GPA1 IN PDF

Reserved Cells

Ex. 1	
Ex. 2	
Ex. 3	
Ex. 4	
Ex. 5	
Ex. 6	
Tot.	

Algorithms and Programming

31 January 2019

Part I: Theory

Register Number _____ Family Name _____ First Name _____

Course: ○ 01*OGDLP* 10 credit ○ 02*OGDLM* 12 credit

No books or notes are allowed. Solve exercises directly within the reserved space. Additional sheets are accepted only when strictly necessary. Available time: 60 minutes.

1. (2.0 **points**)
Write a function implementing the algorithm of insertion sort and state its asymptotic complexity.
After that, given the following sequence of integers stored in an array:

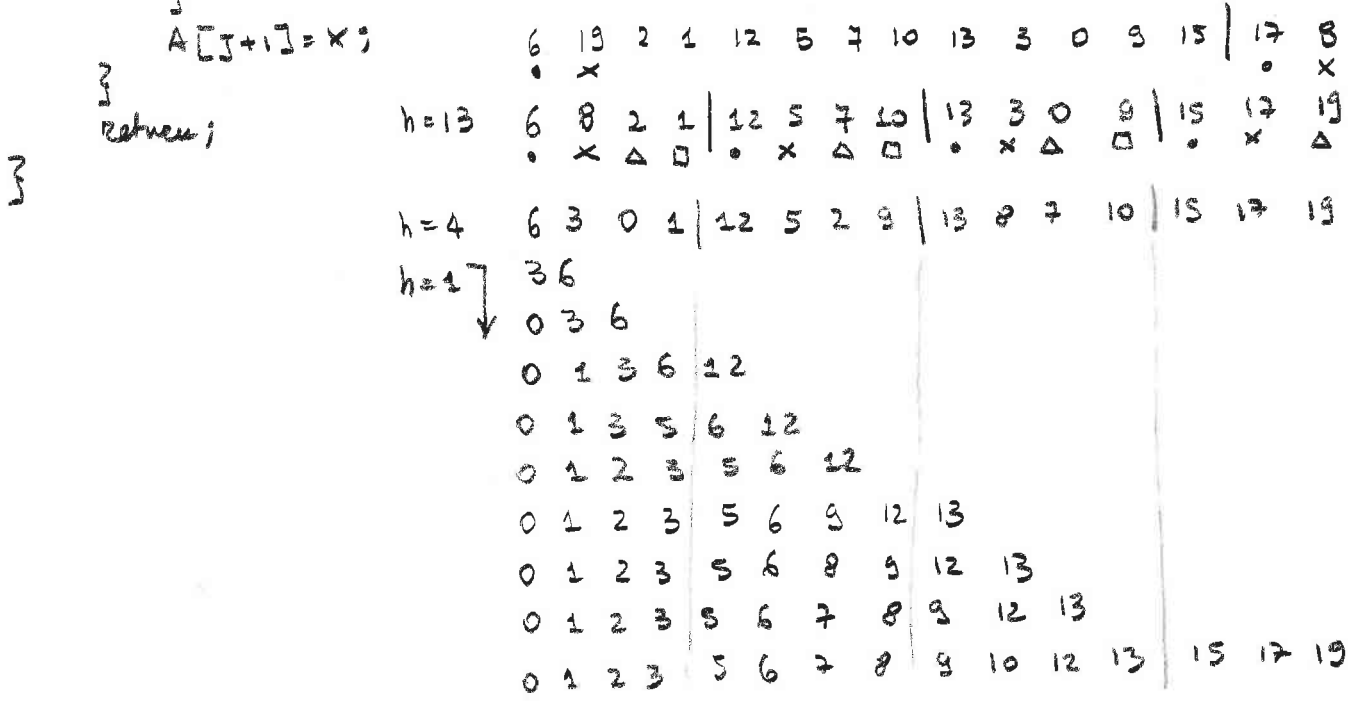
6 19 2 1 12 5 7 10 13 3 0 9 15 17 8

sort it in ascending order with shell sort. Use the Knuth's sequence $h = 3 \cdot h + 1$ (1, 4, 13, etc.). Show all relevant intermediate steps.

```
void InsertionSort(int A[], int n){
    int i, J, x;
    for(i=1; i<n; i++){
        x = A[i];
        J = i-1;
        while(J>=0 && x<A[J]){
            A[J+1] = A[J];
            J--;
        }
        A[J+1] = x;
    }
    return;
}
```

$T(n) = \Theta(n^2)$ worst case

$T(n) = O(n)$ best case



2. **(2.0 points)**
Write a function implementing a visit of a n-ary tree (i.e., a tree in which each node has at most n children). State a possible implementation for its nodes, considering strings as keys.

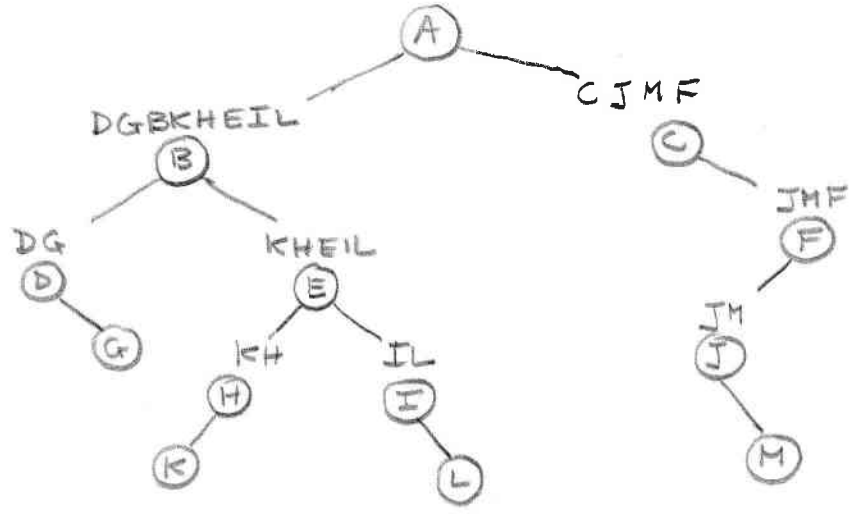
After that, consider a binary tree with 13 nodes. Its visits return the following sequences:

pre-order:	*A*	*B*	*D*	*G*	*E*	*H*	*K*	*I*	*L*	*C*	*F*	*J*	*M*
in-order:	*D*	*G*	*B*	*K*	*H*	*E*	*I*	*L*	*A*	*C*	*J*	*M*	*F*
post-order:	*G*	*D*	*K*	*H*	*L*	*I*	*E*	*B*	*M*	*J*	*F*	*C*	*A*

Draw the original binary tree.

```
typedef struct node_s node_t;
struct node_s {
    char *key;
    int degree;
    node_t *children;
};
```

```
void visit(node_t *root) {
    int i;
    if(root == NULL) {
        return;
    }
    for(i=0; i<root->degree; i++) {
        visit(root->children[i]);
    }
    printf("%s\n", root->key);
    return;
}
```



3. **(2.0 point)**
Given the sequence of keys $INFINITYWAR$, where each character is identified by its index in the English alphabet ($A = 1, \ldots, Z = 26$), draw the final configuration of an initially empty hash table of size 23 where insertion of the previous sequence character by character occurs.

Assume open addressing with double hashing ($h_1(k) = k\%23$, $h_2(k) = 1 + k\%97$). Show all relevant intermediate steps.

Which is the load factor of the hash-table after the previous sequence has been inserted?

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

$$h(k) = [h_1(k) + i \cdot h_2(k)] \% M$$
$$= [(k \% 23) + i(1 + k \% 97)] \% 23$$

		$h(k)$
I_1	9	9
N_1	14	14
F	6	6
I_2	9	9 ✗
N_2	14	14 ✗
I_3	9	9 ✗
T	20	20
Y	25	2
W	23	0
A	1	1
R	18	18

$(9 + (1+9)) \% 23 = 19 \% 23 = 19$ ✓

$(14 + (1+14)) \% 23 = 29 \% 23 = 6$ ✗ $(14 + 2(1+14)) \% 23 = 44 \% 23 = 21$ ✓

$(9 + (1+9)) \% 23 = 19 \% 23 = 19$ ✗

$(9 + 2(1+9)) \% 23 = 29 \% 23 = 6$ ✗

$(9 + 3(1+9)) \% 23 = 39 \% 23 = 16$ ✓

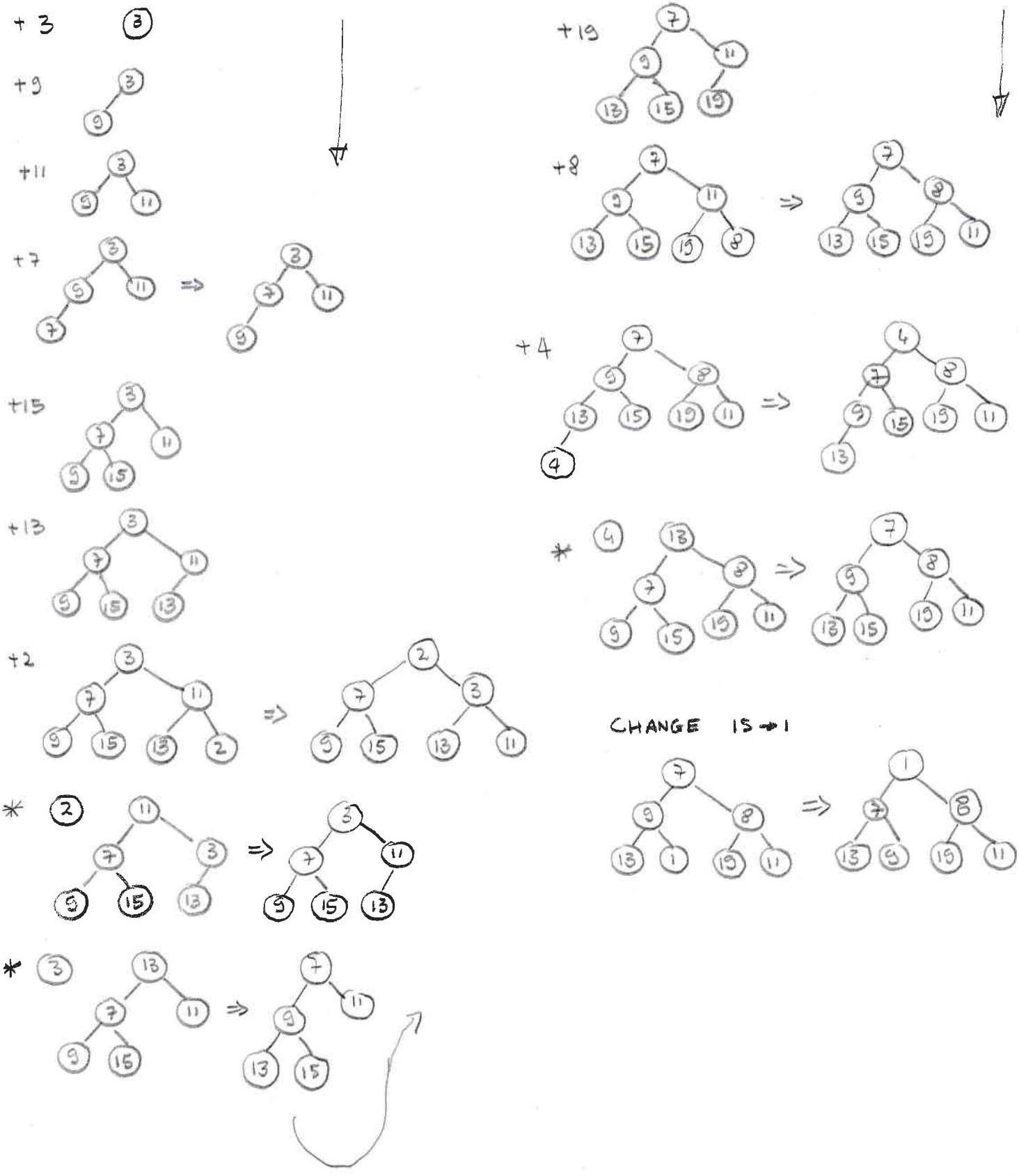
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
W	A	Y				F		I_1						N_1	I_3	R	I_2	T	N_2			

Load Factor $= \alpha = \frac{11}{23} \simeq 0.48$

4. (2.0 **points**)
Suppose to have an initially empty priority queue implemented with a minimum heap. Given the following sequence of integers and $\star$ character:

3 9 11 7 15 13 2 $\star$ $\star$ 19 8 4 $\star$

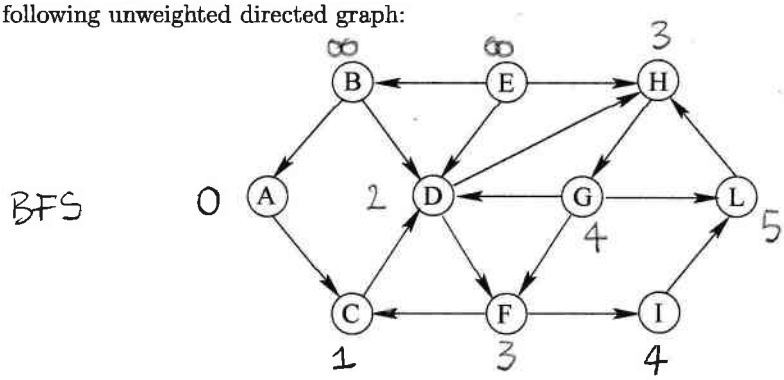
where each integer corresponds to an insertion into the priority queue and character $\star$ corresponds to an extraction, show the priority queue after each operation and return the sequence of values extracted. At the end, change the priority of 15 into 1 and draw the corresponding priority queue.



5. (2.0 **points**)

10 credit course (01*OGDLP*)

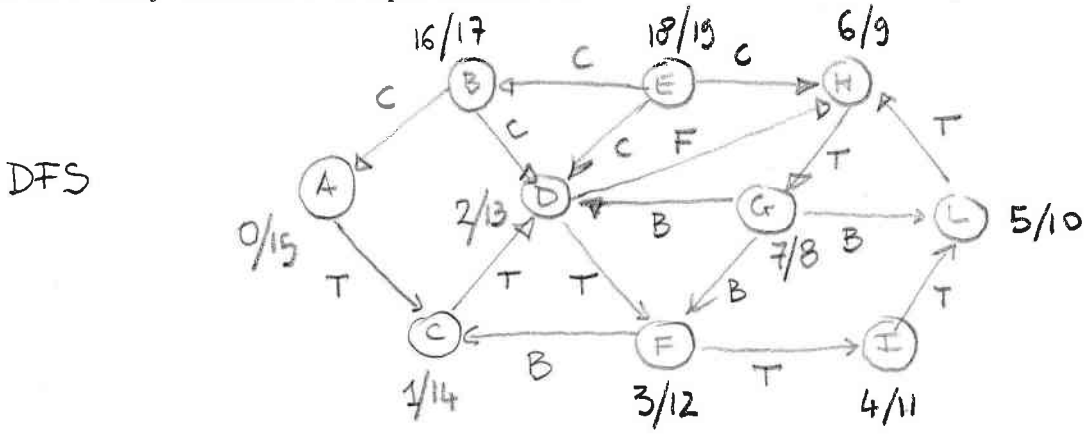
Given the following unweighted directed graph:

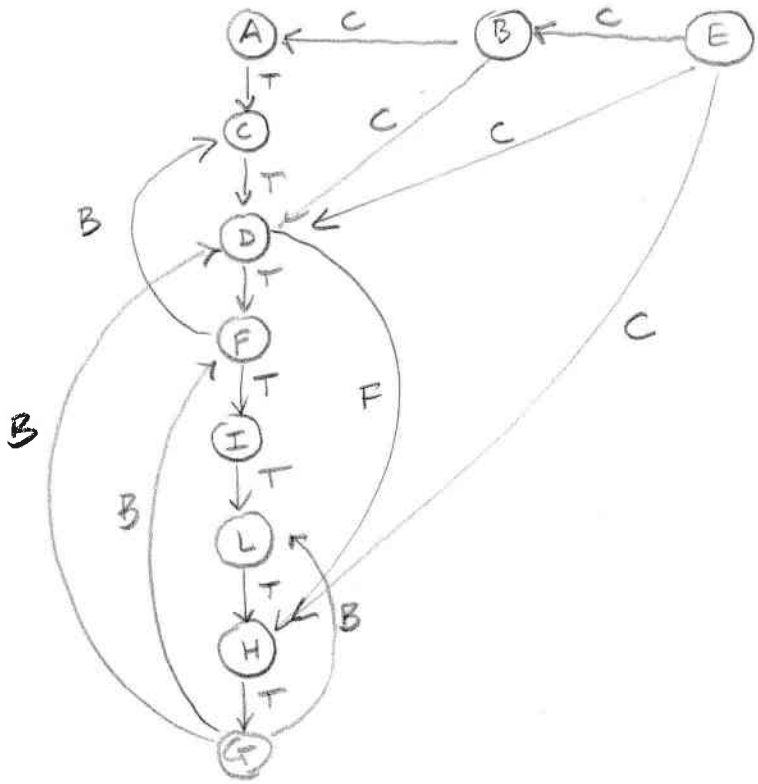


0 A
1 C
2 D
3 F H
4 G I
5 L

- Visit it in breadth-first starting from node A.
- Visit it in depth-first starting at node A. Label nodes with discovery and end-processing times in the format $time_1/time_2$.
- Redraw it labeling each edge as T (tree), B (back), F (forward), C (cross), starting at node A.

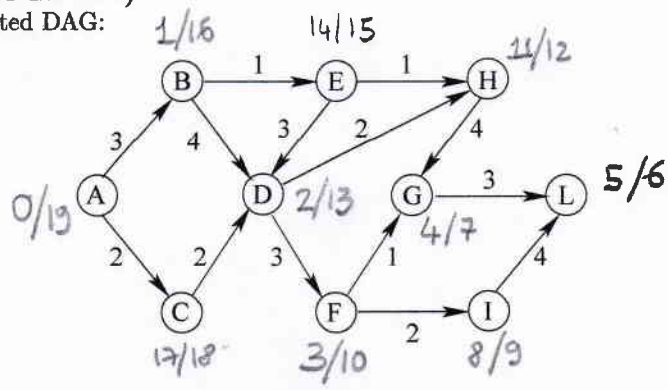
Whenever necessary consider nodes in alphabetical order.





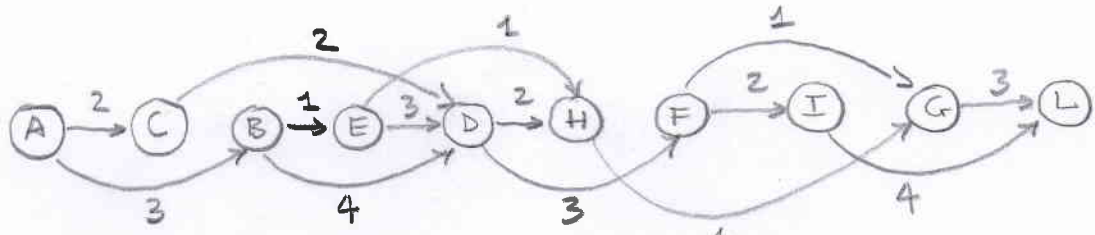
12 credit course (02*OGDLM*)

Given the following weighted DAG:

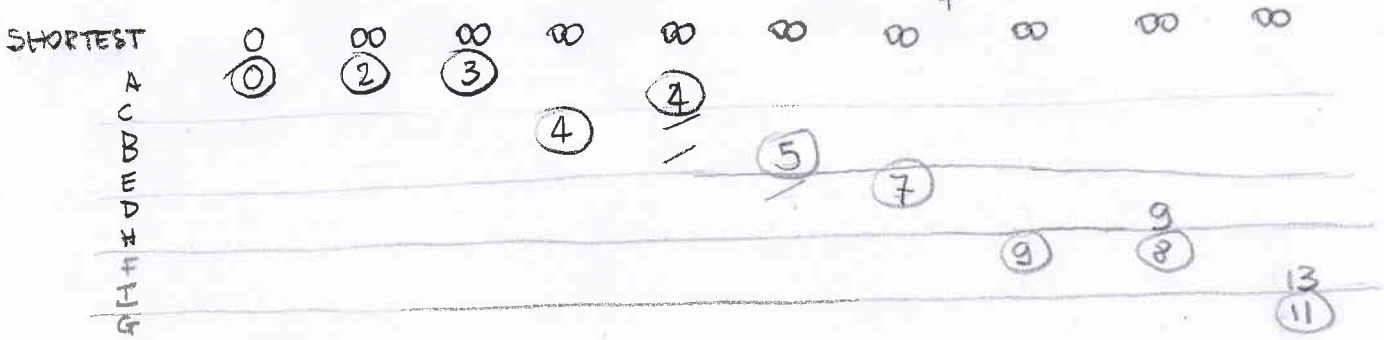


Find all shortest and all longest paths from vertex A, using the simplified algorithm for directed acyclic graphs. If necessary, consider nodes in alphabetical order. Show all relevant intermediate steps.

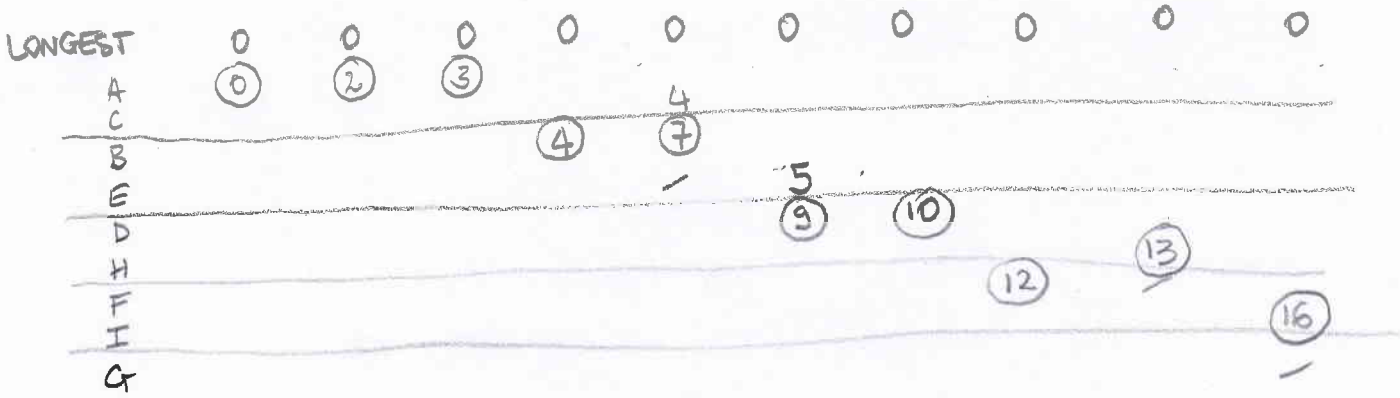
TOPOLOGICAL SORT



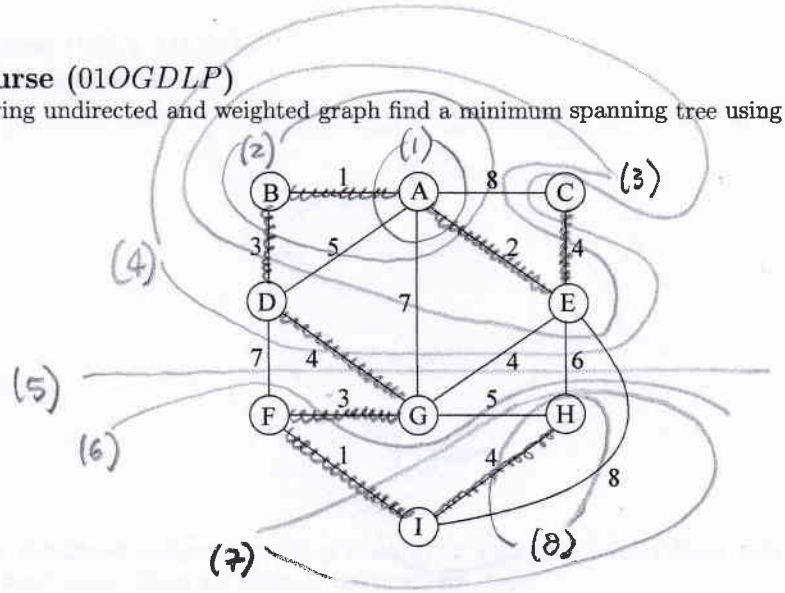
SHORTEST



LONGEST



6. (2.0 **points**)
10 credit course (01*OGDLP*)
Given the following undirected and weighted graph find a minimum spanning tree using **Prim's** algorithm starting from vertex *A*.



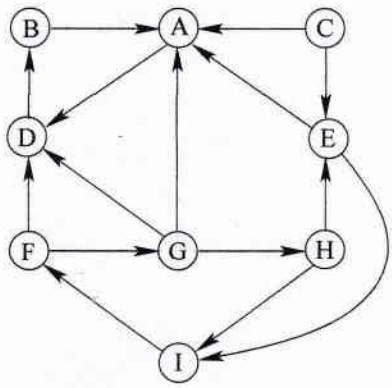
Draw the tree and return the minimum weight as a result. Show all relevant intermediate steps and all cuts generated.

Define and explain what a "light edge" and a "safe edge" are.

AB	1
AE	2
BD	3
CE	4
DG	4
FG	3
FI	1
HI	4
	22

- AN EDGE IS A "LIGHT EDGE" IF ITS WEIGHT IS MINIMUM AMONG THE EDGES CROSSING THE CUT
- A "SAFE EDGE" IS A LIGHT EDGE THAT CAN BE ADDED TO A SUBSET OF THE MINIMUM SPANNING TREE (MST) OR WHICH CONNECTS TWO CONNECTED COMPONENTS OF THE MST TO MAKE IT CONVERGES TOWARD THE (FINAL) MST

12 **credit course** (02*OGDLM*)
Given the following directed graph:



Find its strongly connected components using Kosaraju's algorithm. Start from node A. If necessary, consider nodes in alphabetical order. Show all relevant intermediate steps.