UNIVERSITY OF LONDON

GOLDSMITHS COLLEGE

Department of Computing

B. Sc. Examination 2013

IS52017B
Algorithms and Complexity Theory

Duration: 2 hours 15 minutes

Date and time:

There are five questions in this paper. You should answer no more than THREE questions. Full marks will be awarded for complete answers to a total of THREE questions. Each question carries 25 marks. The marks for each part of a question are indicated at the end of the part in [.] *brackets.*

There are 75 marks available on this paper.

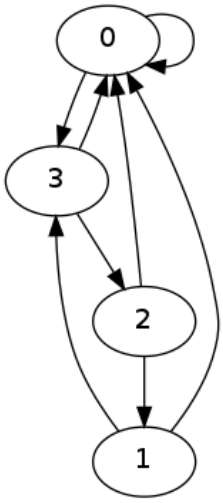
THIS PAPER MUST NOT BE REMOVED
FROM THE EXAMINATION ROOM

Question 1

(a) Consider the following adjacency matrix:

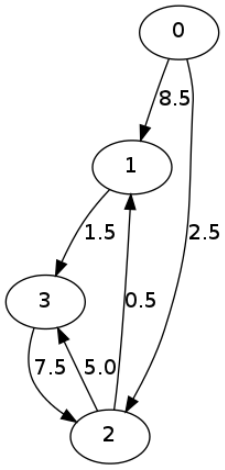
```
int [] [] a = {{1,0,0,1},{1,0,0,1},{1,1,0,0},{1,0,1,0}}
```

Draw a picture of the graph assuming the vertices are labelled 0-3.



[4]

(b) Consider the following weighted directed graph



Which is the length of the shortest path (in terms of distance) from 0 to 3?

4.5

[3]

(c) If a directed graph has many vertices and not many edges, it is more efficient in terms of space to represent it as a `treeMap <Integer, HashSet <Integer>>`,

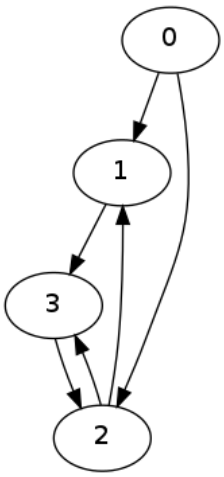
mapping each vertex to the set of its immediate neighbours. Consider the following method

```
int f(treeMap <Integer, HashSet <Integer>> g, int i)
{
   int k=0;
   for (Integer v: g.keySet())
   {
     if (g.get(v).contains(i)) k++;
   }
  return k;
}
```

What do you think the purpose of this method is?

It returns the in degree of vert i in graph g [4]

(d) Consider the following directed graph:



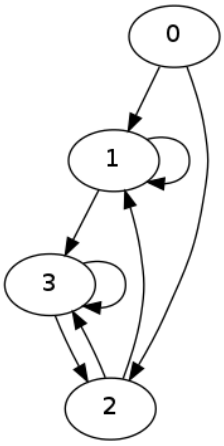
Which one of the following adjacency matrices represents it?

i. `int [] [] a = {{0,1,1,0},{0,0,0,1},{0,1,0,0},{1,0,1,0}}`
ii. `int [] [] a = {{0,1,1,0},{1,0,0,1},{1,0,0,0},{1,0,1,0}}`
iii. `int [] [] a = {{0,1,1,0},{1,0,0,1},{1,0,0,0},{0,0,1,0}}`
iv. None of the above.

[2]

None of the above

(e) Consider the following directed graph

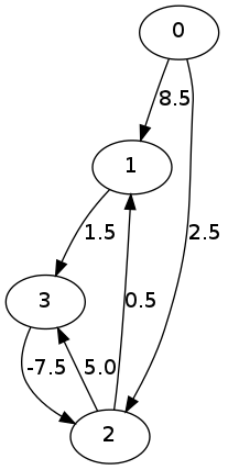


How many paths are there from 0 to 2?

[3]

Infinitely many

(f) Consider the following weighted directed graph



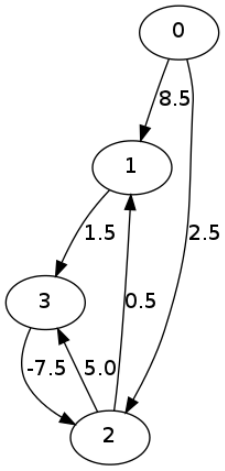
Is there a shortest path (in terms of distance) from 0 to 1? Explain your answer. [4]

No. Every time we go from 3 to 2 the more negative the shortest path becomes.

(g) Consider the following Java program

```
import java.util.HashSet;
class it1
{

    static HashSet <Integer> f (double [][] graph, int i)
    {
      HashSet <Integer> K= new HashSet <Integer>();
      for (int j=0;j<graph.length;j++)
      if (graph[i][j]>0) K.add(j);
      return K;
    }
}
```

If `a` was the adjacency matrix representing the graph:



What would the output of executing the statement

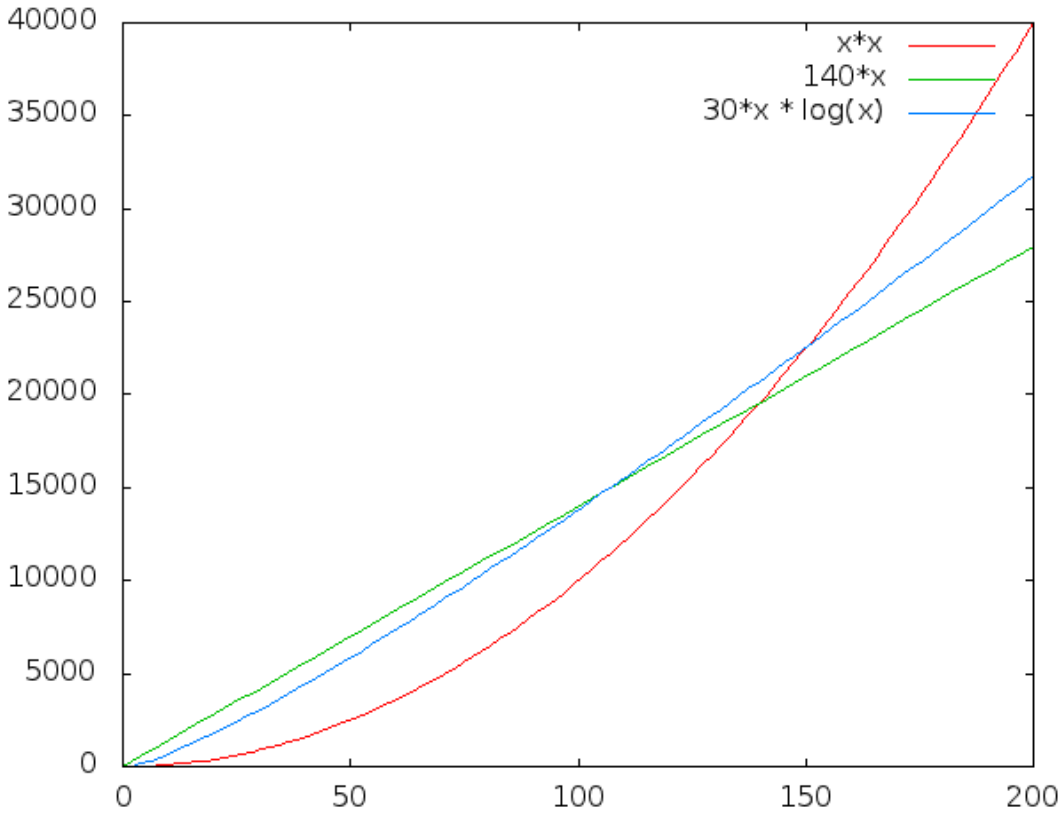
```
System.out.println(f(a,3))?
```

Explain your answer. [5]

1. It is the out degree of 3 in the graph.

Question 2

(a) Consider the following plot:



What is the equation of the middle curve in the plot. i.e. the one which has the value of just over 30000 when x has the value 200?

*30*x*log(x)* [2]

(b) Which of the following functions will produce the largest values (asymptotically) as N gets bigger:

i. $f(N) = 30 * N$
ii. $f(N) = N^2$
iii. $f(N) = N * log(N)$

[2]

$f(N) = N^2$

(c) What is the worst-case time complexity of *insertion sort*? [2]

quadratic

(d) If it take 5 nanoseconds to sort 10 elements using insertion sort, roughly how long will it take to sort 20 elements in the worst case? [2]

20

(e) What is the time-complexity of this function in terms of N?

```
int f(int N)
{
  if (N<2) return 1;

  return f(N-1)+f(N-2);
}
```

[3]

Exponential

(f) Here is a method for computing x^n:

```
static int powerA(int x, int n)
{
     int total=0;
     while (n>0) {total=total*x;n--;}
     return total;
}
```

What is its time complexity? [3]

Linear

(g) Here is a method for computing x^n:

```
static int powerB(int x, int n)
{
     if (n==0) return 1;
     int k=n/2;
     int z=powerB(x,k);
     int r=z*z;
     if (n%2==0) return r;
     return x*r;

}
```

What is its time complexity? [3]

Log

(h) What is the time-complexity of this function in terms of N?

```
int f(int N)
{
  int total=0;
  for (int i=0;i<N;i++)
    for (int j=0;j<N;j++)
      total=total+i+j;
  return total;
}
```

Explain your answer. [4]

Quadratic loop within a loop

(i) What is the time-complexity of this function in terms of N?

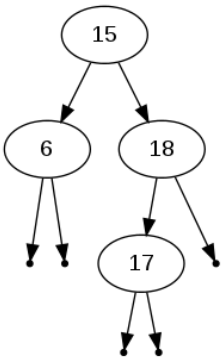
```
int f(int N)
{
  int total=0;
  for (int i=0;i<N;i++)
    for (int j=0;j<N;j++)
      for (int k=0;k<N;k++)
      total=total+i+j+k;
  return total;
}
```

Explain your answer. [4]

Cubic loop within a loop within a loop

Question 3

(a) What is the depth of the following Binary Tree?



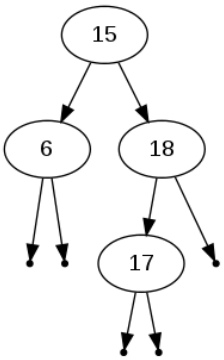
[2]

(b) Consider the following Java classes:

```
public abstract class binaryTree <T>
{
    abstract T root ();

    abstract binaryTree left ();

    abstract binaryTree right ();
}


class emptyTree  <T> extends binaryTree  <T>
{
     T root()
        {throw new IllegalArgumentException
         ("Can't do root of empty binaryTree <T>");
        }

    binaryTree <T> left ()
    {throw new IllegalArgumentException
     ("Can't do left of empty binaryTree <T>");
    }

    binaryTree <T> right ()
    {throw new IllegalArgumentException
     ("Can't do right of empty binaryTree <T>");
    }
}
```

```
class consbinaryTree <T> extends binaryTree <T>
{
    T root;
    binaryTree <T> left,right;

    consbinaryTree (T roo, binaryTree <T> l, binaryTree <T> r)
    {root=roo;left=l;right=r;}

    T root()
    {return root;}

    binaryTree <T> left ()
    {return left;}

    binaryTree <T> right ()
    {return right;}
}
```

What is the expression that generates the binary tree:




```
new consbinarytree(15,
                   new consbinarytree(6,new empty(),new empty())
                   new consbinarytree(18,
                                      new consbinarytree(17,new empty(),new empty())
                                      new empty()
                                      )
                   )
```

[5]

(c) Formally define a binary search tree.

Its either an empty binary tree or a non empty binary tree Where (a) The left and right subtrees are both binary search trees.

and

(b) Every vertex in the left subtree is $\leq$ the root

(c) Every vertex in the right subtree is $\geq$ the root

[5]

(d) Explain what the following method does:

```
int f (binaryTree <Integer> t)
{
   int (t.left().isEmpty()) return t.root();
   else return f(t.left());
}
```

when applied to a binary search tree. [5]

It returns its smallest element.

(e) Consider the following Java method:

```
ArrayList <Integer> flatten (binaryTree <Integer> t)
{
   ArrayList <Integer> m = new ArrayList <Integer> ();
   if (t.isEmpty()) return m;
    ArrayList <Integer> z1= flatten(t.left());
    ArrayList <Integer> z2= flatten(t.right());
    z1.add(t.root());
    z1.addAll(z2());
    return z1;
}
```

If `b` is a binary search tree, `flatten(b)` will produce a list with what properties?

sorted in ascending order [2]

(f) Consider the following Java class:

```
class f
{
 static binaryTree <Integer> insert(int i, binaryTree <Integer> b)
 {
   if (b.isEmpty())
       return new consbinaryTree (i,new emptyTree(),new emptyTree());
   else
```

```
    if (i < b.root())
       return new consbinaryTree (b.root(), insert(i,b.left()), b.right());
    else
       return new consbinaryTree (b.root(), b.left(), insert(i,b.right()));
 }

 static binaryTree <Integer> makeBST(int [ ] a)
 {
   binaryTree <Integer> b = new binaryTree <Integer>();
   for (int i=0;i<a.length;i++) b=insert(a[i],b);
   return n;
 }

 public static void main( String[] argz)
 {
   int [] a = {1,2,3,4,5,6,7,8};
   binaryTree <Integer> t = makeBST(a);
 }
]
```

What will the depth of `t` be after running the program?

8 [4]

(g) Which of the following assignments would result in `makeBST(a)` having depth 3? (See previous question)

i. a={1,2,3,4,5,6,7};

ii. a={4,1,5,3,7,2,6};

iii. a={4,7,3,6,2,1,5};

iv. None of the above.

[2]

iii

Question 4

(a) Given the grammar G defined as

$$S \to a$$
$$S \to aSa$$
$$S \to b$$
$$S \to bSb$$

Which of the following strings is not in $L(G)$?

i. aaa

ii. aab

iii. $aaabbbaaa$

iv. bbb

[2]

(b) Given the grammar G_4 defined as

$$S \to TL$$
$$S \to T + S$$
$$T \to F$$
$$T \to F * T$$
$$F \to n$$

Which of the following strings is not in $L(G_4)$?

i. n

ii. $n + n$

iii. $n * n+$

iv. $n * n * n * n * n * n * n * n$

[2]

(c) Given the grammar G defined as

$$S \to a$$
$$S \to aSa$$
$$S \to b$$
$$S \to bSb$$

Draw the parse tree of $aabaa$.

[5]

(d) Explain with an example what an *ambiguous* grammar is.
2 marks for definition and three for example

[5]

(e) In the following Java Abstract Syntax Tree definition for arithmetic expressions with just one binary operator `plus`, something is missing in the `plus` class:

```
abstract class exp
{
   abstract int evaluate();
}

class num extends exp
{
  int n;

  num(int x)
  {
    n=x;
  }

  public int evaluate()
  {
    return n;
  }
}

class plus extends exp
{
   exp e1,e2;

   plus(exp x, exp y)
   {
     e1=x;
     e2=y;
   }

  public int evaluate()
  { //!something missing here!
  }
}
```

What is missing?

```
return e1.evaluate()+e2.evaluate();
```

[4]

(f) Write a Java expression for constructing the Abstract Syntax Tree corresponding to `1+2+3`?

```
new plus(new (1), new plus(new (2), new (3)))
```

[4]

(g) Consider the following Java code:

```
static String f(String s)
{
  String t=s;
  while (!t.isEmpty() &&
          (t.charAt(0)=='a' || t.charAt(0)=='b')
        )
        t=t.substring(1);
  return t;
}
```

What value is returned by the Java expression `f("a12ab34")`?

```
"aab"
```

[3]

Question 5

(a) In a game of chess there are about 30 legal moves in every position. A typical game of chess lasts about 40 moves. A game of chess is a sequence of moves. Roughly, how many different (40 move) games of chess are there? (Express in the form n^m) [2]

30^{40}

(b) Let $x = 2^y$ Which of the following is true:

i. $y = log_e(x)$
ii. $y = log_2(x)$
iii. $x = log_2(y)$
iv. None of the above.

[2]

ii

(c) Consider the following Java program

```
import java.util.ArrayList;
class it
{
 public static void main(String[] a)
   {
        ArrayList <Integer> K= new ArrayList <Integer>();
        K.add(1);
        K.add(1);
        System.out.println(K.size());
   }

}
```

What does it output? [4]

[1,1]

(d) Consider the following Java program

```
import java.util.HashSet;
class it
{
 public static void main(String[] a)
   {
        HashSet <Integer> K= new HashSet <Integer>();
        K.add(1);
        K.add(1);
        System.out.println(K);
   }

}
```

What does it output? [4]

[1]

(e) Consider the following Java program

```
import java.util.TreeMap;
class it
{
 public static void main(String[] a)
   {
        TreeMap <Integer,Integer> K= new TreeMap <Integer,Integer>();
        K.put(1,2);
        K.put(1,3);
        System.out.println(K.get(1));
   }

}
```

What does it output?

3 [4]

(f) Which is the most suitable representation of a path of a graph in Java?

i. HashSet
ii. ArrayList
iii. Map
iv. ArraySet

Briefly explain your answer. [5]

ii + 3 marks for explanation.

(g) Here is some pseudo-code for a well known algorithm:

```
Set S = {start};
Map <Integer,Double> Q = Map each Vertex to Infinity, except map start -> 0;
Map <Integer, ArrayList <Integer> > paths = Map each vertex to an empty ArrayList
while (Q is not empty)
{
  let v be the key of Q with the smallest value;
  if (v is end) return paths(end);
  let w be the value of v in Q;
  add v to S;
  for (each neighbour u of v that is not in S) do
  {
    let w1 be the the weight of the (v,u) edge + w;
    if w1 < the value of u in Q, then do the following:
                    {
                       update Q so now the value of u is w1
                       update paths(u) to be paths(v) with u stuck on the end
                    }
  }
  remove v from Q;
}
return [];
```

Name the algorithm and state what is it for?

Prim's Algorithm for finding a minimum spanning tree of a graph. [4]