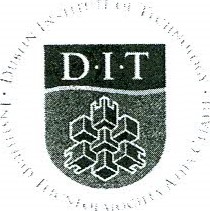
S228/212C, S282/212C



DUBLIN INSTITUTE OF TECHNOLOGY



DT228 BSc. (Honours) Degree in Computer Science

Year 2

DT282 BSc. (Honours) Degree in Computer Science

(International)

Year 2



SUMMER EXAMINATIONS 2016/2017



ALGORITHM DESIGN AND DATA STRUCTURES [CMPU20011

MR. RICHARD LAWLOR

DR. DEIRDRE LILLIS

MR. PAUL COLLINS

FRIDAY 12TH MAY 4.00 P.M. - 6.00 P.M.

Two HOURS

ATTEMPT THREE OUT OF FOUR QUESTIONS.

ALL QUESTIONS CARRY EQUAL MARKS.

ONE COMPLEMENTARY MARK FOR THE PAPER.

PAGE OF

1. (a) Write a simple Java interface to express the services provided by the Abstract Data Type (ADT) Queue. You can assume the queue stores int values.

(5 marks)

***Interface Queue***

***{***

***Public void enQueue(int x);***

***Public int deQueue();***

***Public Boolean isEmpty();***

***}***

* 1. Provide a Java class which implements the Queue interface based on a linked list implementation but showing only data structure, constructor and no other methods.
     1. marks)

***class QueueLinkedList implements Queue***

***{***

***LinkedList head, tail;***

***Int size;***

***Public QueueLinkedList()***

***{***

***Head = tail = null;***

***Size = 0;***

***}***

***}***

* 1. Give the implementation of the method deQueue() for the class in part (b). Explain why you might use or not use a tail reference in your implementation. What is the complexity of deQueue() with and without the use of a tail pointer?

<http://gmetrics.sourceforge.net/gmetrics-CyclomaticComplexityMetric.html>

(9 marks)

Int deQueue()

{

If (!(isEmpty()))

{

Int toReturn = head.data;

Head = head.next;

Size--;

Return toReturn;

}

Return -1;

}

I would use a tail to keep track of the position of the contents of the queue as it would be visually easier to find any part of the queue by starting and the head and working till you get to the tail

The complexity of the either usage is O(2) as you are checking if the queue is empty, then doing another check on if the first is false. That is what is being done for both of them.

* 1. Provide a partial circular buffer implementation of ADT Queue. Show the data structure used and the code for deQueue() only.

(9 marks)

Class circularBuffer Implements Queue

{

Int queue[];

Int head, tail;

Public int deQueue()

{

If (!(isEmpty()))

{

Int toReturn = Queue[head++];

Return toReturn;

}

Return -1;

}

}

* 1. Write down a simple equation for a Stack in terms of pop() and push() that expresses the last-in-first-out (LIFO) behaviour of the Stack.
     1. marks)

The simple equation for expressing an stack in terms of using pop() and push() is the following: pop() ==> element = stack[size--] && push() ==> stack[++size] = element

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



(4 marks)

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

* The biggest number moves to the end

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

* Every time we move a number to the end we move

The endpoint back by one. This allows us to

Ignore the end as we know it is always the biggest

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

* 1. Explain what is meant by Tortoises and Hares with reference to bubble sort. Provide a simple example to illustrate.

(4 marks)

The hare is the value on the left side of the array, known as the first element. Normally this is a big value so it is going to take multiple steps to get to its desired location in a bubble sort. The tortoises are normally at the end and can only take one step at a time to get to the beginning of an array. As a result, the efficiency of the bubble sort slows down. The idea is to remove all the tortoises so the efficiency can go at hare speed.

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

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

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

The brown is the hares and the grey is the tortoises

* 1. Write an adaptation of bubble sort in pseudocode which helps with the problem of Tortoises and Hares.

(10 marks)

int i, j, temp;

int sorted;

for( i = 1; i < n; ++i) // repeat bubble pass n-1 times

{

sorted = true;

for( j = 0; j < n-i; ++j)

if( a[j] > a[j+1])

{

temp = a[j];

a[j] = a[j+1];

a[j+1] = temp;

sorted = false;

}

if (sorted) break;

* 1. Show how your algorithm from part (c) sorts the above array.
     1. marks)
  2. Using diagrams, show the array in part (a) can be converted to a heap.
     1. marks)
  3. Using diagrams, show how heap sort works on the heapified array from part (d).
     1. marks)

1. (a) Provide a brief explanation on of how Kruskal's MST algorithm works and then write the algorithm in pseudocode.

(8 marks)

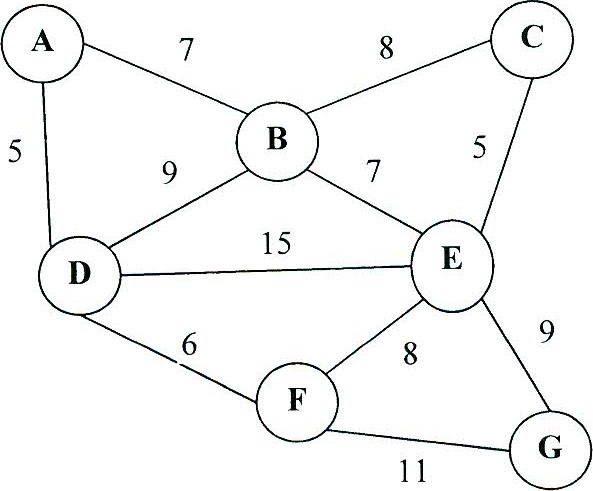
* 1. What representation of a graph is appropriate for implementing Kruskal's algorithm? What is the complexity of creating this data structure and removing the edges as

required from it?

* + 1. marks)
  1. Explain the Union-Find data structure and what it is used for in Kruskal's algorithm. Also, with the aid of diagrams, outline a possible implementation of this data structure and give an example showing how its two significant operations work.

(10 marks)

* 1. Illustrate in detail how Kruskal's algorithm computes a MST for the graph below showing the contents of the union-find sets at each stage.



(10 marks)

1. (a) Show how binary search works when searching for 17 in the following array:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | 10 | 12 | 14 | 17 | 21 |

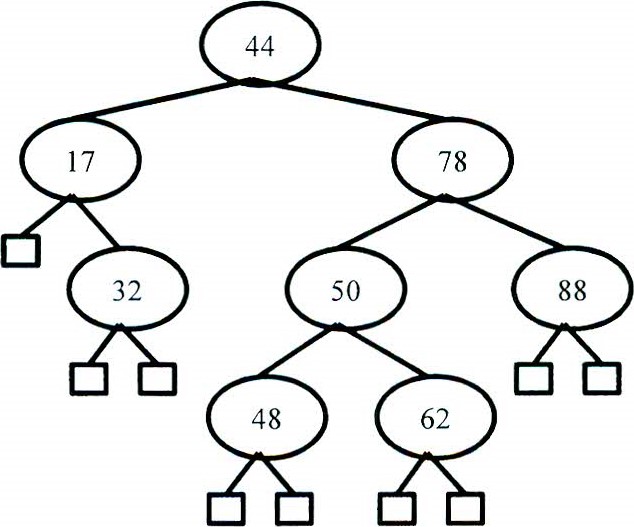
(5 marks)

* 1. What is a binary search tree (BST)? Mention any specific advantage or possible disadvantage. What is the complexity of searching a BST?

(6 marks)

* 1. Write in pseudocode the algorithm for searching a BST.

(6 marks) (d) Given the following binary search tree, show how it would be modified by inserting 54.



* + 1. marks)

(e) What is an AVL-tree? Include in your answer the idea of a rotation.

Show how the tree that results from inserting 54 in part (d) would be rebalanced if it were an AVL-tree.

(l I marks)

Of