# Algorithms Homework #3
# Solutions

Due: 2017/12/13 (Wed.) 03:00 (Programming)
2017/12/13 (Wed.) 14:20 (Hand-written)

Contact TAs: alg2017ta@gmail.com

## Instructions

- There are two sections in this homework, including the hand-written part and the programming part.
- **Hand-written.** Please submit your answers to the instructor at the beginning of the class. TA will collect the answer sheets in the first break of the class. **NO LATE SUBMISSION IS ALLOWED**.
- **Programming.** Please upload your source codes in a single .tar or .zip file (e.g., b04901001_hw3.zip) to Ceiba. You must implement your program with **Python 3**.
- **Delay policy.** You have a six-hour delay quota for programming assignments for the whole semester. Once the quota is used up, any late submission will receive no credits.
- **Collaboration policy.** Discussion with other students is strongly encouraged, but you must obtain and write the final solution by yourself. Please specify the references (e.g., the name and student id of your collaborators and/or the Internet URL you consult with) for each **hand-written and programming question** on your answer sheet of the hand-written problems. If you solve some problems by yourself, please also specify "no collaborator". Homeworks without reference specification may not be graded.
- **Academic honesty.** Cheating is not allowed and is against university policy. Plagiarism is a form of cheating. If cheating is discovered, all students involved will receive an F grade for the course (NOT negotiable).

# Hand-Written Problems

Note that in this section, you can apply all graph algorithms introduced in class as a black box without further explanation unless specified.

## Problem 1 (15%)

Please answer the following questions.

1. (2%) A tic-tac-toe game can be modeled as a directed graph by converting each board position (i.e., a game state) to a vertex and each legal move to an edge. What is the in-degree and out-degree of the board position shown in Figure 1. (It is X's turn.)

   *Hint*: Any "legal" move is an edge.

   **Solution:**
   There are three possible cases leading to the board position and X can be put at four different grids. Thus, in-degree = 3 and out-degree = 4.

2. (3%) Apply Dijkstra's algorithm to the graph shown in Figure 2 starting from vertex $S$. What is the order of vertices that get removed?

   **Solution:**
   Perform the Dijkstra's algorithm and the sequence is (S,A,C,E,B,D,F).

3. (5%) What is the running time of depth-first search, in terms of |V| and |E|, implemented with (1) adjacency list, and (2) adjacency matrix?

   **Solution:**

   (1) (2%) The running time of depth-first search implemented with adjacency list is $O(V + E)$ as taught in class.

   (2) (3%) Depth-first search visits each vertex once and as it visits each vertex, and we need to find all of its neighbors to figure out where to search next. Finding all its neighbors in an adjacency matrix requires $O(V)$ time, so overall the running time will be $O(V^2)$.

4. (5%) An **Euler tour** of a strongly connected, directed graph $G(V, E)$ is a cycle that traverses each edge of $G$ exactly once, although it may visit a vertex more than once. Show that $G$ has an Euler tour **if and only if** in-degree($v$) = out-degree($v$) for each vertex $v \in V$.

   **Solution:**

   (1) (2%) $G$ has an Euler tour $\Rightarrow$ in-degree($v$) = out-degree($v$) for each vertex $v$:
   Given a graph $G$ with an Euler tour, we can find a tour that goes through all edges in $G$ without repetition with the same source and target. It is clear that, for each vertex $v'$ in the cycle, the numbers of edges that go in and go out $v'$ must be equal.

(2) (3%) In-degree($v$) = out-degree($v$) for each vertex $v \Rightarrow G$ has an Euler tour:
Since the in-degree and out-degree of each vertex $v \in V$ are equal, we can find a cycle from an arbitrary vertex. Then, the edges in the cycle are removed. We also can find a cycle from a remaining vertex that still has edges connecting to it. The process repeats until no edge is available. By traversing edges in the depth first order once a cycle is met, we can find an Euler tour.

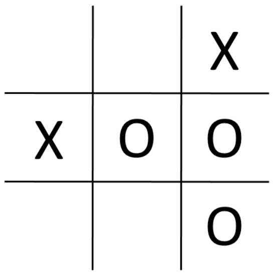

Figure 1: A tic-tac-toe board position

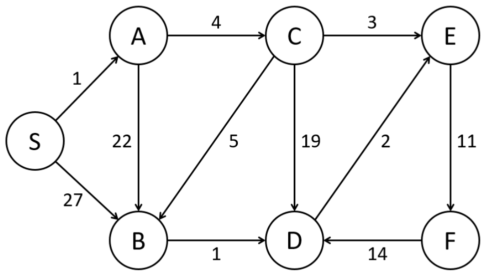

Figure 2: A directed graph

## Problem 2 (10%)

A **binary-weighted graph** is a graph where each edge has a weight of either 0 or 1. Given a source vertex $S$ in a binary-weighted graph, please devise a $O(V+E)$ algorithm to find the shortest path from $S$ to all the other vertices. Briefly justify the correctness.

**Solution:**
In a graph where all edges have equal weight, applying a normal breadth-first search catches the shortest path from the source vertex to every other vertices. In this problem, we will apply BFS with a minor modification.

Here we use a double-ended queue to store the nodes and perform BFS on the graph. If a vertex $u$ is dequeued, for each adjacent vertex $v$, there are two different cases:

(1) If the cost of edge $(u, v) = 0$, then $v$ is pushed at the front of the double-ended queue, and $v.d = u.d$.

(2) If the cost of edge $(u, v) = 1$, then $v$ is pushed at the back of the double-ended queue, and $v.d = u.d + 1$.

**Correctness:**
When a $u$ vertex is dequeued, it is always the vertex with the minimum distance among the remaining vertices. Running BFS using the double-ended queue ensure that the shortest path from the source vertex is caught as in a normal BFS.

**Running time:**
Running time of BFS = $O(V+E)$.

## Problem 3 (10%)

Sheldon lives in Pasadena, California. He'd like to join the comic-con that will take place in San Diego in three days. He decides to drive to San Diego; however, he is a poor driver. To avoid a car accident, he labeled each road between Pasadena and San Diego with the probability (a real number in [0, 1]) that he can drive safely on that road. If a car accident happens on his drive, Sheldon will not be able to join the comic-con and will become extremely "irritating"!
Please help his friend Leonard, who lives with Sheldon, to prevent such a tragedy from happening. Describe an efficient algorithm to find a path from Pasadena to San Diego that maximizes the probability of a safe drive by modeling the map as a **directed graph**. Briefly justify the correctness and analyze the running time.

*Hint*: Model all towns between Pasadena and San Diego as vertices and the road between two towns as two directed edges.

**Solution:**
Follow the hint, we have a directed graph $G(V, E)$, and each edge $e \in E$ have a weight $p_i$ indicating the probability of a safe drive on that road. Further assume that Pasadena is the vertex $s$, and San Diego is the vertex $t$. Convert each edge weight $p_i$ to $-(\log(p_i))$. Run Dijkstra's algorithm on the graph to find the shortest path from $s$ to $t$.

**Correctness:**
Intuitively, we would like to maximize $\prod_i p_i$ over the vertices in the path we take from $s$ to $t$. Since the log function is monotonic, it is the same as maximizing $\sum_i \log(p_i)$, which is the same as minimizing $-\sum_i \log(p_i) = \sum_i(-\log(p_i))$. Therefore, by creating an auxiliary graph in which the weight $w$ of each edge is replaced with $-\log(w)$, the shortest $s$-$t$ path is the maximum probability path. Additionally, $p_i \in [0, 1] \implies \log(p_i) \leq 0 \implies -(\log(p_i)) \geq 0$, so all edge weights in the auxiliary graph are nonnegative. Since all edge weights are nonnegative, we can apply Dijkstra algorithm to find the shortest $s$-$t$ path and it is the path with the maximum probability.

**Running time:**
The creation of the auxiliary graph takes $O(E)$ time and the Dijkstra's algorithm takes $O(E + V \log V)$ when using a Finbonacci heap. Thus, the total time complexity of the algorithm is $O(E + V \log V)$.

## Problem 4 (15%)

Zombie apocalypse suddenly breaks in South Korea! Jonathan and his daughter Amy want to travel from Seoul to Busan to meet with his beloved/her mother Isabelle. They have a map of all train routes between Seoul and Busan, which records the time it will take to travel from one station to the other. However, on some of the train routes, they will be attacked by the zombies. Jonathan has a confidence that he can protect his daughter from the zombie attack for **once** during their trip from Seoul to Busan. However, if they encounter the zombie attack more than once during the trip, they are destined to become one of those horrible monsters. Fortunately, Jonathan also has the information indicating whether each train route is free from zombie attacks or not.
Please help Jonathan design a **polynomial time algorithm** (to the number of train routes and stations) to let them travel from Seoul to Busan **as fast as possible** without becoming a zombie. Briefly justify the correctness and analyze the running time.

**Solution:**
We can model the train map between Seoul and Busan as a graph $G(V, E)$, with each train station being a vertex $v \in V$ and each train route being an edge $e \in E$. For each edge $e$, the weight of $e$ is the corresponding travel time of the train route. Also, assume a boolean attribute $z(e)$ for each edge $e$ indicates if they will encounter zombie attack on $e$. With the modeling, this problem is essentially a shortest path problem with at most one edge of the path that $z(e) = 1$.

To solve this problem, transform the graph $G(V, E)$ to get the graph $G' = (V', E')$ in the following way:

(1) For every vertex $v \in V$, create two vertices $v_s$ and $v_z$.

(2) For every edge $(u, v) \in E$ where $z((u, v)) = 0$, create edges $(u_s, v_s)$ and $(u_z, v_z)$ in $G'$.

(3) For every edge $(u, v) \in E$ where $z((u, v)) = 1$, create an edge $(u_s, v_z)$ in $G'$.

Run Dijkstra's algorithm to compute the shortest paths from $s_s$ to $t_s$ and from $s_s$ to $t_z$ and select the minimum of the two.

**Correctness:** The "zombied" vertices $v_z \in V'$ model the scenario when they encounter one zombie attack during the path. From the construction, we can guarantee that any path in $G'$ can have at most one "zombied" edge. As soon as a zombied edge is traversed, from the safe vertices region, we reach the zombied region and now there are no outgoing zombied edges from this region. Only safe edges can be traversed from this point onwards. Thus, running Dijkstra's algorithm on the auxiliary graph successfully catch the shortest path with at most one edge of the path that $z(e) = 1$.

**Running time:** Creating $G'$ from $G$ takes $O(V + E)$ and has $O(2V)$ vertices and $O(2E)$ edges. Running Dijkstra's algorithm takes $O(2E + 2V \log 2V) = O(E + V \log V)$. Thus, the total time complexity of the algorithm is $O(E + V \log V)$.

## Problem 5 (15%)

Given $n$ cities $C_1, C_2, ..., C_n$. The cost of constructing an undirected road between cities $C_i$ and $C_j$ requires cost $w_{ij}$. The cost of constructing an airport at city $C_i$ is $p_i$. One can travel between two cities using an airport if both cities have airports. Given all costs $w_{ij}$ and $p_i$, design a **polynomial time algorithm** to find the minimum cost set of airports and roads such that people from every cities can travel to any other cities. Briefly justify the correctness and analyze the running time.

**Solution:**
Construct a graph $G = (V, E)$ where each vertex $v \in V$ represents a city, and each edge $e = (v_i, v_j) \in E$ represents the undirected road between two cities (with weight $w_{ij}$).

There are two possibilities:

(1) The optimal solution does not use airports. In this case, ignore airports and construct the minimum spanning tree as usual.

(2) The optimal solution uses airports. In this case, add a dummy node called “sky” to the graph and connect all existing node to the sky. The cost of building a road from a city $C_i$ to the sky is the cost of building an airport $p_i$. The cost of the optimal solution using airports is the cost of the minimum spanning tree in this auxiliary graph.

Run Kruskal's algorithm on the two resulting graphs to get two minimum spanning trees. Choose the one with the smaller cost and it is the solution to the problem.

**Correctness:**
Consider constructing an airport at $C_i$ with cost $p_i$ as constructing a road from the “sky” to $C_i$ with cost $p_i$. Find the minimum spanning tree of the graph including the sky will connect all cities together. Also, if the optimal solution does not use airports, simply run the MST algorithm can get the solution. Thus, by choosing the smaller cost MST of the two, it is the optimal solution.

**Running time:**
There are $O(n)$ vertices and $O(n^2)$ edges. The running time of the Kruskal's algorithm is $O(E \log V) = O(n^2 \log n)$.

## Programming Problems

Note that in the programming problems, you **cannot import any module or library**. Also, you can only implement your program with **Pyhton3**. Only `p1.py` and `p2.py` (if you have finished the bonus problem) should be uploaded. Put the file(s) in a folder named {student id} and compress the folder as .zip or .tar format.

## Problem 1 (35%)

Professor Wang traveled to London for an important computer vision conference, ECCV. After receiving his fifth consecutive best paper award in the conference, he realized that no one can beat him in the community of computer vision and it's time to search for something more challenging. Thus, he went to London King's Cross and took a train at Platform $9\frac{3}{4}$ to Hogwarts to start his new life as a wizard.
There are so many magic courses he should take to graduate from Hogwarts. However, he finds that some of the magic courses have prerequisites. For example, before learning `Divination`, he should have learned `Charms` before.

Here comes the question. Given the total number of magic courses to take and a list of prerequisite pairs, return the minimum number of years it takes to graduate from Hogwarts and an ordering of courses he should take to finish all of them. Note that Hogwarts also have two semesters and some of the magic courses are only available in one of the two semesters. Suppose a semester is 0.5 year.

### Input Format

The first line contains two integers $n$ and $p$, indicating the number of courses a student should take to graduate, and the number of prerequisite pairs, respectively. The following $n$ lines contain two integers $i$ and $j$, where $i$ is the course id (from 0 to $n-1$) and $j$ indicates which semester the course is available (0 for the first semester, 1 for the second semester, and 2 for both of the semesters). The following $p$ lines contain two integers $x$ and $y$, indicating to take course $x$, you should have finished course $y$.

- $1 \leq n \leq 1 \times 10^4$
- $0 \leq p \leq 5 \times 10^5$

### Output Format

If there is no legal solution, output a single $-1$. If there is a legal solution, in the first line, please output how many years it takes to graduate. In the following lines, please output the set of course ids to take in each semester (separated by a space, and -1 for no class in this semester).

### Time Limit

15 seconds

### Sample Input 1

```
2 1
0 1
1 0
0 1
```

### Sample Output 1

```
1.0
1
0
```

### Sample Input 2

```
2 1
0 0
1 0
0 1
```

### Sample Output 2

```
1.5
1
−1
0
```

### Sample Input 3

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

### Sample Output 3

```
1.5
0
1 2
3
```

### Sample Input 4

```
2 2
0 2
1 2
0 1
1 0
```

### Sample Output 4

```
−1
```

## Problem 2 (bonus 20%)

We have calculated how many years it takes for Prof. Wang to graduate from Hogwarts. However, we forget to take the "magic power" a person possesses into consideration. Taking a magic course requires several points of "magic power". **In each semester**, the sum of "magic power" requirement of every courses cannot exceed a person's magic power limit; otherwise, he will become a Muggle.

Given the magic power limit $m$ of Prof. Wang, please determine how many years it takes for him to graduate from Hogwarts and an ordering of courses he should take.

### Input Format

The first line contains three integers $n$, $p$ and $m$, where $n$ and $p$ are the same as in problem 1, and $m$ indicates the magic power limit of Prof. Wang. The following $n$ lines contain three integers $i$, $j$, and $k$, where $i$ is the class id (from 0 to $n-1$), $j$ indicates which semester the class is available (0 for the first semester, 1 for the second semester, and 2 for both of the semesters), and $k$ indicates the magic power requirement for this course. The following $p$ lines contain two integers $x$ and $y$, indicating to take course $x$, you should have finished course $y$.

- $1 \leq n \leq 1 \times 10^5$
- $0 \leq p \leq 5 \times 10^6$
- $1 \leq m \leq 2 \times 10^3$

### Output Format

There is always a legal solution for each test case. In the first line, please output how many years it takes to graduate. In the following lines, please output the set of class ids to take in each semester (separated by a space, and -1 for no class in this semester).

### Time Limit

30 seconds

### Grading Policy

- Output any legal solution correctly in the time limit: bonus 10 points
- Help Prof. Wang graduate earlier than 50% of solutions in our class: bonus 5 points
- Help Prof. Wang graduate earlier than 80% of solutions in our class: bonus 5 points

**Sample Input 1**

```
3 2 25
0 2 25
1 2 1
2 2 1
2 0
2 1
```

**Sample Output 1**

```
1.5
0
1
2
```

**Solution:**
Reference program is available on Ceiba. Special thanks to Pei-Chieh Yuan for providing the reference source codes, which outperforms others in our testcases. Also, if you are interested in the testcase generator (and all testcases), the result checker, and the reference program written in C++, they are available in the following Github repository:
`https://github.com/FuyuChuang/algo_pa`