Programme Code: TU856 & TU858
Module Code: CMPU2001
CRN: 22395 & 26458

TECHNOLOGICAL UNIVERSITY DUBLIN

Grangegorman

BSc. (Honours) in Computer Science
BSc. (Honours) in Computer Science International

Year 2

SEMESTER 2 EXAMINATIONS 2023/24

CMPU2001 - Algorithms & Data Structures

Internal Examiners:
Mr. Richard Lawlor
Dr. Paul Doyle

External Examiner:
Ms. Pamela O'Brien

Instructions To Candidates:

Attempt three out of four questions. All questions carry equal marks. One complementary mark for paper.

Exam Duration: 2 hours

Special Instructions /Handouts/ Materials Required: none

1. (a) What are the differences, if any, between a Heap array and a sorted array?

What are the main advantages of a binary Heap compared to both an unsorted array and a sorted array from the point of view of **insert()** and **remove()** operations?

(6 marks)

(b) Selection sort and Heap sort are similar in that they are both examples of *hard-split easy-join*. Comment on this.

(6 marks)

(c) Draw the following heap array as a two-dimensional binary tree data structure:

k	0	1	2	3	4	5	6	7	8	9	10	11
a[k]		12	11	8	10	7	5	1	9			

Also, assuming another array hPos[] is used to store the position of each key in the heap, show the contents of hPos[] for this heap.

(6 marks)

(d) By using tree and array diagrams, illustrate the effect of inserting a node whose key is **13** into the heap in the array of part (c). You can ignore effects on hPos[].

(8 marks)

(e) Write in pseudocode a recursive version, maxHeapify(int k), of the siftDown(int k) Heap operation.

(7 marks)

2. **(a)** Using diagrams, show the detailed workings for the first two passes of the outer loop of a bubble sort on the following array.

6	3	1	9	8	2	4	7	0	5

Use your resulting array diagram to point out in your own words a significant flaw in bubble sort.

(6 marks)

(b) What is the basic idea behind comb sort that makes it an improvement on bubble sort. Apply comb sort to the array in part (a) until it becomes bubble sort..

(8 marks)

(c) Show the first two partitionings that occur when quick sort is applied to the array in part (a).

(7 marks)

(d) Provide a simple example which shows quick sort at its worst. What is the complexity of quick sort at its worst?

(5 marks)

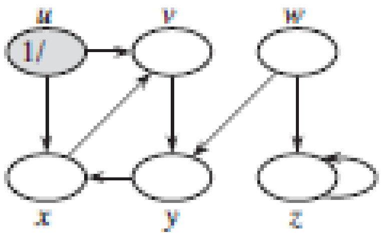
(e) Write down the complexity of heap sort and outline how it is arrived at. In your own words, contrast it with that of bubble sort and quick sort.

(7 marks)

3 (a) Given the following Depth First Search algorithm and graph, show step by step how it traverses the graph by computing u.d, u.f, u.color and u.π for each vertex u.

```
DFS(G)
1  for each vertex $u \in G.V$
2      $u.color$ = WHITE
3      $u.\pi$ = NIL
4  $time = 0$
5  for each vertex $u \in G.V$
6      if $u.color$ == WHITE
7          DFS-VISIT(G, u)
```

```
DFS-VISIT(G, u)
 1  $time = time + 1$            // white vertex u has just been discovered
 2  $u.d = time$
 3  $u.color$ = GRAY
 4  for each $v \in G.Adj[u]$     // explore edge (u, v)
 5      if $v.color$ == WHITE
 6          $v.\pi = u$
 7          DFS-VISIT(G, v)
 8  $u.color$ = BLACK            // blacken u; it is finished
 9  $time = time + 1$
10  $u.f = time$
```

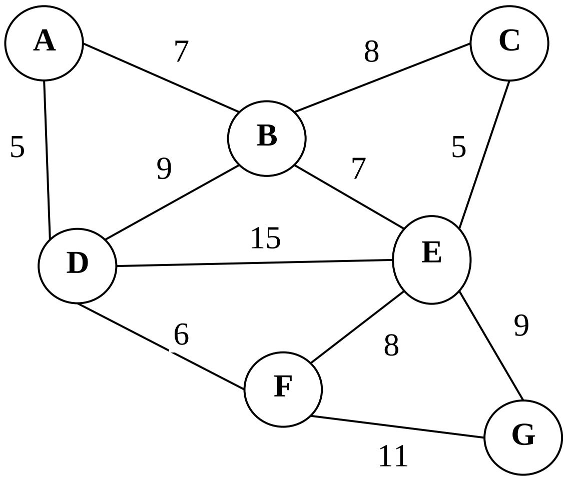


(10 marks)

(b) Using the pseudocode in part (a) as a basis, derive the complexity of *DFS(G)* assuming that G is a single connected graph.

(6 marks)

(c) Write down in pseudocode Dijkstra's *shortest path tree* algorithm assuming an adjacency-lists representation of the graph and then show its detailed working on the following graph.



(17 marks)

4. (a) Explain in your own words the purpose of UnionFind data structure in Kruskal's algorithm.

(8 marks)

(b) Mention two ways of representing UnionFind sets and give the complexity of the significant operations for one of these representations.

(4 marks)

(c) Given the following array for the tree representation of UnionFind sets:

index	A	B	C	D	E	F	G	H	I	J	K
treeParent[]	C	C	C	B	D	F	D	K	H	H	K

i) Draw the corresponding trees.
ii) What do the operations *set1* = *findSet(G)* and *set2* = *findSet(H)* return? Briefly comment on how you arrived at the answer.
iii) Draw the resulting tree from the operation *union(set1, set2)* where *set1* and *set2* are from part ii).
iv) Suppose you had implemented *findSet()* with tree-compression, show the effect *findSet(E)* would have on the corresponding set-tree. Also show the updated *treeParent[]* array.

(21 marks)