `.zip`/`.tar` file, make sure that it can be verified automatically by the script. Usage is described as follows:

```
$ python selfCheck.py hw3_b05901000.zip
Passed! Your student ID is b05901000.
```