CS-M12 2013

Software Concepts and Efficiency

(Attempt 2 questions out of 3)

Question 1: Algorithm and Complexity

Suppose `a` is a given array of length `n`. Consider the following function:

```
int something (int a[n])
{
    int temp = 0;
    int i, j;
    for (i = 0; i < n; i++)
        for (j = i + 1; j < n; j++)
            if ( abs(a[j] - a[i]) > temp)
                temp = abs (a[j] - a[i]);
    return temp;
}

// abs() - returns the absolute value of a number.
```

a) What does the function perform? Explain briefly what the function does in general terms, not how it executes the task.

[5 Marks]

b) Analyze the time complexity of the function [5 marks] in Big-O notation [5 marks]?

[10 Marks]

c) Write a snippet of code or pseudo-code which performs the same task but which is ***asymptotically faster*** (not a constant factor) in time complexity [7 marks]. Analyze the complexity of your algorithm in Big-O notation [3 marks].

[10 Marks]

Question 2: Linked List and Binary Search Tree

a) Explain succinctly what a singly linked list is, in plain English.

You should describe
1) the internal representation [3 marks], and
2) how to tell if it is empty [2 marks].

[5 Marks]

b) Explain succinctly what a binary search tree is -
the definition [3 marks], and
its internal representation [2 marks].

[5 Marks]

c) Given a pointer to the root of a binary search tree, write a snippet of code or pseudo-code that creates a sorted (***in descending order***) singly linked list from the tree, ***using recursion***. You should not use any library functions but can assume that the value at a tree node can be obtained using the `getValue()` method and the children of a node using the `getLeft()` and `getRight()` methods. `null` is returned when there is no corresponding child. You can assume the linked list functions `addFirst()`, `set/getNext()`, `addLast()`, `split()`, and `concate()` are available. You should clearly specify the general and base case in your algorithm.

You may use the following template:

```
LinkList Tree2ListDsc() {
      return toListDsc (root);
}
LinkList toListDsc(Node node) {
      // ... your code here ...
      // ...
      // ...
}
```

[8 marks]

d) Given a pointer to the `head` of a singly linked list, write a snippet of code or pseudo-code that reverses the list. Your algorithm should not create a new list but merely rearrange the existing one. You should not use any library functions or additional data structure but can assume that successor node can be obtained using the `getNext()` method (`null` is returned when there is no successor) and a new successor node can be set using the `setNext()` method. Tips: it does not require recursion. You may use the following template:

```
public void reverse() {
      // You code here...
      // ...
      // ...
}
```

[7 marks]

Question 3: Graph and traversal

a) Explain succinctly what a directed graph is -
its internal structure [3 marks], and
what does "directed" mean [2 marks].

[5 marks]

b) Explain succinctly what a breadth first search (BFS) graph traversal algorithm is, in plain English -
1) what is the main purpose of a traversal algorithm [1 marks],
2) a high-level description of how BFS works (2 sentences) [3 mark], and
3) what problem can BFS solve whilst depth first search (DFS) cannot [1 mark]?

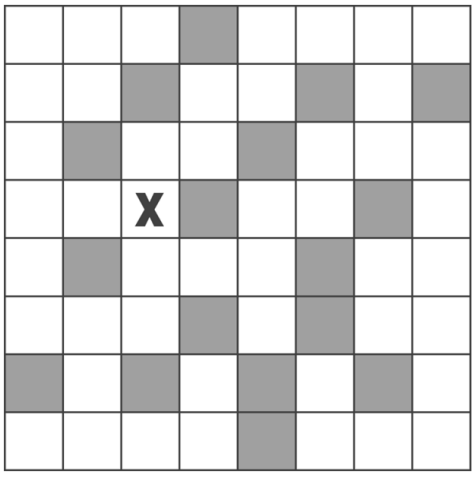
[5 marks]

Dungeon

A token (marked 'X' in the diagram) is in a maze. You may move the token around according to the following rule.

RULE: in each move the token may travel to ***any*** white square ***vertically*** or to ***one*** adjacent white square ***horizontally***, but it cannot pass over or stop on a shaded square.

For example, from its starting position the token could travel either one square left, one square up, one square down or two squares down in a single move. To reach any other square would require more than one move.



c) What is the minimum number of moves that you need to ensure that the token can reach any white square from its starting position, in the above maze?

[3 marks]

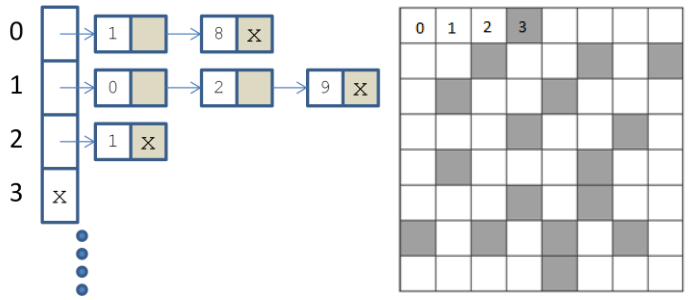
(A) 8 (B) 9 (C) 10 (D) 11 (E) 12

The above problem can be transformed into a graph representation and solved using a graph traversal algorithm. Suppose you are given the following:

`map[1:n][1:n]` - a 2D array where the entries contain `0` (a shaded square) or `1` (a white square).

`g` - a Graph `g` contains `n×n` number of vertices. Vertices are number 0 to n^2-1. Each vertex corresponds to an entry in the 2D array. There is a directed edge if a "white space" vertex is adjacent to another "white space" vertex.

The graph, in the above Dungeon example (`n=8`), looks like this in the adjacency-list representation:



For example, there is a directed edge between vertex 0 (`map[0][0]`) and vertex 1 (`map[0][1]`) and 8 (`map[1][0]`).

d) Write a snippet of code or pseudo-code to add ***extra*** directed edges to the graph so that all vertices within one move are connected to each other using the ***RULE*** mentioned above. You can assume the following are provided:

`g.addDirectedEdge(`*`vertex_id1`*`, `*`vertex_id2`*`)`
- add a directed edge so that vertices with *`vertex_id1`* and *`vertex_id2`* are connected.

You may use the following template.

```
public void AddEdges(Graph g, int[][] map) {
      // Your code here...
      // ...
      // ...
}
```

[8 marks]

And then discuss, *in plain English*, how you can compute the minimum number of moves for a token to reach any white square using the constructed graph.

[4 marks]

---- END ----