Coláiste na Tríonóide, Baile Átha Cliath
Trinity College Dublin
Ollscoil Átha Cliath | The University of Dublin

Faculty of Engineering, Mathematics and Science

School of Computer Science and Statistics

Integrated Computer Science **Trinity Term 2017**
BA (Mod) Computer Science and Business
BA (Mod) Computer Science and Language
BA Management Science and Information Systems Studies (MSISS)

Year 2 Annual Examinations

CS2010: Algorithms and Data Structures

Tuesday 2nd May 2017 **Sports Centre** **14.00-17.00**

Dr Hugh Gibbons, Dr Vasileios Koutavas

Instructions to Candidates:

- This exam has TWO PARTS.
- Answer TWO of the three questions in PART A.
- Answer TWO of the three questions in PART B.
- Use SEPARATE answer books for each part.
- Each question is worth 25 marks.
- Add brief explanations to any code you write.

Materials permitted for this examination:

- None

PART A

Question 1 [25 marks]

(a) i. Write the main operations in the API of the **Symbol Table** Abstract Data Type (ADT) and briefly describe what they do. *[2 marks]*

ii. Using the asymptotic notation, give the **worst-case running time** of each of these operations in the standard implementation of Symbol Table using a *red-black (balanced) binary search tree.* *[3 marks]*

iii. Discuss the benefits of implementing a Symbol Table using a red-black tree, instead of a hashtable. Also discuss the benefit of implementing a Symbol Table using a hashtable, instead of a red-black tree. *[5 marks]*

(b) The following is the class of the nodes of a linked list.

```
class LNode {
    public int data;
    public LNode next;
    public LNode(int d, LNode nxt) { data = d; next = nxt; }
}
```

Implement the method

```
LNone deleteAt(LNode head, int pos)
```

which deletes the element at position `pos` from the list with head node `head`.

Assume that the first element in the list is at position zero (0).

If the list is empty (`head == null`) then the method should return `null`.

If the first element of the list is deleted (`pos == 0`) then the method should return the new head of the list.

In all other cases the method should return `head`.

[15 marks]

Question 2 [25 marks]

(a) You are given an algorithm **A** which runs in $O(n^2 \log_2(n))$. Providing justification, indicate which of the following statements are correct:

i. Algorithm **A** runs in $O(n^3)$. *[2 marks]*

ii. Algorithm **A** runs in $O(n^2)$. *[2 marks]*

iii. Algorithm **A** runs in $\Theta(n^3)$. *[2 marks]*

iv. Algorithm **A** runs in $\Theta(n^2 \log_2(n))$. *[2 marks]*

v. Algorithm **A** runs in $O(n^2 \log_{10}(n))$. *[2 marks]*

(b) A **binary tree** has nodes which are objects of the following class:

```
class TreeNode {
  int key;
  TreeNode left, right;
}
```

i. The following method `checkBounds` is given the root, `rt`, of a binary tree and two integers, `minval` and `maxval`.

The method returns **`true`** if all the nodes in the tree contain keys greater than `minval` and less than `maxval`. Otherwise, the method returns **`false`**.

```
public int checkBounds(TreeNode rt, int minval, int maxval)
```

Implement this method using recursion. *[8 marks]*

ii. Change your implementation of `checkBounds` so that it returns **`true`** if the following two conditions are satisfied:

- all the nodes in the input binary tree contain keys greater than `minval` and less than `maxval`,
- **and the tree is a binary search tree.**

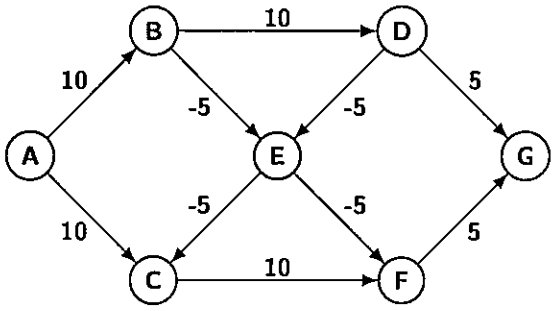
Otherwise, the method should return **`false`**. *[7 marks]*

Question 3 [25 marks]

(a) Describe what is the **postorder** and the **topological sort** of the vertices of a Directed Acyclic Graph. *[5 marks]*

(b) Explain in English the algorithm that finds the topological sort of a Directed Acyclic Graph. You can assume that basic algorithms such as DFS and BFS are known and do not need to be explained. *[5 marks]*

(c) Give the **postorder** and **topological sort** of the vertices of the following graph G (ignoring edge weights).



[5 marks]

(d) Three algorithms for finding the Shortest Path Tree (SPT) in Weighted Directed Graph, starting from a given source vertex **v** are: (1) **Dijkstra's** algorithm; (2) the algorithm that uses **Topological Sort**; (3) the **Bellman-Ford** algorithm.

For each of the three algorithms, discuss its running time and limitations. Describe the type of graphs for which each algorithm is best suited. *[5 marks]*

(e) Give the SPT, rooted at vertex **A**, in the above graph G. Which algorithm is best suited for finding the shortest paths in this graph? *[5 marks]*

PART B

Question 4 [25 marks]

(a) Implement the Java function

```
double binary_sqrt(double low, double high,
                   double tol, double x)
{ // Pre:  low^2 ≤ x < high^2
```

that uses the algorithm of Binary Search to return a result, r, such that

$$r \leq \sqrt{x} < r + tol$$

where *tol* is a small tolerance value e.g. 0.001.

Initially, $low^2 \leq x < high^2$.

[8 marks]

(b) Consider the `Basic_Matrix` class on the next page.

i. Implement from the `Basic_Matrix` class the following Java method:

```
int min_index(int[] arr)
{// returns an index of the minimum item in arr
```

[7 marks]

ii. A Matrix M has a Saddle Point iff **for some position (i,j), M(i,j) is the minimum of row i and maximum of column j.** Present a Java method

```
void one_saddle(Basic_Matrix m)
```

that will print out a saddle point, if any, of the matrix, `m`.

[10 marks]

```
class Basic_Matrix
{
    int rows;           // #rows
    int cols;           // #columns
    int[][] item;       // 2-d array

    Basic_Matrix_Math(int m, int n)
    {// create m x n matrix of 0's

    Basic_Matrix_Math(int [][] arr_2d)
    {// create matrix from a 2-d array, arr_2d

    Basic_Matrix transpose()
    {// return the transpose of the matrix

    int min_index(int[] arr)
    {// returns an index of the minimum item in arr

    int [] min_row()
    {// returns the array of the minimums of the rows
     // in the matrix

    int max_index(int[] arr)
    {// returns an index of the maximum item in arr

    int [] max_row()
    {// returns the array of the maximums of the rows
     // in the matrix

} // Basic_Matrix
```

Question 5 [25 marks]

Consider the Java class template for `Merge_sorter` on the next page. Implement the Java functions:

(a) `LinkedList<String> merge(LinkedList<String> left, LinkedList<String> right)` *[10 marks]*

(b) `LinkedList<String> merge_sort(LinkedList<String> in_list)` *[8 marks]*

(c) `LinkedList<String> reverse (LinkedList<String> xs)` *[7 marks]*

The Java Library Class, `LinkedList`, includes the following methods:
(where `T` is the type of items in the `LinkedList`)

```
isEmpty()

size()

removeFirst()
//removes and returns the first item

addFirst(T x)
//inserts the item, x, at the beginning of the LinkedList

addLast(T x)
//appends the item, x, to the end of the LinkedList

get(int k)
//returns the item at postion k.
```

The Java Library Class, ListIterator, includes methods such as:

```
hasNext()
//returns true if this list iterator has more items
//in the forward direction.

next()
//returns the next item in the list and advances the cursor.
```

```
import java.util.LinkedList;
import java.util.ListIterator;
class Merge_sorter
{
   Merge_sorter() {
       LinkedList<String> xs,zs;
       xs = from_file("in.txt");
       zs = merge_sort(xs);
       TextIO.put("\nSorted␣List␣=␣");
       print_list(zs);
   } // Merge_sorter

   LinkedList<String> merge(LinkedList<String>left,
                            LinkedList<String> right)
   { // is_ordered(left) && is_ordered(right)

   LinkedList<String> merge_sort(LinkedList<String> in_list)
   { // returns sorted version of in_list via
     // the algorithm for merge sort

   LinkedList<String> reverse (LinkedList<String>  xs)
   { // returns the reverse of the LinkedList, xs

   LinkedList<String> from_file(String File_name)
   { // reads from a file into a LinkedList

   void  print_list(LinkedList<String> xs)
   { // prints the items in the LinkedList, xs
      ListIterator<String> iter = xs.listIterator();
      TextIO.putln();
      while ( iter.hasNext() )
          TextIO.putln(iter.next());
   } // print_list

   boolean le(String s1, String s2)
   {  // return s1 <= s2

   boolean lt(String s1, String s2)
   { // returns  s1 < s2

} // Merge_sorter
```

Question 6 [25 marks]

(a) On an 8×8 board, implement a boolean function

```
boolean middle(int i, int j)
```

that will determine if (i, j) is a element of the 16 middle squares on the board. The set of the 16 middle squares can be defined by:
$middle = \{(i, j) \mid i \in \{0..7\} \wedge j \in \{0..7\} : 2 \leq i \leq 5 \wedge 2 \leq j \leq 5\}$

[7 marks]

(b) Consider a class, `Eight_Queens`, [see next page] for finding the solutions to the 8-Queens problem of placing 8 queens on an 8×8 chessboard so that no queen can take another with the constraint that no queen is on any of the 16 middle squares.

Implement the methods

i. `boolean safe(int i, int j)`

ii. `void all_queens(int i)`
that find all solutions to the 8-Queens problem with the constraint that no queen is on any of the 16 middle squares.

[18 marks]

```
class Eight_Queens
{
   int [] q;
   boolean [] used_col;
   boolean [] up_diag;
   boolean [] down_diag;

   Eight_Queens()
   {
        q = new int [8];
        used_col = new boolean [8];
        up_diag = new boolean [15];
        down_diag = new boolean [15];
        TextIO.putln("The␣solutions␣are:␣");
        all_queens(0);
   } // Eight_Queens

   void all_queens(int i)
   {//generates all solutions to the 8-Queens problem with the
    //constraint that no queen is on any of the 16 middle squares

   boolean safe(int i, int j)
   { // is the position (i,j) safe for a queen

   void set_queen(int i, int j)
   {// the queen is set on board

   void reset_queen(int i, int j)
   {// the board is reset for another attempt

   boolean middle(int i, int j)
   {// (i,j) is a element of the 16 middle squares

   void print_queen()
   {// prints a solution to the queens problem

} // Eight_Queens
```