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

Faculty of Engineering, Mathematics & Science

School of Computer Science and Statistics

Integrated Computer Science Programme **Trinity Term 2016**
BA (Mod) CSLL
BA (Mod) Business & Computing
BA MSISS
Year 2 Annual Examinations

CS2010: Algorithms and Data Structures

Wednesday 4 May 2015 **RDS Main Hall** **9:30 – 12:30**

Dr. Hugh Gibbons, Dr. Vasileios Koutavas

Instructions to Candidates:

- This exam paper 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) Write the main operations in the API of the **Union-Find** Abstract Data Type (ADT) and briefly describe what they do. Using the asymptotic notation, give the **worst-case running time** of each of these operations in the standard implementation of Union-Find (without performance improvements). Briefly explain why each operation has the running time you gave.

Describe an implementation strategy of the Union-Find operations that will result in $O(\lg n)$ **amortized running time.** *[10 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

```
  LLNone appendIfNew(LNode head, int num)
```

which adds num at the end of the list with head node head, **provided num does not already exist in the list.**

If the list is empty (head == **null**) then the method returns a new head node for the list, containing num. Othewise it returns head. *[15 marks]*

Question 2 [25 marks]

(a) Write the following asymptotic costs in order, from the smallest to the largest, writing one of the symbols $<$ or $=$ between them. If two asymptotic costs are equal then their relative order is unimportant.

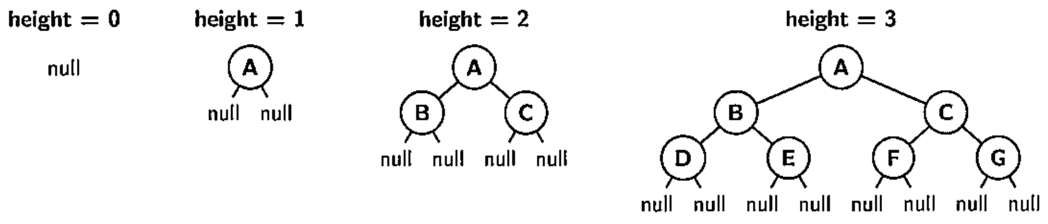
$$O(n^2+2n+1) \quad O(n^2) \quad O(n^3+n^2+2n+1) \quad O(n^2\log_{10}(n))$$
$$O(10^{10}) \quad O(n^{10}) \quad O(\log_2(n)) \quad O(\log_{10}(n))$$

[10 marks]

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

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

A binary tree is **full** if each node has either: two non-null children, or two null children. Note that in such a tree all null nodes have the same distance from the root. The following are examples of full binary trees.



Implement the following method which returns the height of the tree with root `root`, if the tree is full, and -1 otherwise.

```
public int fullHeight(TreeNode root)
```

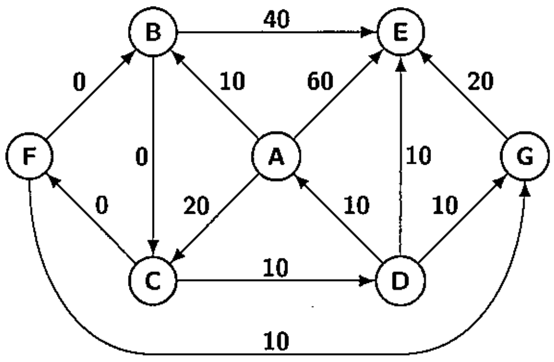
The method should return zero (0) if the root of the tree is **null**.

Your implementation should **run in $O(N)$ time** in the worst case, where N is the number of nodes in the tree. *[15 marks]*

Question 3 [25 marks]

(a) Describe what is the **shortest path tree (SPT)**, starting from a given source vertex **v**, in a weighted directed graph G. *[5 marks]*

(b) Illustrate **Dijkstra's algorithm** by showing the iterations of the algorithm for solving the SPT problem in the following graph, starting from the source vertex **A**:



For each iteration of the algorithm give a table containing one row for each vertex of the graph containing:

- the vertex name
- its computed distance from vertex **A** at the end of the iteration
- the edge in the SPT leading to the vertex at the end of the iteration

The table for the first iteration is:

1st iteration:

Visited	Vertex	Distance from **A**	Edge to vertex
→	**A**	0	–
	B	10	**A** → **B**
	C	20	**A** → **C**
	D	–	–
	E	60	**A** → **E**
	F	–	–
	G	–	–

[15 marks]

(c) Give the SPT of the above graph with root at vertex **A**. *[5 marks]*

PART B

Question 4 [25 marks]

(a) Implement the Java functions

```
boolean isOrdered(String [] words, int low, int high)
{// Pre: arr != null
```

that returns whether the array section, words[low..high], is (alphabetically) ordered.

and

```
int count(String w, String [] words, int low, int high)
{ assert ( isOrdered(words, low, high) )
```

that counts the number of occurrences of a word, w, in the ordered section, words[low..high]. If the word, w, does not occur the function returns 0. In an ordered array all the same repeated items are contiguous. *[10 marks]*

(b) Consider the the following Java Matrix class

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

    Matrix_Math(int r, int c)
    {// create r x c matrix of 0's

    Matrix_Math identity(int n)
    {// Identity matrix of order n

    Matrix_Math copy_mat()
    { // returns a copy of the this matrix

    Matrix_Math times(Matrix_Math B)
    { // returns a the matrix product:  this*B

    Matrix_Math f_exp_mat(int n)
    { // rows = cols  (square matrix) && n >= 0
      // result = this^n

} // Matrix_Math
```

(Q.4.b continues on next page)

i. Implement the function for matrix product by completing the missing code in the following Java function:

```
Matrix_Math times(Matrix_Math B)
{ assert( cols == B.rows );
        Matrix_Math C = new Matrix_Math(rows, B.cols);
        for (int i = 0; i < C.rows; i=i+1)
            for (int j = 0; j < C.cols; j=j+1)
                << missing code >>
        return C;
} // Post: C = this*B
```

The function call, `A.times(B)`, returns the matrix product, A*B.

ii. A Java function can be implemented that returns M^n in $O(\log b)$ running time based on the following properties of, x^k, (Integer $k \geq 0$).

$$x^0 = 1$$
$$x^k = (x * x)^{\frac{k}{2}},\ \textit{if even}(k)$$
$$x^k = x * x^{k-1},\ \textit{if odd}(k)$$

Implement the following $O(\log n)$ Java function, `f_exp_mat(int n)`, by completing the missing code:

```
Matrix_Math f_exp_mat(int n)
{ // Pre: rows = cols  (square matrix) && n >= 0
   Matrix_Math result = new Matrix_Math(rows,cols);
   Matrix_Math mm = new Matrix_Math(rows,cols);
   if ( n == 0 )
        result = identity(rows);
   else if ( n == 1 )
            result = copy_mat();
        else
        {
           mm = copy_mat();
           result = identity(rows);
           << missing code >>
        }
        return result;
} //Post:  result = this^n
```

iii. Let the matrix $F = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$. Using the mathematical result that

$$F^n = \begin{bmatrix} \textit{fib}(n-1) & \textit{fib}(n) \\ \textit{fib}(n) & \textit{fib}(n+1) \end{bmatrix}$$

where $\textit{fib}(n)$ is the n^{th} fibonacci number, implement a Java function, `fib(int n)`, that will return the n^{th} fibonacci number.

[15 marks]

Question 5 [25 marks]

(a) Assume a Java class has an array attribute,

```
String [] arr;
```

and integer attributes,

```
int L, R;
```

Present a Java method

```
void partition(int L0, int R0, String p)
```

that will partition in place the array, `arr`, into 2 sections.

After partitioning, the items in the left section are less than or equal to the pivot item, p, and the pivot item, p, is less than or equal to all the items in the right section, i.e. after partitioning, the array, `arr`, satisfies

L0 R	L R0
items $\leq$ p	p $\leq$ items

[9 marks]

(b) Implement a Java function

```
String [] merge(String [] left, String [] right)
{// Pre: is_ordered(left) && is_ordered(right)
```

that will merge two ordered arrays, `left` and `right`, into an ordered array.

[9 marks]

(c) By filling in the missing code in the following

```
String [] merge_sort(String [] in_arr, int low, int high)
{ // Sort in_arr by method of merge sort for arrays
    assert (low < high);
    String [] left, right;
    String [] result = new String[high-low];
    if ( high - low == 1 )
        result[0] = in_arr[low];
    else
    {
           << missing code >>
    }
    return result;
} // merge_sort
```

implement a Java function

```
String [] merge_sort(String [] in_arr, int low, int high)
```

that will sort the array segment, `in_arr[low.high]`, by the algorithm of merge sort. *[7 marks]*

Question 6 [25 marks]

Consider the following Java class

```
public class Knights
{
  int [] [] board;
  int size;
  boolean success;
  int [] row_offset = {-2,-1,1,2,2,1,-1,-2};
  int [] col_offset = {1,2,2,1,-1,-2,-2,-1};
  int exit_x, exit_y;

  Knights (int r, int c, int enter_x, int enter_y,
           int xn, int yn)
  {
     board = new int [r] [c];
     size = r*c;
     exit_x = xn; exit_y = yn;
     board[enter_x][enter_y] = 1;
     success = false;
     try_next_move(2,enter_x,enter_y);
     if ( success )
       display_board();
     else
       TextIO.putln("No␣Solution");
  } // Knights

  void try_next_move(int k, int x, int y)
  { // Try move number k from position (x,y)
  boolean acceptable(int s, int t)
  { // is position (s, t) valid
  void display_board()

} // Knights
```

that finds a Knight's Journey on an r×c rectangular board, if there is one, that starts at square (enter_x, enter_y) and finishes at square (exit_x, exit_y) after visiting all squares on the board exactly once, i.e. the Knight 'enters' the board on square (enter_x, enter_y), visits all squares on the board exactly once and then 'exits' the board from square (exit_x, exit_y).
Implement the methods:

```
void try_next_move(int k, int x, int y)
```

and

```
boolean acceptable(int s, int t))
```

[25 marks]