CS2040S: Data Structures and Algorithms

Discussion Group Problems for Week 11

For: March 30–April 3

Goals:

- Union-Find review
- Shortest paths
- Dijkstra's Algorithm

Problem 1. (Union-Find Review)
Problem 1.a. What is the worst-case running time of Union-Find with path compression (but no weighted union)?

Problem 1.b. Here's another algorithm for Union-Find based on a linked list. Each set is represented by a linked list of objects, and each object is labelled (e.g., in a hash table) with a set identifier that identifies which set it is in. Also, keep track of the size of each set (e.g., using a hash table). Whenever two sets are merged, re-label the objects in the smaller set and merge the linked lists. What is the running time for performing m Union and Find operations, if there are initially n objects each in their own set?

More precisely, there is: (i) an array *id* where $id[j]$ is the set identifier for object j; (ii) an array *size* where $size[k]$ is the size of the set with identifier k; (iii) an array *list* where $list[k]$ is a linked list containing all the objects in set k.

```
Find(i, j)
return (id[i] == id[j])
Union(i, j)
if size[i] < size[j] then Union(j,i)
else // size[i] >= size[j]
k1 = id[i]
k2 = id[j]
for every item j in list[k2]: set id[j] = k1
append list[k2] on the end of list[k1] and set list[k2] to null
size[k1] = size[k1] + size[k2]
size[k2] = 0
```

Assume for the purpose of this problem that you can append one linked list on to another in $O(1)$ time. (How would you do that?)

Problem 1.c. Imagine we have a set of n corporations, each of which has a (string) name. In order to make a good profit, each corporation has a set of jobs it needs to do, e.g., corporation j has tasks $T^j[1 \dots m]$. (Each corporation has at most m tasks.) Each task has a priority, i.e., an integer, and tasks must be done in priority order: corporation j must complete higher priority tasks before lower priority tasks.

Since we live in a capitalist society, every so often corporations decide to merge. Whenever that happens, two corporations merge into a new (larger) corporation. Whenever that happens, their tasks merge as well.

Design a data structure that supports three operations:

- `getNextTask(name)` that returns the next task for the corporation with the specified name.
- `executeNextTask(name)` that returns the next task for the corporation with the specified name and removes it from the set of tasks that corporation does.
- `merge(name1, name2, name3)` that merges corporation with names `name1` and `name2` into a new corporation with `name3`.

Give an efficient algorithm for solving this problem.

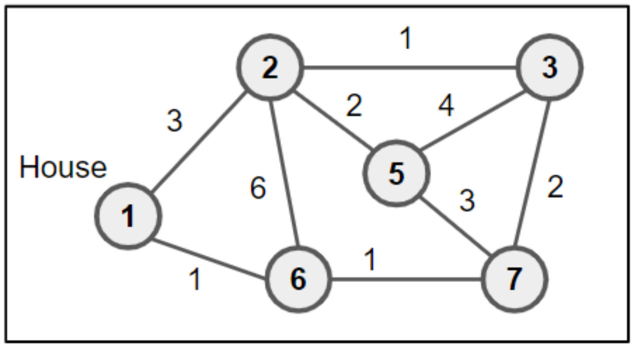
Problem 2. Elephant Encounters
Related Kattis Problems:

- https://open.kattis.com/problems/arbitrage
- https://open.kattis.com/problems/getshorty

Travelling can be a risky proposition: you never know when you will meet a large pink elephant. Luckily, you have been given a map where each road segment is labelled with the exact probability that someone driving on that road will encounter a large pink elephant. You want to drive from New York to Los Angeles. What route should you take so as to have the smallest chance of meeting a large pink elephant?

More formally, you are given a directed graph $G = (V, E)$, where every edge e has an independent safety probability $p(e)$. The safety of a path is the product of the safety probabilities of its edges. Design and analyze an algorithm to determine the safest path from a given start vertex s to a given target vertex t.

Problem 3. Running Trails



I want to go for a run. I want to go for a long run, starting from my home and ending at my home. And I want the first part of the run to be only uphill, and the second part of the run to be only downhill. I have a trail map of the nearby national park, where each location is represented as a node and each trail segment as an edge. For each node, I have the elevation (value shown in the node). Find me the longest possible run that goes first uphill and then downhill, with only one change of direction.

Problem 4. (Bad Dijkstra)
Give an example of a graph where Dijkstra's Algorithm returns the wrong answer.

Problem 5. (A Random Problem with a dude called Dan)
Problem 5.a. Dan is on his way home from work. The city he lives in is made up of N locations, labelled from 0 to $(N-1)$. His workplace is at location 0 and his home is at location $(N-1)$. These locations are connected by M **directed** roads, each with an associated *(non-negative)* cost. To go through a road, Dan will need to pay the cost associated with that road. Usually, Dan would try to take the cheapest path home.

The thing is, Dan has just received his salary! For reasons unknown, he wants to flaunt his wealth by going through a *really expensive road.* However, he still needs to be able to make it back home with the money he has. Given that Dan can afford to spend up to D dollars on transportation, help him find **the cost of the most expensive road that he can afford to go through** on his journey back home.

Take note than Dan only cares about the most expensive road in his journey; the rest of the journey can be really cheap, or just as expensive, so long as the entire journey fits within his budget of D dollars. He is also completely focused on this goal and does not mind visiting the same location multiple times, or going through the same road multiple times.

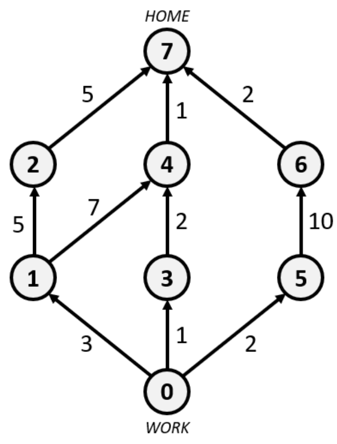


Figure 1: Example city 1

For example, suppose Dan's budget is $D = 13$ dollars. Consider the city given in Figure 1, consisting of $N = 8$ locations and $M = 10$ roads.

The path that Dan will take is $0 \to 1 \to 4 \to 7$. In this journey, the total cost is 11 dollars and the most expensive road has a cost of 7 dollars - the road from locations 1 to 4. Therefore, the expected output for this example would be "7".

Note that this path is neither the cheapest path ($0 \to 3 \to 4 \to 7$), nor is it the most expensive path that fits within his budget of 13 dollars ($0 \to 1 \to 2 \to 7$).

There is also a more expensive road within this city - the road from locations 5 to 6 with a cost of 10 dollars. However, the only path that goes through this road, $0 \to 5 \to 6 \to 7$, has a total cost of 14 dollars which exceeds Dan's budget.

Problem 5.b. *(Optional)* Another month, another salary for Dan to flaunt. The situation is similar to that of the previous part.

This time, however, instead of maximizing the cost of the *most expensive road* in his journey, he wants to maximize the cost of *the second most expensive road* in his journey. In other words, he no longer cares about the most expensive road in his journey; that road can be 100 times more expensive than the second most expensive road in his journey for all he cares.

For example, consider the city in Figure 2 with $N = 4$ locations and $M = 5$ roads.

If Dan's budget is $D = 12$ dollars, the path that he will take is $0 \to 2 \to 3$. In this journey, the total cost is 11 dollars and the second most expensive road has a cost of 5 dollars, the road from locations 0 to 2. Therefore, the expected output for this example would be "5".

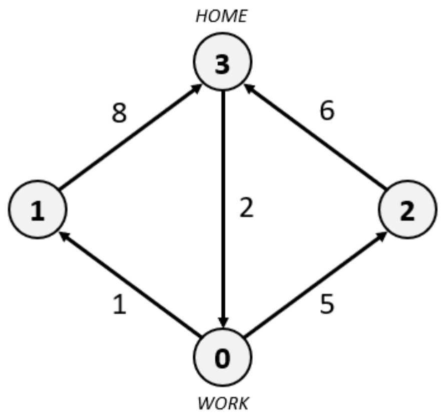


Figure 2: Example city 2

Notice that while he can afford to go through the path $0 \to 1 \to 3$ with an expensive 8 dollar road, the second most expensive road in that journey only costs 1 dollar.

If Dan's budget is $D = 20$ dollars, then the path he will take is $0 \to 1 \to 3 \to 0 \to 1 \to 3$. As irrational as this 20 dollar journey is, it allows him to go through the road from locations 1 to 3 twice, thus making the second most expensive road in his journey cost 8 dollars.

Problem 6. (Costly Cycles)

Given a weighted directed graph $G = (V, E)$ in which each edge has a weight between 0 and 1: we say that a cycle with c edges is *costly* if the sum of the weights is $> c - 1$.

Problem 6.a. Give an $O(V^3 \log V)$ algorithm for finding the minimum cost (directed) cycle in G.

Problem 6.b. Give an $O(V^3 \log V)$ algorithm for determining whether G has a *costly* cycle. (Hint: use part (a) on a graph with modified edge weights such that a costly cycle has a small total weight.)