Reserved Cells

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

Algorithms and Programming

26 June 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)**
Evaluate analytically the worst case time complexity of the following algorithm:

```
void f (int A[], int n) {
  int i, j, temp;

  for (i=0; i<n-1; i++) {
    for (j=0; j<n-i-1; j++) {
      if (A[j] > A[j+1]) {
        temp = A[j];
        A[j] = A[j+1];
        A[j+1] = temp;
      }
    }
  }
  return;
}
```

$1+(n-1-0+1)+(n-1-0)$

$\sum_{i=0}^{n-2}\left[1+(n-i-1-0+1)+(n-i-1-0)\right]$

Worst Case

$\sum_{i=0}^{n-2}(n-i-1)$

Exchange (Bubble) sort

$T(n) = 1+(n)+(n-1)+\sum_{i=0}^{n-2}\left[1+(n-i)+(n-i-1)\right]+4\sum_{i=0}^{n-2}(n-i-1)$

$2n-2i$

$= n+2\sum_{i=0}^{n-2}n-2\sum_{i=0}^{n-2}i+4\left[\sum_{i=0}^{n-2}n-\sum_{i=0}^{n-2}i-\sum_{i=0}^{n-2}1\right]$

$= n+6\sum_{i=0}^{n-2}n-6\sum_{i=0}^{n-2}i-4\sum_{i=0}^{n-2}1$

$= n+6\cdot(n-1)n-6\frac{(n-2)(n-1)}{2}-4\cdot(n-1)$

$= 3n^2+n-2$

$= O(n^2)$ = quadratic

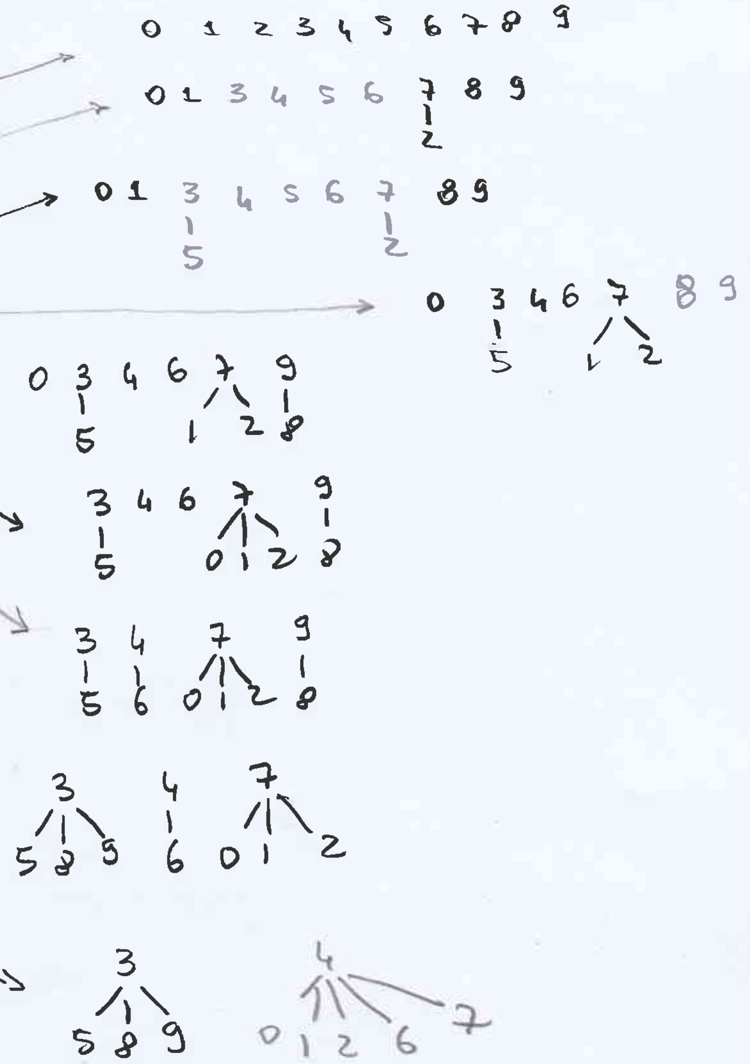
2. (2.0 **points**)
Given the following sequence of pairs, where the relation i-j means that node i is adjacent to node j:

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

apply an on-line connectivity algorithm with quick-find showing at each step the content of the array and the corresponding tree representation. Node names are integers in the range from 0 to 9.

Describe the differences (in terms of logic and complexity) among quick-find, quick-union, and weighted quick-union.

	0	1	2	3	4	5	6	7	8	9
	0	1	2	3	4	5	6	7	8	9
2-7	0	1	7	3	4	5	6	7	8	9
5-3	0	1	7	3	4	3	6	7	8	9
1-7	0	7	7	3	4	3	6	7	8	9
8-9	0	7	7	3	4	3	6	7	9	9
0-1	7	7	7	3	4	3	6	7	9	9
6-4	7	7	7	3	4	3	4	7	9	9
8-3	7	7	7	3	4	3	4	7	3	3
1-2	no change									
5-9	no change									
2-6	4	4	4	3	4	3	4	4	3	3
4-0	no change									
3-8	no change									



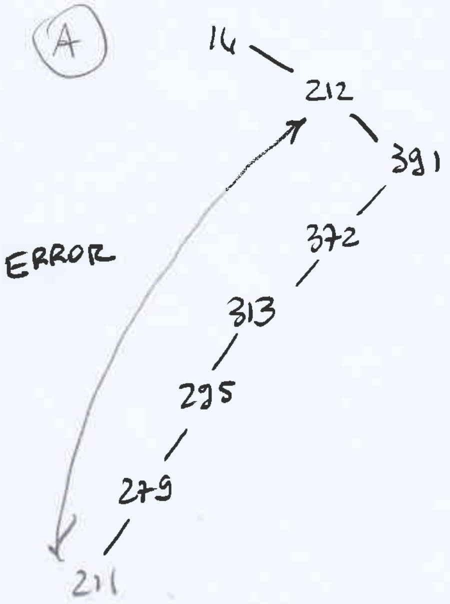
Quick-Find : ∀p-q if (id[p] != id[q]) substitute all id[p] with id[q]
#op = #pairs · array size

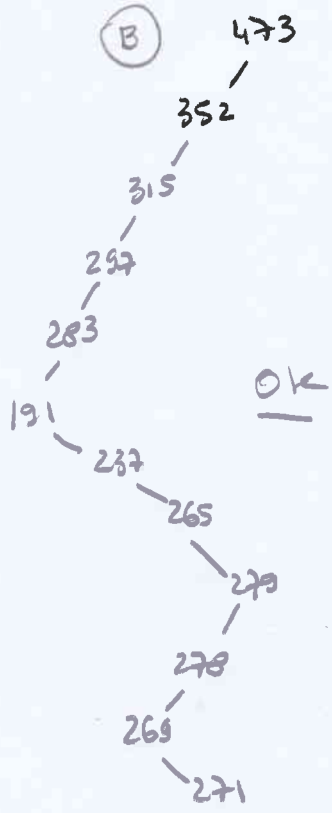
Quick-Union : i and j connected IFF (id[i])* = (id[j])*
#op = #pairs · chain length

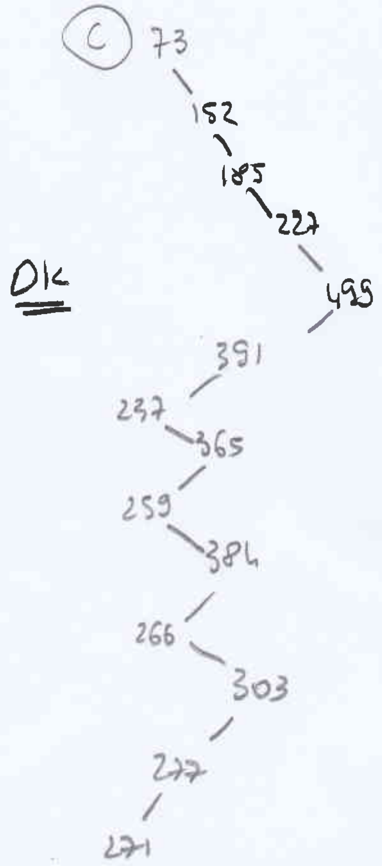
Weighted Quick-Union : like quick-union but smaller tree goes below larger one
#op = #pairs · chain length but chain length grows logaritmically

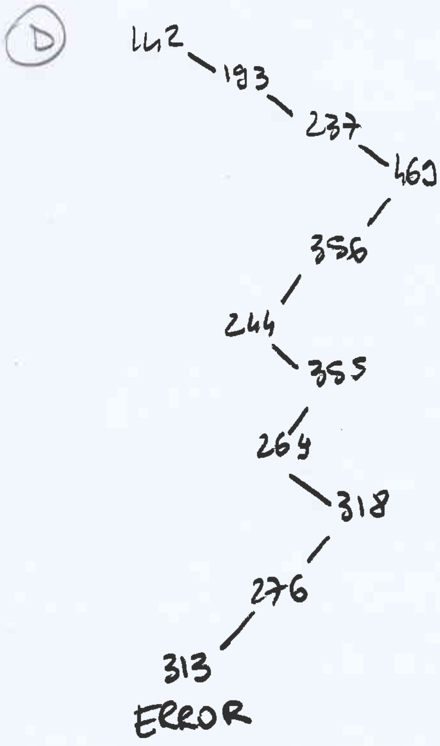
3. **(1.5 point)**
A correct BST contains integer keys in the range 1-500. The user searches for key 271. Among these sequences, which are the ones that cannot occur? Why?

Sequence A :	14	212	391	372	313	295	279	211	237	265	275	271		
Sequence B :	373	352	315	297	283	191	237	265	279	278	269	271		
Sequence C :	73	152	189	227	499	391	237	365	259	348	266	303	277	271
Sequence D :	142	193	237	469	356	244	355	269	318	276	313	287	271	









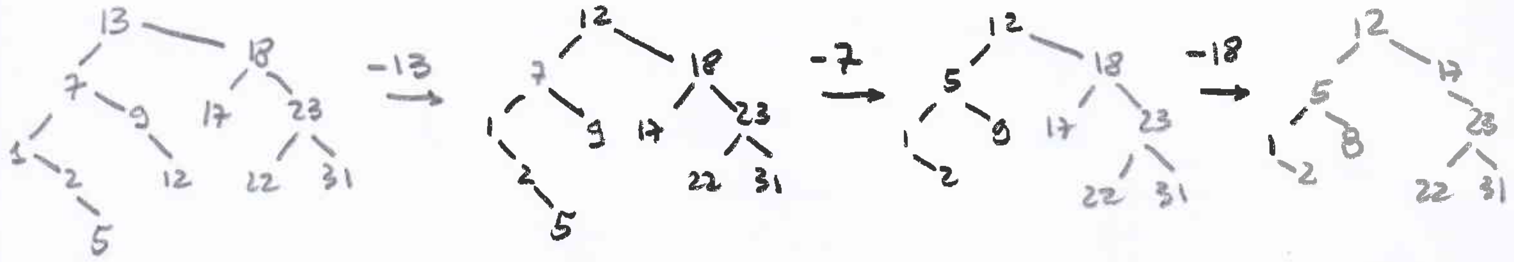
4. (1.5 **points**)
10 **credit course** (01*OGDLP*)
Given an initially empty BST, perform the following sequence of operations on the BST leaves:

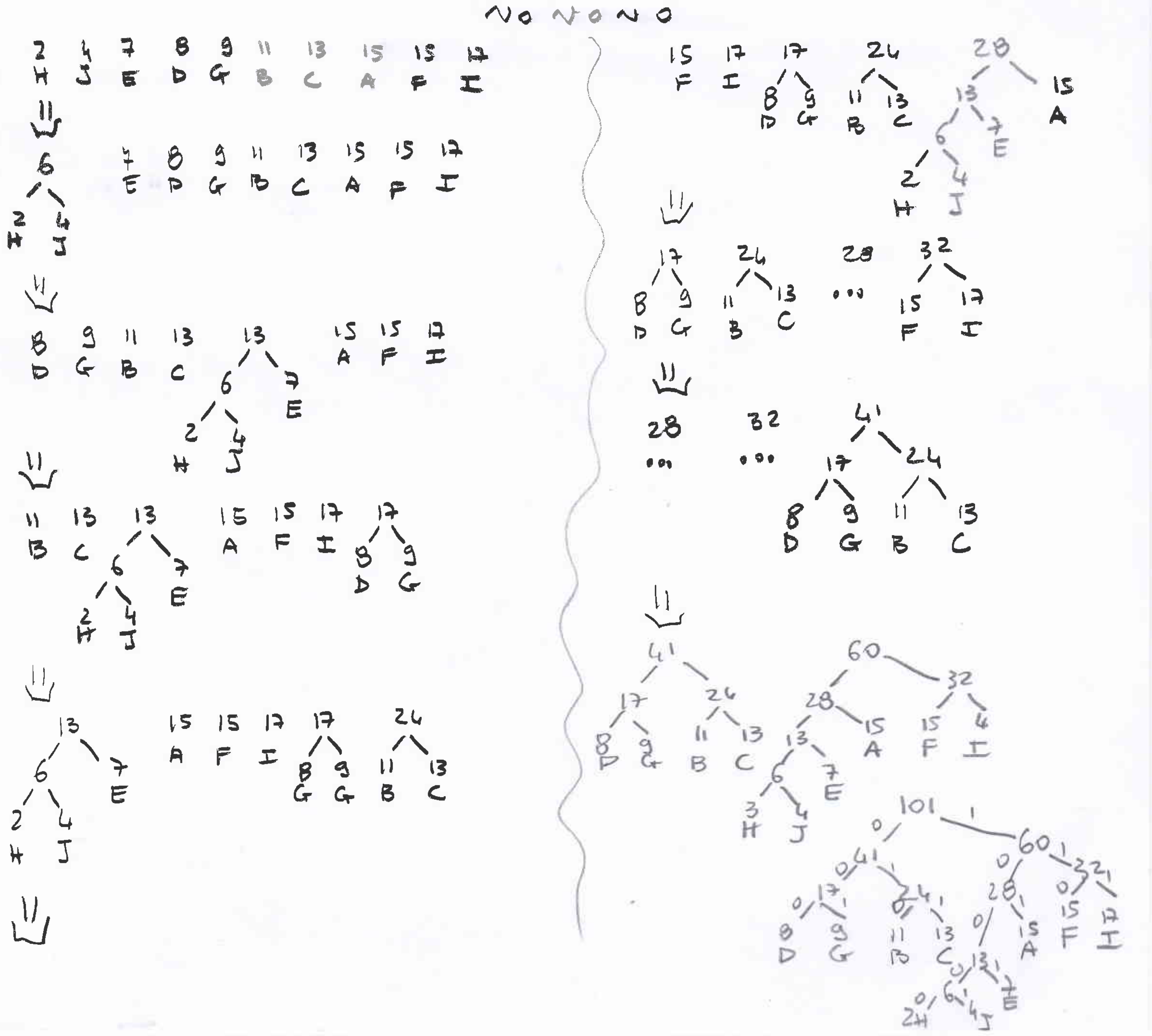
$+13 \quad +7 \quad +1 \quad +18 \quad +23 \quad +9 \quad +12 \quad +31 \quad +22 \quad +17 \quad +2 \quad +5 \quad -13 \quad -7 \quad -18$

where each + indicates a leaf insertion and each − a node extraction.

12 **credit course** (02*OGDLM*)
Using a greedy algorithm find an optimal Huffman code for the following symbols with the specified frequencies:

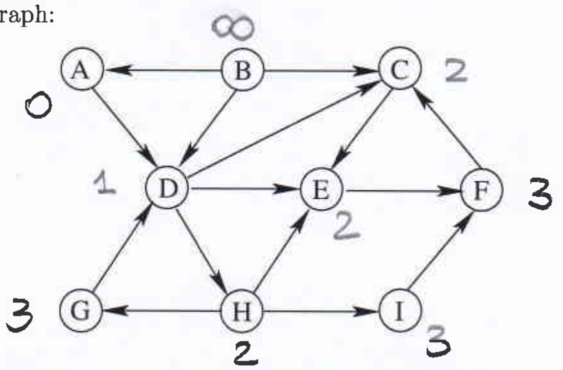
A:15 *B*:11 *C*:13 *D*:8 *E*:7 *F*:15 *G*:9 *H*:2 *I*:17 *J*:4
101 010 011 000 1001 110 001 1000 111 1001





5. (3.0 **points**)
Given the following directed graph:

BFS



A	0
B	∞
C	2
D	1
E	2
F	3
G	3
H	2
I	3

0	A
1	D
2	C E H
3	F G I

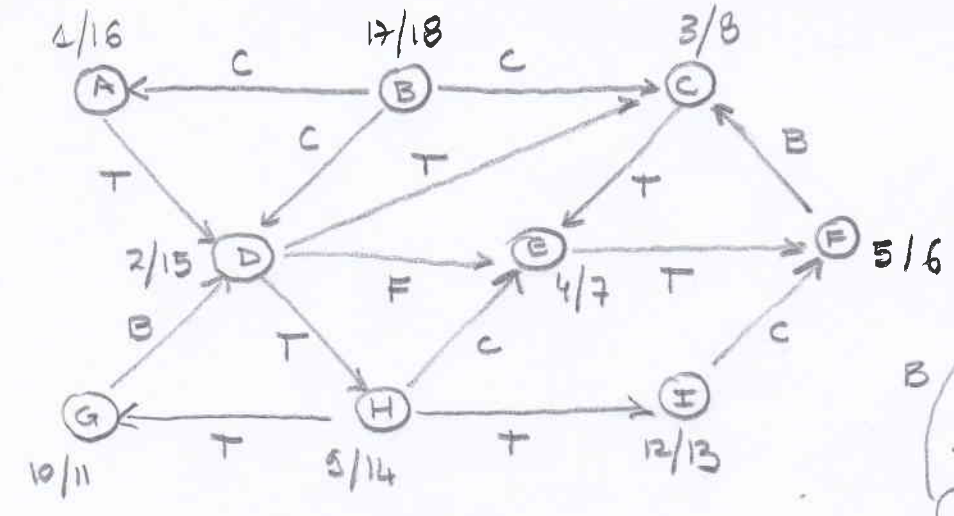
- **ONLY for the** 10 **credit course** (01*OGDLP*): Represent it as an adjacency list and as an adjacency matrix.
- **FOR EVERYBODY**: Visit it in breadth-first starting at node A.
- **FOR EVERYBODY**: Visit it in depth-first starting at node A. Label nodes with discovery and end-processing times in the format $time_1/time_2$. Redraw it labeling each edge as T (tree), B (back), F (forward), C (cross).
- **ONLY for the** 12 **credit course** (01*OGDLP*): Represent the reverse graph and find all strongly connected components.

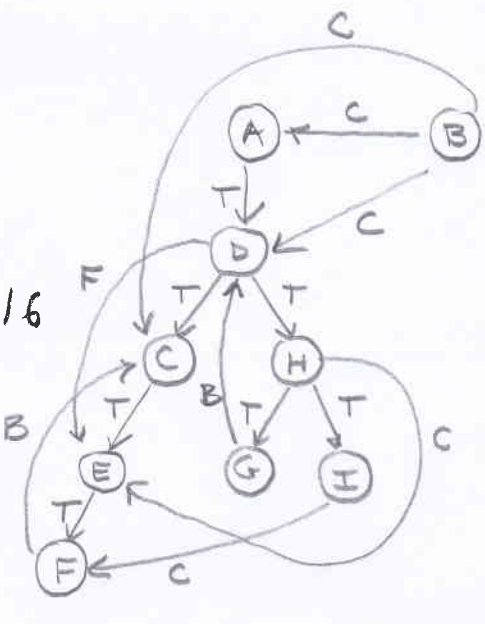
Whenever necessary consider nodes in alphabetical order.

	A	B	C	D	E	F	G	H	I
A				1					
B	1		1	1					
C					1				
D			1		1			1	
E						1			
F			1						
G				1					
H					1		1		1
I						1			

A → D
↓
B → A → D → C
↓
C → E
↓
D → C → E → H
↓
E → F
↓
F → C
↓
G → D
↓
H → E → G → I
↓
I → F

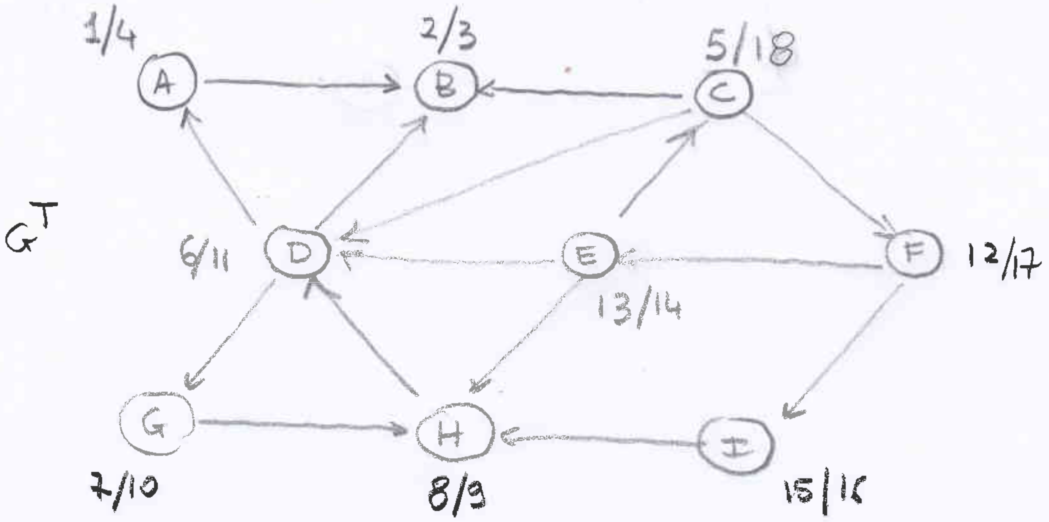
DFS





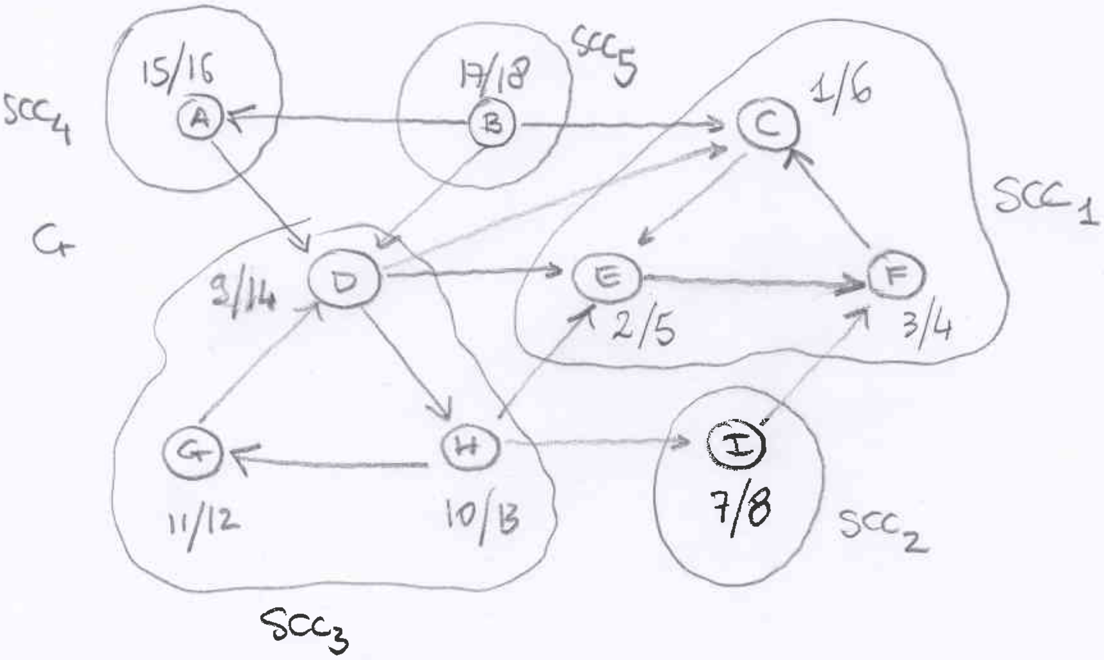
SCC

DFS on G^T, then on G in decreasing endprocessing times



DECREASING ENDPROCESSING TIMES

C
F
I
E
D
G
H
A
B

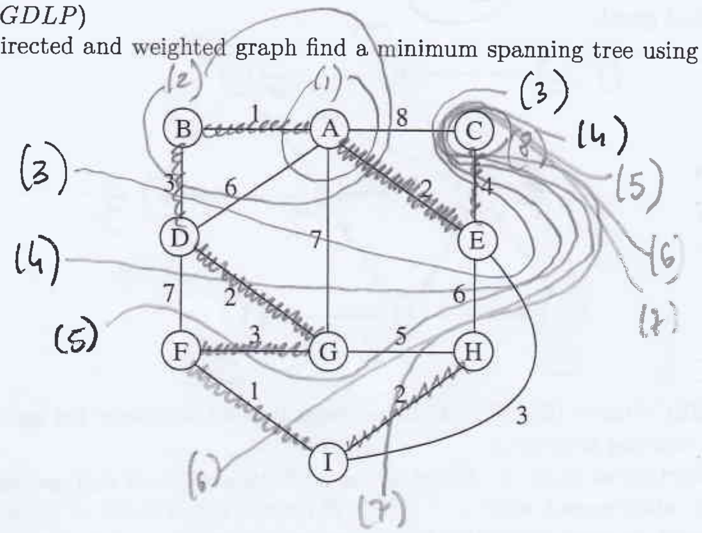


SCC_1	C E F
SCC_2	I
SCC_3	D G H
SCC_4	A
SCC_5	B

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*.



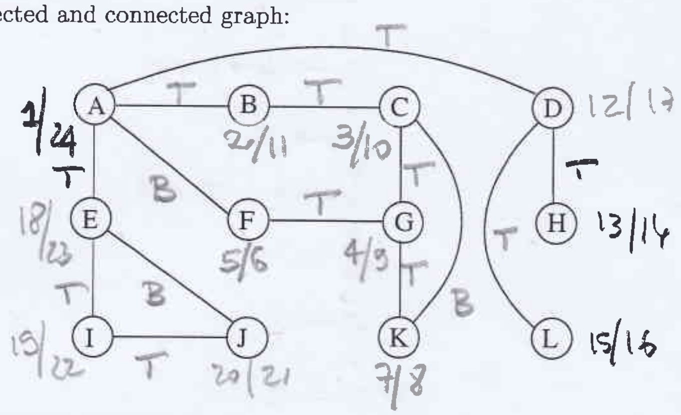
1 AB
1 AE
3 BD
2 DG
3 FG
1 FI
2 IH
4 CE

18

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

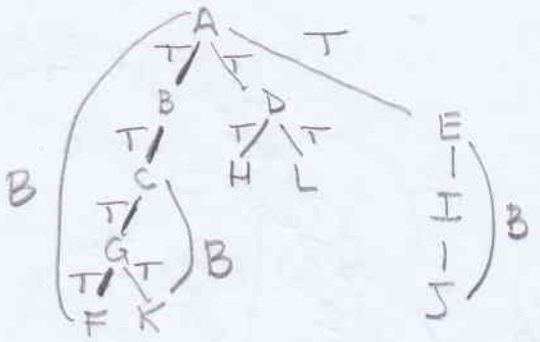
12 **credit course** (02*OGDLM*)

Given the following undirected and connected graph:



find all bridges and all articulation points.

If necessary, consider nodes in alphabetical order. Show all relevant intermediate steps.



BRIDGES : AD DH DL ~~AE~~

ARTICULATION POINTS : A D E