CST 370 – Summer A 2020
Homework 4
Due: 05/19/2019 (Tuesday) (11:55 PM)

Name:____Andrew Bell_____

ID: _____1138_____

How to turn in?

Write your answer to the questions from 1 to 4 on a PDF file and submit it on the iLearn. Note that we **accept only a PDF** file. Do not submit a different file format. Also, don't forget to write your name and class ID at the top of your homework document.

For the questions 5, 6, and 7, you should submit three source programs on the iLearn.

Thus, you have to submit total four files (one PDF file and three source files) on the iLearn.

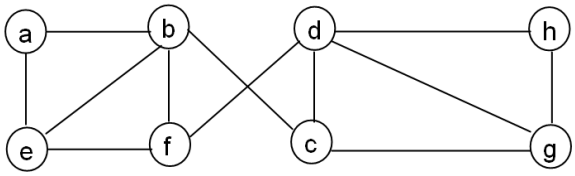
1. Assume that you want to solve a **knapsack problem**. Your knapsack capacity is "**6**" and there are five items as below. Note that **each item has only one** quantity.

Item	Capacity	Value
A	3	$25
B	2	$20
C	1	$15
D	4	$40
E	6	$50

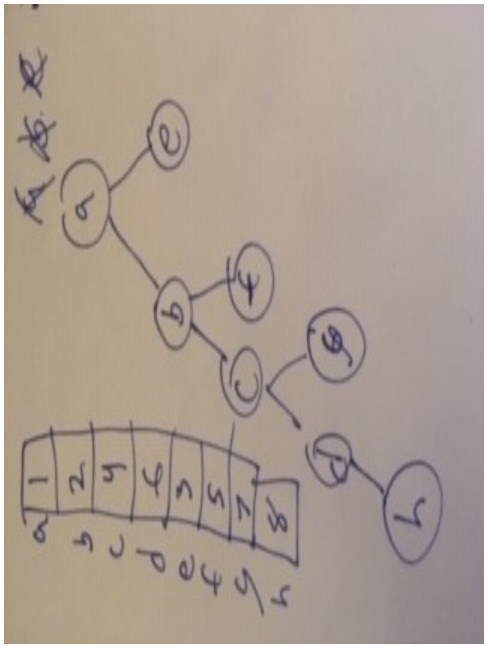
(a) To bring the maximum value, **which item(s) will you bring to the knapsack**?
A,B,C or B,D

(b) Do you have **only one optimal solution (= set)**? (Yes/**No**) No

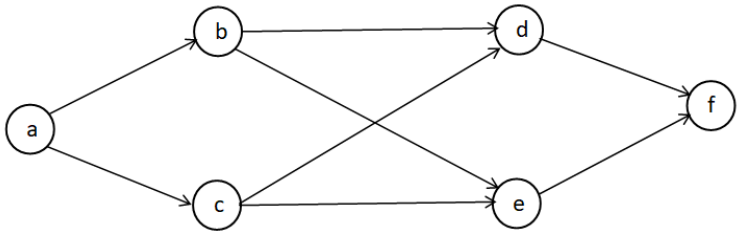
2. Consider the following graph.



Starting at vertex ***a***, traverse the graph by the **breadth-first search** algorithm, and present a **mark array** and a **breadth-first search tree** as you learned in the lecture. For the algorithm, you have to use **our convention** of the class (= **ascending order for the alphabetical characters**).

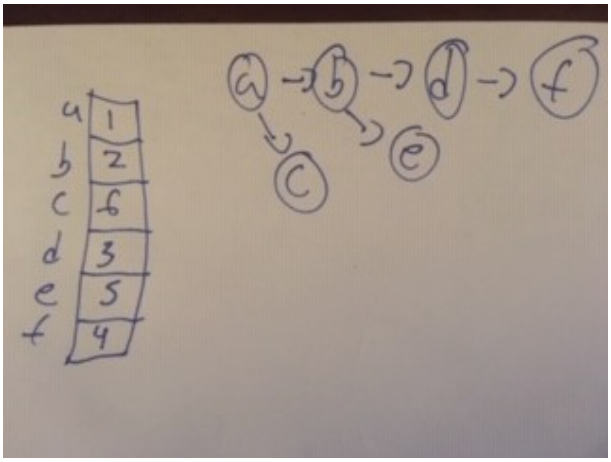


[**Note**: If it is difficult to draw the results on the word file, draw them on paper by hand. Then take a picture of the paper and attach the picture here.]3. Consider the following **directed graph**.



Starting at the vertex *a*, traverse the graph by **depth-first search** and present the **mark array** and a **DFS tree** as you learned in the lecture. When you traverse the tree, you should follow our convention in the class (= alphabetical order).

[**Note**: If it is difficult to draw the results on the word file, draw them on paper by hand. Then take a picture of the paper and attach the picture here.]



4. Consider the Master Theorem as we covered in the lecture.

$T(n) = aT(n/b) + f(n)$ where $f(n) \in \Theta(n^d)$, $d \geq 0$

Master Theorem: If $a < b^d$, $T(n) \in \Theta(n^d)$
If $a = b^d$, $T(n) \in \Theta(n^d \log n)$
If $a > b^d$, $T(n) \in \Theta(n^{\log_b a})$

Based on the theorem, determine the time efficiency of the following formula T(n).

(a) $T(n) = 2 * T(n/2) + n^2$
a = 2, b = 2, d = 2
$2 < 2^2$ $T(n) \in \Theta(n^d)$

(b) $T(n) = 4 * T(n/2) + 5n$
a = 4, b = 2, d = 1
$4 > 2^1$: $T(n) \in \Theta(n^{\log_2 4})$

==========

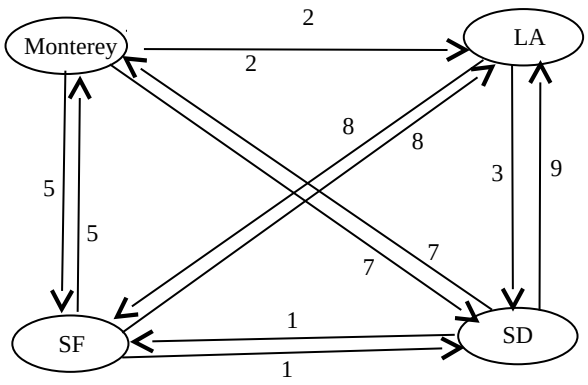
This is the HackerRank link: https://www.hackerrank.com/cst370-su20-hw4

5. Write a C++ (or Java) program called **hw4_1.cpp or (hw4_1.java)** that reads an input graph data from a user first. Then, it should present a path for the travelling salesman problem (TSP). In the assignment, you can assume that the **maximum number of vertices** in the input graph is **less than or equal to 20**.

Input format: This is a sample input from a user.

```
4
Monterey
LA
SF
SD
12
Monterey LA 2
Monterey SD 7
Monterey SF 5
LA Monterey 2
LA SF 8
LA SD 3
SF Monterey 5
SF LA 8
SF SD 1
SD Monterey 7
SD LA 9
SD SF 1
```

The first line (= 4 in the example) indicates that there are four vertices in the graph. The following four lines describe the names of four cities such as Monterey, LA, SF, and SD. For the problem, you should assume that the first city (= Monterey in the example) is the starting city of the TSP. In addition, you can assume that the city name is always one word. The next line (= 12 in the example) indicates the number of edges in the graph. The remaining 12 lines are the edge information with the “source city”, “destination city”, and “cost”. Thus, the input data describes a directed graph like below:



Sample Run 0: Assume that the user typed the following lines

4

```
Monterey
LA
SF
SD
12
Monterey LA 2
Monterey SD 7
Monterey SF 5
LA Monterey 2
LA SF 8
LA SD 3
SF Monterey 5
SF LA 8
SF SD 1
SD Monterey 7
SD LA 9
SD SF 1
```

This is the correct output. Your program should present the path and total cost in separate lines.

```
Path:Monterey->LA->SD->SF->Monterey
Cost:11
```

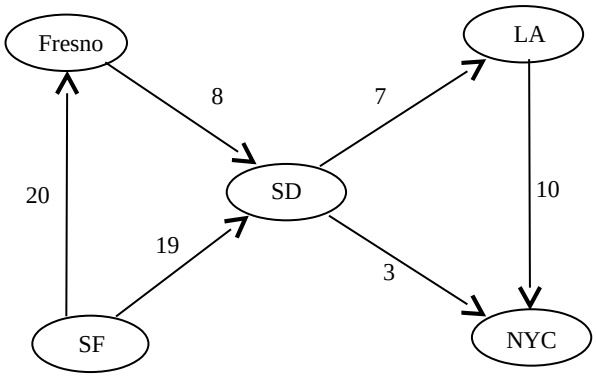
Sample Run 1: Assume that the user typed the following lines

```
5
Fresno
LA
SD
SF
NYC
6
Fresno SD 8
SD LA 7
SD NYC 3
LA NYC 100
SF Fresno 20
SF SD 19
```

This is the correct output.

```
Path:
Cost:-1
```

Note that if there is no path for the TSP, your program should **present empty path and -1 cost**.



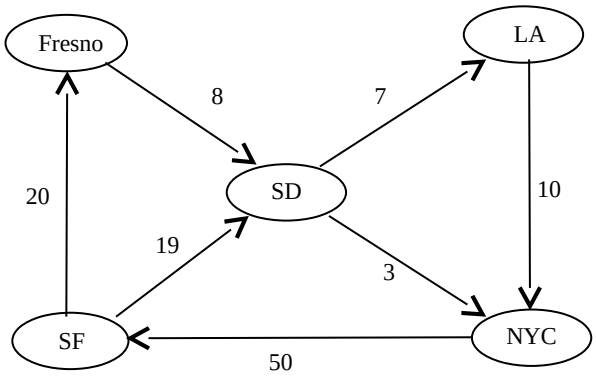
Sample Run 2: Assume that the user typed the following lines

```
5
Fresno
LA
SD
SF
NYC
7
Fresno SD 8
SD LA 7
SD NYC 3
LA NYC 100
SF Fresno 20
SF SD 19
NYC SF 50
```

This is the correct output of your program.

```
Path:Fresno->SD->LA->NYC->SF->Fresno
Cost:185
```

This is the directed graph of the input data:



[Hint]: To solve this problem, you can use all permutations with the cities, except the origin city. For example, there are three cities, LA, SF, and SD in the first sample run, except the origin city Monterey. To make the explanation easy, you can think that LA is number 1, SF is number 2, and SD is number 3. This is all permutations with the three cities

1, 2, 3
1, 3, 2
2, 1, 3,
2, 3, 1
3, 1, 2
3, 2, 1

For each permutation, you can calculate the cost. For example, in the case of the permutation "1, 2, 3", add the origin city at the very beginning and end to the permutation which generates "0, 1, 2, 3, 0". This list corresponds to the path "Monterey → LA → SF → SD → Monterey". Therefore, the cost of this path is "18". Similarly, the next permutation (= 1, 3, 2) corresponds to "Monterey → LA → SD → SF → Monterey". The cost is 11. This way, you can check all possible paths for the TSP.

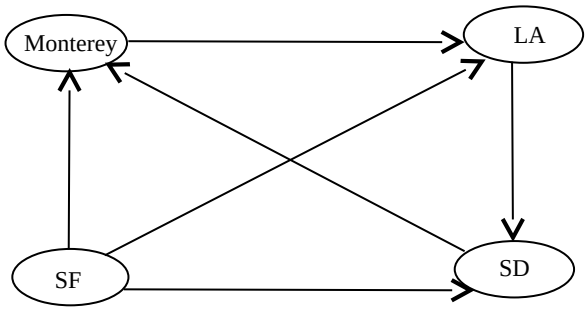
This is a sample C++ program to display permutations: https://repl.it/@YBYUN/permutationsusingSTL

6. Write a C++ (or Java) program named **hw4_2.cpp** (or **hw4_2.java**) that displays the city names in the range of hop(s) from a starting city. For this problem, refer to the sample BFS program at https://repl.it/@YBYUN/BFS

Input format: This is a sample input from a user.

```
4
Monterey
LA
SF
SD
6
Monterey LA
LA SD
SD Monterey
SF Monterey
SF LA
SF SD
Monterey
2
```

The first line (= 4 in the example) indicates that there are four vertices in the graph. The following four lines describe the names of four cities. The next line (= 6 in the example) indicates the number of edges in the graph. The following six lines are the edge information with the "source city" and "destination city". The following city (= Monterey in the example) is the starting city, and the last line (= 2 in the example) indicates the hops from the starting city. Thus, the problem is to list the city names that can be reached from the source in two hops. This is the graph with the input information.



Sample Run 0: Assume that the user typed the following lines

```
4
Monterey
LA
SF
SD
6
Monterey LA
LA SD
SD Monterey
SF Monterey
SF LA
```

```
SF SD
Monterey
2
```

This is the correct output. Your program should present the **city names in the ascending order** like below. Note that LA can be reached from Monterey in one hop, Monterey in zero hop, and SD in two hops. For the problem, you can **assume that the city name is always one word**.

```
LA
Monterey
SD
```

Sample Run 1: Assume that the user typed the following lines

```
4
Monterey
LA
SF
SD
6
Monterey LA
LA SD
SD Monterey
SF Monterey
SF LA
SF SD
LA
1
```

This is the correct output.

```
LA
SD
```

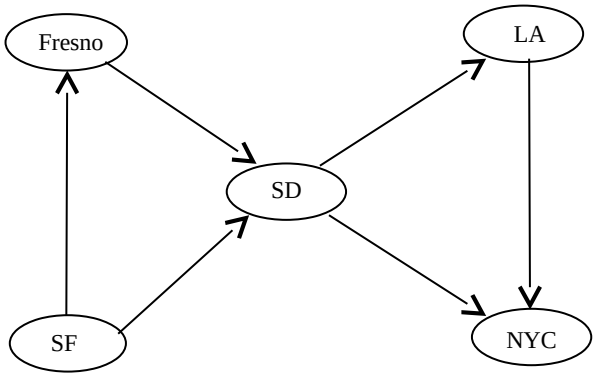
Sample Run 2: Assume that the user typed the following lines

```
5
Fresno
LA
SD
SF
NYC
6
SD LA
SD NYC
LA NYC
SF Fresno
SF SD
Fresno SD
SF
2
```

This is the correct output.

```
Fresno
LA
NYC
SD
SF
```

This is the graph for the input data

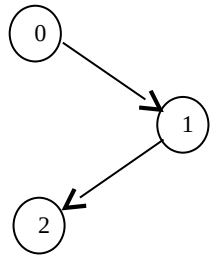


7. Write a C++ (or Java) program called **hw4_3.cpp (**or **hw4_3.java)** that implements the ***Depth-First Search (DFS) algorithm*** using a stack and a mark array as you learned in the class.

Input format: This is a sample input from a user.

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

The first line (= 3 in the example) indicates that there are three vertices in the graph. For the homework, you can assume that the first vertex starts from the number 0. The second line (= 2 in the example) represents the number of edges, and following two lines are the edge information. This is the graph with the input information.



Sample Run 0: Assume that the user typed the following lines

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

This is the correct output. Your program should display the mark array of DFS. For the problem, you can **assume that the starting vertex is always 0**. And also, you can **assume that the graph is connected**.

```
Mark[0]:1
Mark[1]:2
Mark[2]:3
```

Sample Run 1: Assume that the user typed the following lines

```
5
6
0 1
0 2
0 3
1 3
2 3
3 4
```

This is the correct output.

```
Mark[0]:1
Mark[1]:2
Mark[2]:5
Mark[3]:3
Mark[4]:4
```

Sample Run 2: Assume that the user typed the following lines

```
5
6
0 1
0 2
0 3
1 4
2 3
3 4
```

This is the correct output.

```
Mark[0]:1
Mark[1]:2
Mark[2]:4
Mark[3]:5
Mark[4]:3
```