CS5800 Algorithms *Out: 5 October 2021*

Solutions to Problem Set 3

Name: Lei Zhang *Due: 15 October 2021*

Group members:

Problem 1 (Greedy Coins) 10+5

1. Recall the change-making algorithm from lecture. Suppose that the available coins are in the denominations that are powers of c, i.e., $c^0, c^1, \ldots, c^k$ for some integers $k > 0$, $c > 1$. Prove that the greedy algorithm always yields an optimal solution.
2. Give a set of coin denominations for which the greedy algorithm does not give an optimal solution. Your set should include a penny so there's a solution for every change amount n.

Solution:

1. Proof: The optimal solution must follow the rule that the number of the coins in the denominations of c^i where $0 \le i < k$ has to be less than c. Because any c coins in the denomination of c^i where $0 \le i < k$ can be replaced by one coin of c^{i+1}, thus indicating the solution is not optimal. As the greedy algorithm ensures that there won't be c identical coins of value between c^0 and c^{k-1}, it always yields the optimal solution.
2. Coinage: $\{1, 10, 25\}$.
   For value of 30, the greedy algorithm gives the solution of $[25, 1, 1, 1, 1, 1]$, but the optimal solution is $[10, 10, 10]$.

■

Problem 2 (Super or normal) 25

You have one supercomputer and n normal computers on which you need to run n jobs. Each job i first spends s_i time on the supercomputer and then n_i time on the normal computer. A job can only start running on any of the normal computers after it has finished on the supercomputer. However, as soon as any job finishes on the supercomputer, it can immediately start on one of the free normal computers. The goal is to finish running all the jobs as soon as possible. Note that since there is only one supercomputer, you'll always have to wait $\sum_{i=1}^{i=n} s_i$ time so that the jobs finish running on the supercomputer. However, you can optimize when you run the jobs on the normal computers to try to finish running all the jobs as soon as possible. Show the following schedule is optimal: execute jobs after sorting them in *decreasing order* by n_i.

Solution:

The idea is to determine which job should be run last. As the last job always has to wait for $\sum_{i=1}^{i=n} s_i$ time to run on a normal computer, the optimal should run the job with minimum n_i last. Similarly, jobs with smaller n_i will run later.

Proof by induction:
Base case: The job with minimum n_i will be the last to run.

Inductive step: Assume that the last k jobs to run are the k jobs with minimum n_i time, i.e., the last k terms from $(n-k+1)$ to n when sorted in descending order by n_i.
Let S_k be the sum of supercomputer time of the last k jobs. Then, the in no matter what order, the $(n-k)$th job has to wait for $\sum_{i=1}^{i=n} s_i - S_k$ time to run on a normal computer. Thus, it should be the job with the least n_i time, i.e., the $(n-k)$th term of the list where jobs are sorted in descending order by n_i.
Therefore, the last $k+1$ jobs to run are the jobs with the minimum n_i time.
Q.E.D. ■

Problem 3 (Distinct Edge Weights and MSTs) 10+10+10

Assume that all the weights in the undirected graph G are unique (no two edge weights are the same). Prove the following, or provide a counter example.

a. The MST is unique.

b. The edge with the smallest weight e_{min} always belongs to the MST.

c. The edge with the largest weight e_{max} never belongs to the MST.

Solution:

1. True.
   Proof: Suppose that there are two MSTs T_1 and T_2, and let e be the lightest edge that exists in only one of the MSTs, say e is in T_1. Then add e to T_2 will create a cycle and there must be another edge f on the cycle that only exists in T_2 and that e is lighter than f since e is the minimum edge that exists in one of the MSTs. Thus, $T^* = T_2 \cup \{e\} - \{f\}$ is also a MST and that $w(T^*) < w(T_2)$, which contradicts that T_2 is a MST.

2. True.
   Proof: Suppose that the lightest edge e does not belong to the MST T. Then add e to T will create a cycle and there must be another edge f on the cycle that is heavier than e. And $T^* = T \cup \{e\} - \{f\}$ is also a MST and that $w(T^*) < w(T)$, which contradicts that T is a MST.

3. False.
   Proof: Consider a linear graph where each edge is on the same line. Then MST for this graph must contain the heaviest edge.

■

Problem 4 (Programming: An Escape from time) 30

Ulrich from 2019 travels to Winden 1953 to accomplish a secret mission. He travels using the popular Winden Caves. After he has finished his secret mission in Winden - 1953, he now has to return to the 2019. He can travel to 2019 using the caves. However, the cops of Winden know that Ulrich is on the loose, and are attempting to stop him by spreading his picture to the police at their bus stop. You must help Ulrich escape Winden-1953 by determining which busses he should take to reach the caves. However, because the police are actively looking for him, Ulrich has decided that he cannot sit still at any one bus stop for too long. Thus you need to determine which busses he should take so that his layover is never too long. **Input Format**: Line 1 : N M T

– N is the max label of any bus station (1 is the starting point, N is the destination outside the country). Ulrich always starts at station 1 at time 0.

- M is the number of bus rides which appear on the schedule.
- T is the max time Ulrich can spend at any one station without being caught by the Winden Police. Note that he is allowed to spend exactly T time steps at a station, but not a moment more.

Next M Lines: x y t_1 t_2.

- x is the station label where a bus will be leaving from
- y is the station label where this bus will arrive
- t_1 is the time the bus will leave station x.
- t_2 is the time the bus will arrive at station y.

Output Format: "NO" if, no matter what busses Ulrich takes, he will be caught before reaching station N. "YES T_{min}" if he can escape , where T_{min} is the earliest possible time that he arrives at station N.

At the very start, Ulrich is at station 1, at time 0. Suppose that Ulrich arrives at station x at time t. He can move to station y if and only if:

- There is a bus going from x to y at time t_1, where $t_1 > t$, but $t_1 - t \leq T$, the max time he can spend at the station.
- If there is such a bus ride, labeled as x, y, t_1, t_2, then after taking this bus, he will be at station y at time t_2. Furthermore, he will have spent $t_1 - t$ time units waiting at station x.

Sample Input: 4 5 20 1 2 15 20 1 3 1 6 2 4 25 30 3 4 2 7 3 4 30 35

Sample Output: YES 30

Explanation: Ulrich could take the following busses: * Stop 1 to Stop 2, leaves at 15, arrives at 20 * Stop 2 to Stop 4, leaves at 25, arrives at 30

The only other bus that arrives earlier at stop 4 leaves stop 3 at time 2, but it is not possible to get to stop 3 before time 2.

Solution:

1. HankerRank username: **zhang_lei1**
2. Submission handle: https://www.hackerrank.com/contests/cs5800-fall21-pset-3/challenges/cs5800-spy-escape/submissions/code/1337704734

■