Ocean Lu
CS 3310
Professor Damavandi / Johannsen
10/17/19

Homework #3

1. Show with a counterexample that the greedy approach does not always yield an optimal solution for the Change problem when the coins are U.S. coins and we do not have at least one of each type of coin.
   A counterexample where the greedy approach does not yield an optimal solution would be a situation where the coins given are ={1, 10, 25, 50, 100}, and the value we want is 30. The greedy algorithm's result is {25, 1, 1, 1, 1, 1}, with six coins. The optimal solution is {10, 10, 10}, where only three coins are needed. This example proves that the greedy approach does not yield an optional solution for the Change problem.

2. Suppose we assign n persons to n jobs. Let Cij be the cost of assigning the ith person to the jth job.
   a. Use a greedy approach to write a simple algorithm that finds an assignment that attempts to minimize the total cost of assigning all n persons to all n jobs. Analyze your algorithm by determining its order.

```
for (i = 0; i < n; i++) {
        jobAssignedTo = -1;
        personAssignedTo = -1;
}
for  (person = 0; person < n; person++) {
        foundUnassignedJob = false;
        for (job = 0; job < n; job++) {
                if (jobAssignedTo == -1) {
                        if (foundUnassignedJob) {
                                if (minAssignment > cost[person][job]) {
                                        minAssignment = cost[person][job];
                                        personAssignedTo[person] = job;
                                        minJob = job;
                                }
                        } else {
                                foundUnassignedJob = true;
                                minAssignment = cost[person][job];
                                minJob = job;
                        }
                }
        } personAssignedTo[person] = minJob;
}
```

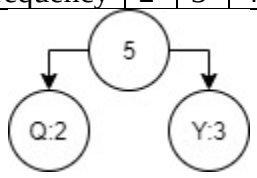
b. Is there a counterexample that shows your algorithm is not always optimal?
In my algorithm, I'll know the cost for each job whether the person is allocated to a job or not, and I search the array for the person with the lowest cost for every job and mark it as allocated to the job. The outer loop runs n times for n jobs and finding the person with the lowest cost runs n times. Thus, my time complexity is $O(n^2)$, that is not always optimal.

3. Use Huffman's Algorithm to construct an optimal binary prefix code for the letters in the following table:

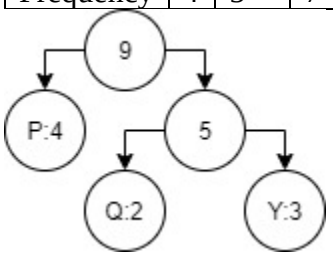
Letter	C	A	P	Y	E	D	Q
Frequency	10	15	4	3	18	7	2

Frequency Sort:

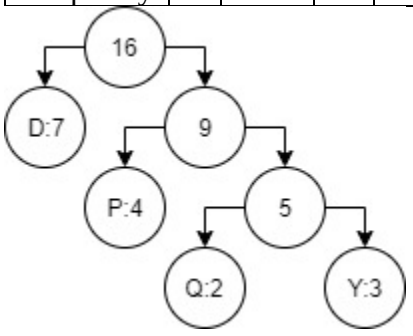
Letter	Q	Y	P	D	C	A	E
Frequency	2	3	4	7	10	15	18



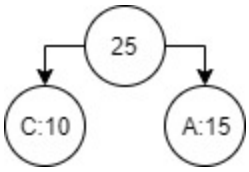
Letter	P	QY	D	C	A	E
Frequency	4	5	7	10	15	18



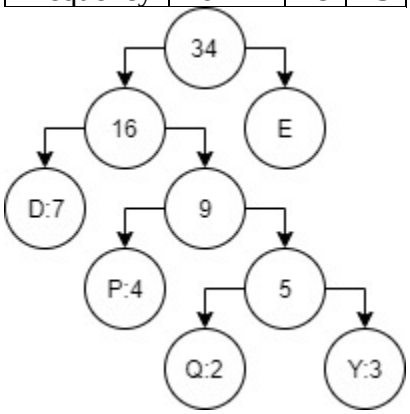
Letter	D	PQY	C	A	E
Frequency	7	9	10	15	18



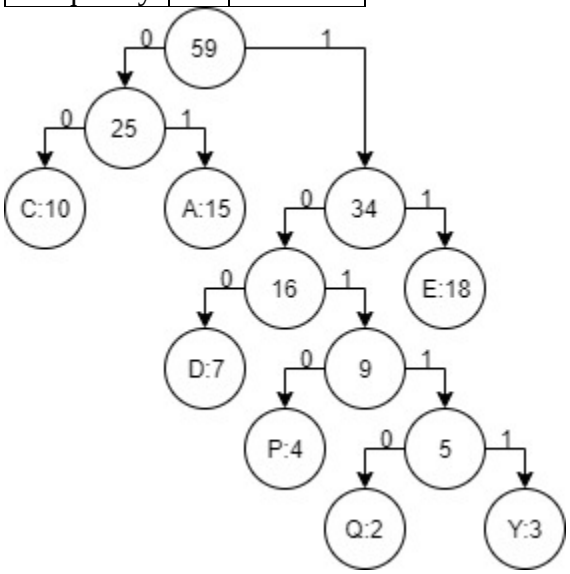
Letter	C	A	DPQY	E
Frequency	10	15	16	18



Letter	DPQY	E	C
Frequency	16	18	25



Letter	C	DPQYE
Frequency	25	34



Letter	Prefix Code
E	11
A	01
C	00
D	100
P	1010
Y	10111
Q	10110

4. Use Dijkstra's Algorithm to find the shortest path from v3 to all the other vertices in the following graph. Show the values in length and touch after each step.

	1	2	3	4	5	6
1	0	∞	1	5	3	8
2	∞	0	14	2	5	5
3	1	∞	0	∞	∞	10
4	5	8	∞	0	2	2
5	∞	8	3	2	0	9
6	2	7	10	5	8	0

Source - Destination	Path	Length
v3 - v1	**3 - 1**	**1**
	3 - 6 - 1	11
v3 - v2	3 - 2	∞
	3 - 6 - 2	**17**
v3 - v4	3 - 4	∞
	3 - 1 - 4	**6**
v3 - v5	3 - 5	∞
	3 - 6 - 5	18
	3 - 1 - 4 - 5	**8**
v3 - v6	**3 - 6**	**10**
	3 - 1 - 4 - 5 - 6	17

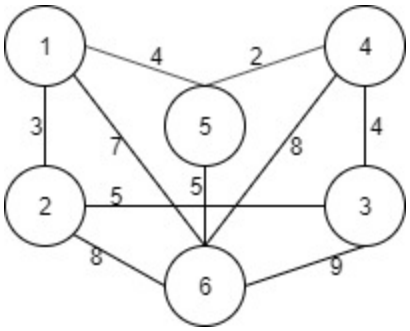
Final solution for graph:

Vertices	Path	Shortest Length
v3 - v1	3 - 1	1
v3 - v2	3 - 6 - 2	17
v3 - v4	3 - 1 - 4	6
v3 - v5	3 - 1 - 4 - 5	8
v3 - v6	3 - 6	10

5. Consider the following adjacency matrix

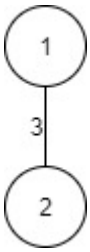
	1	2	3	4	5	6
1	0	3	∞	∞	4	7
2	3	0	5	∞	∞	8
3	∞	5	0	4	∞	9
4	∞	∞	4	0	2	8
5	4	∞	∞	2	0	5
6	7	8	9	8	5	0

   a. Use Prim's algorithm to find a minimum spanning tree. Show the values in the arrays nearest and distance after each step.

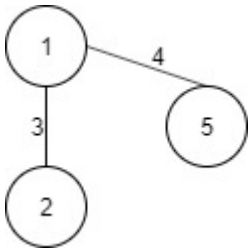


i. Ensure there is no loop and parallel edges in the graph; select vertex 1 as the root: array A = {1}

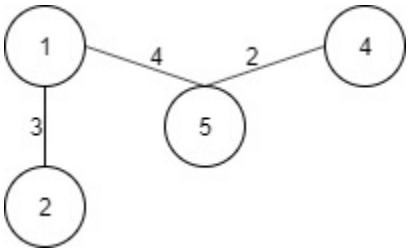
ii. **1 – 2 = 3**
1 – 6 = 7
1 – 5 = 4
A = {1, 2}



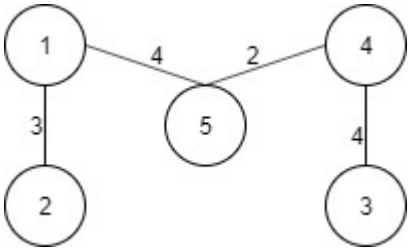
iii. **1 – 5 = 4**
1 – 6 = 7
2 – 3 = 5
2 – 6 = 8
A = {1, 2, 5}



iv. 1 – 6 = 7
2 – 3 = 5
2 – 6 = 8
5 – 4 = 2
5 – 6 = 5
A = {1, 2, 4, 5}

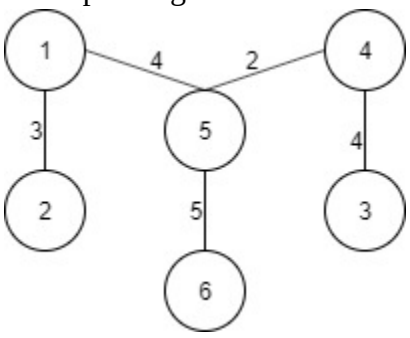


v. 2 – 6 = 8
1 – 6 = 7
4 – 3 = 4
4 – 6 = 8
5 – 6 = 5
2 – 3 = 5
A = {1, 2, 3, 4, 5}



vi. 1 – 6 = 7
2 – 6 = 8
2 – 3 = 5
3 – 6 = 9
4 – 6 = 8
5 – 6 = 5

A = {1, 2, 3, 4, 5, 6};
final spanning tree:

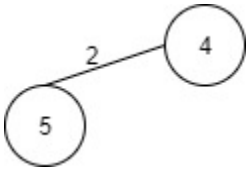


b. Use Kruskal's algorithm to find a minimum spanning tree. Show the disjoint sets as well as the edges in F after each step.

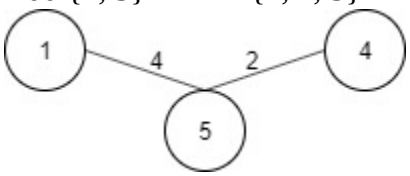
i. Let array A = {{1}, {2}, {3}, {4}, {5}, {6}}

Edge	Weight
{4, 5}	2
{1, 2}	3
{1, 5}	4
{4, 3}	4
{2, 3}	5
{5, 6}	5
{2, 6}	8
{3, 6}	9

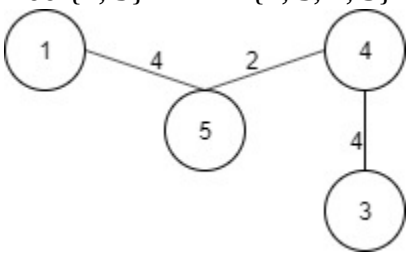
ii. Add {4, 5} → A = {4, 5}



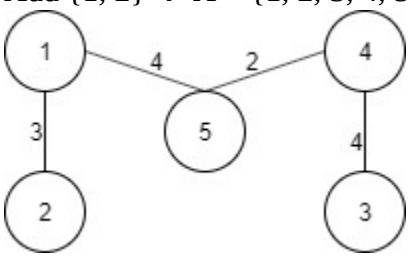
iii. Add {1, 5} → A = {1, 4, 5}



iv. Add {4, 3} → A = {1, 3, 4, 5}



v. Add {1, 2} → A = {1, 2, 3, 4, 5}



vi. Add {5, 6} → A = {1, 2, 3, 4, 5, 6}; final spanning tree: