COSC 336   **Test 2**

Instructions: Time 120 minutes. All the questions have equal weight. If you use an algorithm that was discussed in class you can refer to it by its name and no further description is needed

Question 1. In an application we deal with undirected graphs G with vertices *{1, 2, ... , n}* and with each vertex having degree 4 (so each vertex has 4 neighbors).

(a) How much is *m* (the total number of edges) as a function of *n* ?

m =

(b) A graph G as above is weighted with positive weights, and we want to find the shortest path from node 1 to all the other nodes. Taking into account the relation between n and m from part (a), give the running times for this type of graphs of the following algorithms as a function of n (so without m), and indicate which one is the fastest.

(a) running time of Dijkstra's algorithm with the MIN HEAP implementation:

(b) running time of Dijkstra's algorithm with the array implementation:

(c) running time of Bellman-Ford algorithm:

The fastest algorithm from the above options (a), (b), (c) is:

Question 2. The input is an array *a[1], ... , a[n]* of size *n* (with *n* being an odd integer) that contains numbers in arbitrary order (not sorted), and we want to determine if the *median* is larger than 2 * *min* or not. Recall that the *median* is the *(n/2)+1*-th smallest value (example: the median of 2,4,100,1,3 is 3) and *min* is the smallest value. Write the pseudo-code of an algorithm that solves this task. If you use an algorithm that we covered in class, you can just invoke that algorithm by its name and not include pseudo-code for it. Your algorithm can be randomized and its expected running time should be *O(n)* (so the algorithm will not sort the sequence).

```
boolean median_vs_min (a[1..n]) {\\\returns true if median > 2* min, and false
otherwise
```

Question 3. The input is an array *a[1], ... a[n]* of size *n* contains natural numbers between *1* and *2n* (so *a[i]* ∈ {1, ..., 2n}, for every *i)*. Present in pseudocode an algorithm that determines if all the numbers in the array are different, or not (i.e., there are duplicates). Your algorithm should run in time *O(n)*.

```
boolean all_different (a[1..n]) {
// returns true if all the values in the array are different, false otherwise
```

Question 4. In the following hash table, we insert elements using hashing with open addressing with quadratic probing. Elements have as keys integer numbers, and the hash function for the i-th probe is given by *hash(x, i) = (x + i²)* mod *11*. We insert, in the given order, the elements with the following keys: 1, 12, 23, 34, 45. Indicate, where these elements will be placed in the table.

Index	value
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

Question 5.

Figure 1: Graph for Questions 5
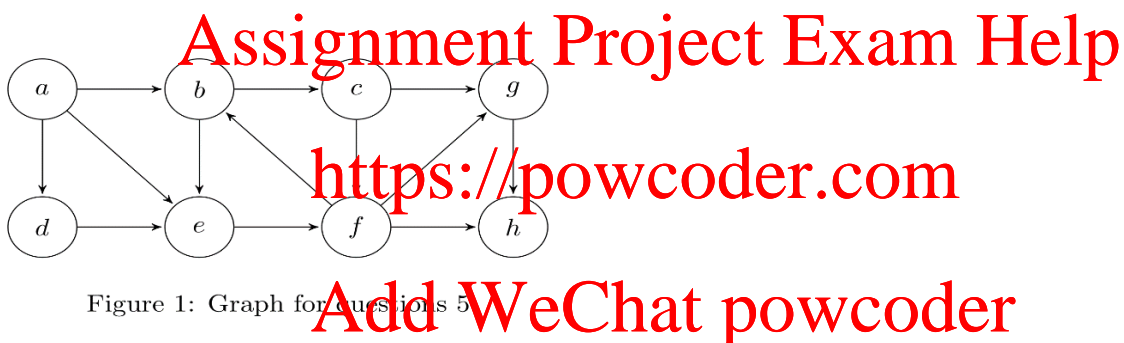
(a) Do a DFS traversal of the graph in Figure 1 starting at node a$ Whenever you have a choice between several nodes, use the alphabetical order. List the nodes in the order in which they are visited. (Note: I have written the first two nodes, you need to add the other ones.)

a,   b,

(b) Do now a DFS with timing of the same graph G starting again at node a

(b.1) What are the discovery and finish times of node a

a.discovery =

a. finish =
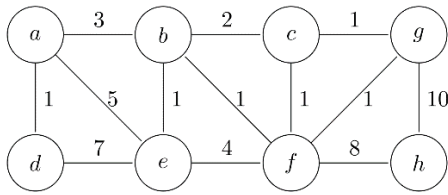
(b.2) List all the back edges of G.

Figure 2: Graph for questions 6 and 7.

Question 6. (a) Construct a minimum spanning tree of the graph in Figure 2 using Kruskal algorithm. Whenever you have a choice between several edges, use the alphabetical order. List the edges in the order in which they are introduced in the tree. (Note: I have written the first edge, you need to add the other ones.)

(a, d)

(b) Construct a minimum spanning tree of the graph in Figure 2 using Prim's algorithm starting at node a. Whenever you have a choice between several nodes, use the alphabetical order. List the edges in the order in which they are introduced in the tree. (Note: I have written the first edge, you need to add the other ones.)

(a, d),

Question 7. (a) In the graph G from Figure 2, consider the cut $S = \{a, d\}$, $V - S = \{b, c, e, f, g, h\}$. What are the edges crossing this cut?

(b) Show that the edge (a, b) appears in every minimum spanning tree (MST) of the graph G. Hint: You can use the basic property of min spanning trees which says that if we consider an arbitrary cut, the smallest crossing edge (assuming there are no ties) must belong to every MST.

(b) Show that any minimum spanning tree of the graph in Figure 2 must contain the edge (b,f). Hint: use the sane basic property as in part (b).

Question 8. Let G be a connected graph representing a computer network. The nodes represent computers, and an edge (u,v) represents two computers linked directly by a cable. You want to remove a cable *(u,v)* (in other words an edge in the graph G), but you need first to check that this removal does not disconnect the network

Present in pseudo-code an *O(n+m)* algorithm that checks if an edge *(u, v)* would disconnect a graph if it is deleted (such an edge is called a *bridge*.) In your pseudo-code, the nodes are 1 , …, n, and the graph is given by the n-by-n adjacency matrix A. You can invoke by its name any algorithm we discussed, without writing the code for it.

```
is_bridge(A,u,v) {
\\returns true if (u, v) is a bridge in the graph with  ix A; false if it \\ is
not
```

Question 9. (a) You are the system administrator of a computer network. Some pairs of computers are linked with a cable, and some pairs are not. You have a map with all the computers, all the cables, and their lengths in meters. In other words, you are given the adjacency matrix with weights of the graph that represents the network. You want to designate one computer as a *hub,* which can send data to all the other computers. Data going from the *hub* to another computer can pass through other computers. The regulations require that data from the hub to any computer should not circulate on a path longer than 500 meters.

Present in pseudocode an algorithm that prints the list with all the computers that can be designated as *hubs*. If you use an algorithm covered in class, indicate it by its name (you don't have to give pseudo-code for it).

```
void list_hubs (L[1..n, 1..n]) {
\\ L is the adjacency matrix with weight is the length of the cable from i to j.
\\ Prints out all vertices that can serve as hubs.
```
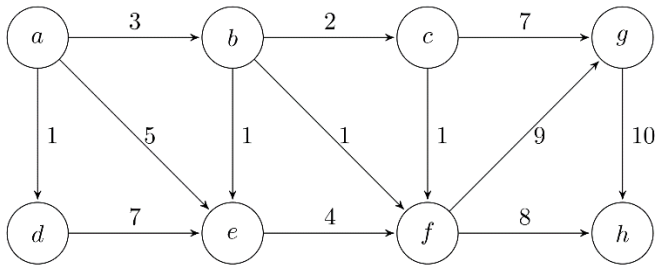
(b) Indicate the running time of your algorithm as a function of n and m, where n and m have the standard meaning in a graph.

Question 10. The graph below represents a network and the capacities are the number written on edges. The source is node *a*, and the target is node *h*.

(a) Show a flow of size 7 units going from the source *a* to the target *h*. (Write on each edge of the graph, next to the capacity, how many units of flow go through that edge.)

(b) Consider the cut *(L, R)*, where *L = {a }* and *R = {b ,d ,e, c, f ,g ,h}* Indicate the edges crossing this cut and find the capacity of this cut.

(c) Joe claims that the Max Flow Min Cut Theorem implies that there is a flow from *a* to *h* of size at least the capacity of the cut from part (b). Explain to Joe why he misunderstood the theorem.

(d) Show using the Max Flow Min Cut Theorem that there is no flow with size larger than 7.