EL-GY 9343 Homework 11

Yihao Wang
yw7486@nyu.edu

1. Design a greedy algorithm for arranging the queuing order in a supermarket. Suppose there are n customers come to the counter at the same time, noted as $c_1, c_2, \dots, c_n$, the time to service i-th customer is $s_i, i = 1, 2, \dots, n$, and the absolute time to finish i-th customer is $T_i, i = 1, 2, \dots, n$. Your goal is to decide a queuing order of n customers to minimize the accumulated completion time (waiting time + service time) of all n customers, that is, to minimize $\sum_{i=1}^{n} T_i$.

(a) Provide an algorithm to solve this issue;

Sort n customers according to service time s_i, and re-index c_i in ascending order. Now c_i is the optimal queuing order that minimize $\sum_{i=1}^{n} T_i$.

(b) Prove the correctness of your algorithm by showing the greedy choice property and optimal substructure;

Assume the customers' indices have been updated and they now indicate the actually serving order. The completion time of the k-th customer, T_k, can be calculated as

$$T_k = s_1 + s_2 + \cdots + s_k = \sum_{i=1}^{k} s_i$$

For the first k customers, the total accumulated serving time is

$$\sum_{i=1}^{k} T_i = k \cdot s_1 + (k-1) \cdot s_2 + \cdots + 1 \cdot s_k = \sum_{i=1}^{k} (k+1-i) \cdot s_i \qquad (1)$$

greedy choice property: The algorithm selects the customer with the shortest service time first, ensuring that the time being accumulated the most times in the waiting time is the shortest service time, as shown in (1). This choice is made at each step, resulting in a locally optimal solution.

optimal substructure: By serving the customer with the shortest service time first, the algorithm creates a subproblem with only the first k customers. The queuing order for the remaining customers can be determined using the same greedy algorithm. This means that the optimal solution for the entire problem can be built from the optimal solutions of its subproblems.

(c) Justify the running time of your algorithm.

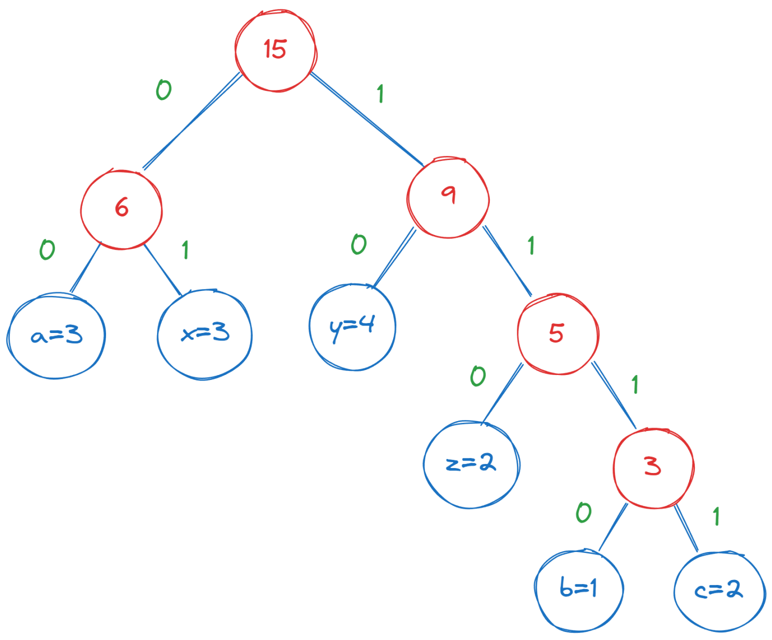
The sorting contributes to all of the running time, which is $O(n \log n)$.

2. How many bits are required to encode the message "aaabccxxxyyyyzz" using Huffman Codes? Please show the coding tree you build.

First, we need to count the frequency of each character and sort them in ascending order:

b	c	z	a	x	y
1	2	2	3	3	4

Then we can build the prefix tree accordingly



Therefore, the Huffman codes for each character are

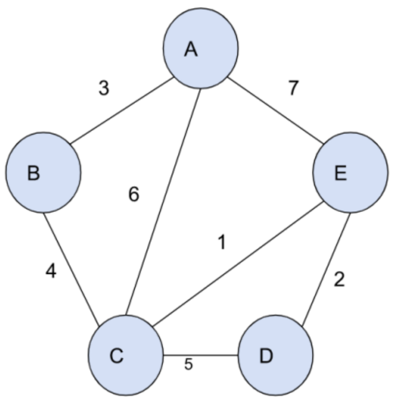
b	c	z	a	x	y
1110	1111	110	00	01	10

In total, we need

$$4 \times (1+2) + 3 \times 2 + 2 \times (3+3+4) = 38 \text{ bits}$$

to encode the message "aaabccxxxyyyyzz".

3. Consider the following graph.



(a) If we run Kruskal's algorithm in the graph, what will be the sequence in which edges are added to the MST?

The sequence will be $\{(C, E), (D, E), (A, B), (B, C)\}$

(b) Demonstrate Prim's algorithm in the graph with A as the source.

1. At first, solution vertex set $S = \{A\}$. Now the accessible edge with minimum weight is (A, B), so we add B to S.
2. $S = \{A, B\}$. Now the accessible edge with minimum weight is (B, C), so we add C to S.
3. $S = \{A, B, C\}$. Now the accessible edge with minimum weight is (C, E), so we add E to S.
4. $S = \{A, B, C, E\}$. Now the accessible edge with minimum weight is (D, E), so we add D to S.
5. In the end, the final soulution set $S = \{A, B, C, D, E\}$

4. Let G be an n-vertex connected undirected graph with costs on the edges, and the number of edges is $m = O(n^2)$. Assume that all the edge costs are distinct.

(a) Prove that G has a unique MST;

Because all the edge costs are distinct, there is an unique sorting of all edges, according to which we can run Kruskal's algorithm in the graph. Therefore, there will generate a uniquely determined MST.

(b) Give an algorithm to find a cycle in G such that the maximum cost of edges in the cycle is minimum among all possible cycles. The time complexity of your algorithm should be $O(m \log n)$. Assume that the graph has at least one cycle, and finding MST takes $O(m \log n)$ time.

1. First, find the MST. During the finding we also have been tracking the edge of minimum cost.
2. Then, among all edges that are not in MST, find the edge of minimum cost, which becomes the edge asked to find.

The second step takes at most $O(m)$ time, Thus the total time complexity is $O(m \log n) + O(m) \sim O(m \log n)$.