CS201 Data Structures November 25, 2004

FINAL EXAM

Attempt all questions. Each question carries 8 marks.

1. Suppose n people are arranged in a circle. Number the people from 1 to n in the clockwise order. We are given an integer $m \leq n$. Beginning with the person with designated number 1, we proceed around the circle (in clockwise order) removing every m^{th} person. After each person is removed, counting continues around the circle that remains. This process continues until all the n people have been removed. The m -- *permutaion* is defined as the order in which the people have been removed. As an example, if $n = 7, m = 3$, then the $3-$ *permutation* is $3, 6, 2, 7, 5, 1, 4$.

   - Give an $O(mn)$ time algorithm which given m outputs the $m-$*permutation*.
   - Give an $O(n \log n)$ time algorithm which given m, outputs the $m-$*permutation*.

2. Suppose there are n people living in a city. We number the people from 1 to n. We would like to represent the information about who knows who by a two dimensional array. So we are given an $n \times n$ array A. $A[i, j] = 1$ if i knows j, otherwise it is 0 (note that it is possible that $A[i, j] = 1$ but $A[j, i] = 0$ – for example, many people may know a film star, but the reverse may not be true). You can assume that $A[i, i] = 1$ for all i. We say that a person i is a *celebrity* if everyone knows this person, but this person does not know anyone other than himself. First show that there can be at most one celebrity. Now give an $O(n)$ time algorithm which given the array A, outputs YES if there is a celebrity in the city; otherwise it outputs NO. Further, if there is a celebrity, it should output its number.

3. Let G be a connected undirected graph. Suppose there is a tree which is both a breadth first search tree and a depth first search tree of G. Is it true that G must be a tree ? Either prove your answer or give a counterexample.

4. Suppose you are given a connected undirected graph G in the adjacency list data-structure. Each edge e has a length l_e. Give an algorithm which given an edge e of G, outputs YES if e is in a minimum spanning tree of G, otherwise it outputs NO. The running time of your algorithm should be $O(m)$, where m is the number of edges in G.

5. Suppose you are given a connected undirected graph $G = (V, E)$. Each edge has a length l_e. Further, with each edge e, we are also given a quantity t_e. t_e basically represents the amount of tax we shall have to pay if we use edge e. As discussed in the class, the length of a path is the sum of l_e of all the edges in this path. Similarly, define the tax paid for using a path as the sum of t_e of all the edges in the path. We are given two vertices u and v in G. We would like to find a path from u to v of smallest length. But there may be several such shortest paths – among these we would like to

prefer the one on which the tax paid is smallest. In other words, we would like to find the shortest path (i.e., a path of smallest length) from u to v on which the tax paid is as small as possible. Give an $O(m \log n)$ time algorithm for this problem, where $m = |E|, n = |V|$.

Name:_____ Entry No:_____

CSL201: Data Structures. II semester, 2007-08.
Major Exam
3:30 PM to 5:30 PM, 3rd April 2008

Question	1	2	3	4	5	6	Total
Marks							

Note. Explain your solutions in words. **No pseudocode**. Write concisely and clearly. If necessary solve the problems on the rough sheet first and then copy clearly onto this sheet.

Q1. We consider a *deterministic* version of skip lists in this question (i.e. without the use of probability) called *1-2-3 skip lists*. Before we describe this structure, recall that each node in a skip list has a *height* associated with it (denoted $h(\cdot)$) which corresponds to the highest list it belongs to i.e. if a node is only in the base list it has height 0, if it was promoted to the next level but not further up it has height 1 and so forth. Now, we describe 1-2-3 skip lists as the skip list dictionary structure on a set S which maintains the following invariant:

> Given 2 keys $x \leq y \in S$, if $h = \min\{h(x), h(y)\} \geqslant 1$ there must be at least one and at most 3 keys $z \in S$ such that $x < z < y$ and $h(z) = h - 1$.Sentinel nodes (at the two extremes) have the height of $current.max + 1$

Q1.1. Starting from an empty 1-2-3 skip list, insert the following items (in the order given): 3, 1, 6, 2, 9, 8, 4, 5, 14, 10, 11, 7, 12, 13, 15. You **do not** have to show all steps, just show the final skip list. **(6 marks)**

Name:_____ Entry No:_____

CSL201: Data Structures. II semester, 2007-08.
Major Exam
3:30 PM to 5:30 PM, 3rd April 2008

Question	1	2	3	4	5	6	Total
Marks							

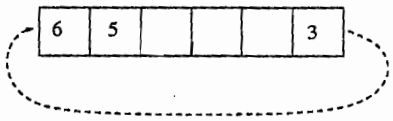
Note. Explain your solutions in words. **No pseudocode**. Write concisely and clearly. If necessary solve the problems on the rough sheet first and then copy clearly onto this sheet.

Q1We consider a *deterministic* version of skip lists in this question (i.e. without the use of probability) called *1-2-3 skip lists*. Before we describe this structure, recall that each node in a skip list has a *height* associated with it (denoted $h(\cdot)$) which corresponds to the highest list it belongs to i.e. if a node is only in the base list it has height 0, if it was promoted to the next level but not further up it has height 1 and so forth. Now, we describe 1-2-3 skip lists as the skip list dictionary structure on a set S which maintains the following invariant:

> Given 2 keys $x \leq y \in S$, if $h = \min\{h(x), h(y)\} \geq 1$ there must be at least one and at most 3 keys $z \in S$ such that $x < z < y$ and $h(z) = h - 1$.Sentinel nodes (at the two extremes) have the height of $current.max + 1$

Q1.1. Starting from an empty 1-2-3 skip list, insert the following items (in the order given): 3, 1, 6, 2, 9, 8, 4, 5, 14, 10, 11, 7, 12, 13, 15. You **do not** have to show all steps, just show the final skip list. **(6 marks)**

Consider an array A of integers of length N. We would like to think of this as a "circular array", i.e., after the last element A[N-1], we go to A[0] (see the picture below for a circular array of length 6). We would like to implement a list of numbers using this array. We maintain two variables start and last - start is equal to the first element of the list of numbers and last is the last element in the list. For example, suppose the array is storing the list of numbers $\{3, 6, 5\}$. If start = 2, last = 4, then A[2]=3, A[3] = 6, A[4] = 5. Note that the variable last could be less than start. For example, if start = 5, last = 1, then A[5] = 3, A[0] = 6, A[1] = 5 (see the figure below).



Complete the following JAVA code which maintains a list of numbers and adds or removes elements from either the beginning or the end of the list. Note some important points :

- If the list is empty, the values of start, last = -1.
- The array can not store a list of size more than N.

In the code below, assume that N is some fixed number. (3×4)

```
class ListofNumbers {

  int[] A;
  int start, last;

  public ListofNumbers() {
      A = new int[N];
      start = -1;
      last = -1;
  }

  public void InsertBeginning(int x) { // insert x at the beginning of the list









  }

  public void InsertEnd(int x) { // insert x at the end of the list
```

```
  }
  public void RemoveEnd(int x) { // remove from the end of the list




  }
```

The methods **InsertBeginning** and **InsertEnd** should print the message **"Array is Full"** if the array is already storing **N** numbers. Similarly, the method **RemoveEnd** should print the message **"Array is empty"** if the array is not storing any number.

An *expression* is defined as below.

```
type expression =
 Const of float
| Var of string
| Sum of expression*expression
| Diff of expression*expression
| Prod of expression*expression
;;
```

For example `Prod ( Const 3.5 , Var "x" )` is a valid expression.

(a) What are the valid expressions corresponding to (2 × 2)
(i) $5x + 7y$

(ii) $(xy + yz) \cdot x$

(b) You can verify that the derivative of such an expression can be calculated using the rules of differentiation, and is also an expression. For example, `deriv Var "x" "x"` should be `Const 1.0` where `deriv exp var` returns the derivative of the expression **exp** with respect to the variable **var**. Complete the following Ocaml program that returns derivative expression of a given input expression. (8)

```
let rec deriv exp dv =

 match exp with

   Const c -> Const 0.0
 | Var v -> if v = dv then Const 1.0 else __________

 | Sum (f,g) -> ____________________________________

 | Diff (f,g) -> ___________________________________

 | Prod(f,g) -> ____________________________________
```

gesort is described as a recursive function that sorts the first half of the elements (recursively), sorts the other half recursively and subsequently merges the two sorted sets. The following ram implements mergesort in an array WITHOUT using recursive calls. Since every element of initial array can be regarded as 1 element sorted sequence, we can merge them into $n/2$ sorted iences of length 2. Subsequently the 2 element sequences can be merged into 4 element sequences so on till there is one sorted sequence. At any stage we use the variable `runlength` to denote length of the sorted sequences (initially 1). Also assume the length of the array is a power of 2, $n = 2^k$ for some $k \geq 1$. Complete the following function in Java that implements this strategy. **NOT define any new variables.** (9)

```
public static void mergesort(double [] A ) {


   int runlength = 2;  // we need not start with runlength = 1 because
                       // sequences of length 1 are already sorted.
   int i, b ;

   while (runlength <= A.length) {

    for ( i = 0 ; ______________________________ ; i = i+ runlength )

    {
       b = runlength/2 ;


       ____________________________________________________
    } // end for

    runlength = ________________________
    } // end while
} //end mergesort

public static void merge (int i1, int i2, int j1, int j2 , double [] A)
```

A[i1] ... A[i2] and A[j1] ... A[j2] are two disjoint sorted subarrays where i1 $\leq$ i2 and j1 $\leq$ j2. The function merges these two subarrays into a sorted sequence and stores the result in the same locations, i.e., A[i1] ... A[i2] and A[j1] ... A[j2]. For example if A = {10, 5, **6, 12, 15**, 3, 1, **7, 9, 18** }, i1 = 2, i2 = 4, j1 = 7, j2 = 9, then after the function finishes, A = {10, 5, **6, 7, 9**, 3, 1, **12, 15, 18** }. You can use this function to merge two sorted portions of the array - you don't have to write the code.

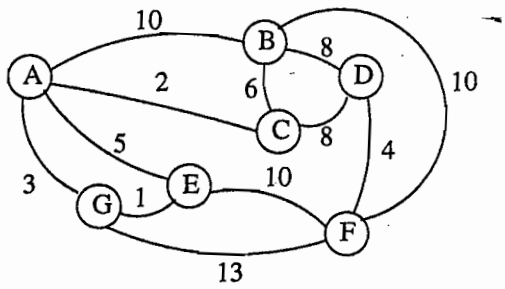
(ii) What is the number of comparisons used in the iterative mergesort ? (Show the analysis assuming that merging two sorted sequences takes time proportional to the sum of the lengths of the two sorted sequences). (3)

mplete the following program that copies one file to another except that consecutive blanks are laced by a single blank. Recall that -1 is the equivalent integer (of character) for end-of-file. The nk character is denoted by ' '. Do NOT use the methods `trim` or `split` of the Java `String` class. **NOT define any new variables.** (12)

```
port java.io.* ;
lass delblanks {

ublic static void main (String[] args)
 throws IOException
{

FileReader fr = null;
PrintWriter pr = new PrintWriter(new BufferedWriter(new FileWriter
                        (args[1]))) ;

 fr = new FileReader(args[0]) ;



 int cc = 0 ;

 while ( __________ != -1 ) {
    cc = fr.read() ;


    if (_____________ == ' ') {

       pr.print(_____________) ;

       while ( cc == ' ') {

         cc = ______________________;

       }
   }
   if (cc != ' ' && cc != _____________ ) {pr.print((char)cc) ; }
     //(char) cc converts the integer represented by cc to the corresponding ASCII
 }


      ____________________;
} //main

} //class
```

Name:_____ **Entry No:**_____

Q4. Show that the maximum height of any 1-2-3 skip list is $\theta(\log n)$. **(6 marks)**



Q2. Run Dijkstra's algorithm on the network given in the figure starting from vertex $s = A$. Show the steps in running the algorithm, you do not need to draw the graph repeatedly, just write which is the next vertex picked and which labels get updated from what value to what value at each step. **(10 marks)**

Name:_____ Entry No:_____

Q1.2. Give the algorithm for inserting an item into a 1-2-3 skip list as used in Q1.1. **(5 marks)**

Q1.3. Is the order of insertion important? Do you get the same 1-2-3 skip list no matter which order the keys are inserted in? If yes, prove your answer. If no, give a counter example. **(5 marks)**

Name:_____ Entry No:_____

Q3. Given an arbitrary binary tree T with integer keys stored at the nodes, design an efficient algorithm which determines whether or not T is a binary search tree. What is the time complexity of your algorithm? **(10 marks)**

Name:_____ Entry No:_____

4. Prove that in a binary tree $n_0 = n_2 + 1$, where, n_0 is the number of nodes with no children is the number of nodes with two children. **(8 marks)**

5. You are given one storage stack (S) and the one display queue (D). A sequence of alphabets **abcdefgh** is used as input. One alphabet at a time from the sequence is either pushed in S or enqueued in D. An alphabet popped from S is enqueued in D. At the end all the items of D are dequeued and the resulting output is **cedbghfa**. Show the sequence of operations that leads to this output. **(8 marks)**

Name:_____ Entry No:_____

Q6. Given a sequence of n integers, a pair of of elements x and y are called an *inversion* in this sequence if $x > y$ but x occurs before y in the sequence. Give an $O(n \log n)$ algorithm for finding the number of inversions in a sequence. [**Hint**:Modify merge sort]. **(12 marks)**

CSL201 II Sem 2006-07, Major Exam

3:30PM to 5:30PM, 8th May 2007

Response Sheet Code 1

Name:

Entry No:

Encircle the correct response(s)

Q. No.	Options	Score
1.	A B C D E	
2.	A B C D E	
3.	A B C D E	
4.	A B C D E	
5.	A B C D E	
6.	A B C D E	
7.	A B C D E	
8.	A B C D E	
9.	A B C D E	
10.	A B C D E	
11.	A B C D E	
12.	A B C D E	
13.	A B C D E	
14.	A B C D E	
15.	A B C D E	
16.	A B C D E	
17.	A B C D E	
18.	A B C D E	
19.	A B C D E	
20.	A B C D E	
Total		

CSL201 II Sem 2006-07, Major Exam

3:30PM to 5:30PM, 8th May 2007

Note. There are twenty multiple choice questions in this exam. Questions may have more than one correct answer. Marking **all** the correct answers of a given question gets you **1.5 marks**. Marking even a single wrong answer gets you **- 0.5 marks**. Marking some correct answers (and no wrong answers) gets you **0.5 marks**.

All answers are to be marked in the response sheet attached to the question paper. Please detach this response sheet from the question paper. There are 6 different question papers with codes from 1 to 6. **Please ensure that the code on top of your response sheet matches the code on the question paper.**

1. You are given a single stack and a sequence of six numbers 123456 that appear as input one at a time. As each number appears it can either be pushed on the stack or sent to output. When a number on the stack is popped it has to be sent to the output (there is no additional storage available.) Given this system, which of the following permutations of this sequence can be output?

   (a) 164235
   (b) 654123
   (c) 465321
   (d) 132546

2. Consider the 2-4 tree given in Figure 2. Which of the following statements are true:

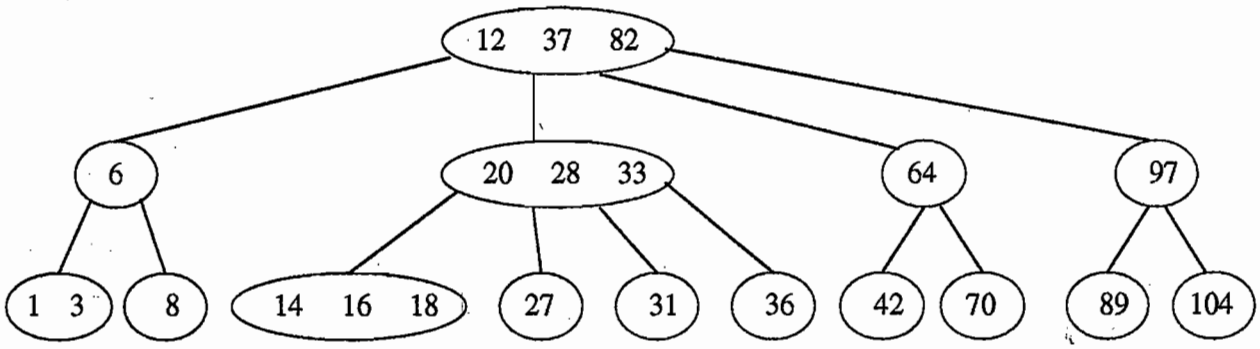


Figure 1: A 2-4 tree

   (a) Inserting 17 increases the height of the tree.
   (b) The tree could have 28 in its root node after 17 is inserted.
   (c) Inserting 90, 91, 92, 93, 94, 95, 96 increases the height of the tree.
   (d) Inserting 29, 30, 32, 33 decreases the height of the tree.

3. Suppose we are given a minimum spanning tree T for a graph G. Let $e = (u, v)$ be an edge of G which is not in T. Suppose we **decrease** the length of edge e from l_e to l'_e. In which of the following cases, we can be **sure** that T will **not** remain a minimum spanning tree.

   (a) l'_e is less than the lengths of all other edges which have an end-point in u or v.
   (b) The path from u to v in T has an edge of length larger than l'_e.

(c) There is a cycle in G which contains e such that l'_e is the length of the smallest edge in this cycle.

(d) Every cycle containing e has at least one edge of length larger than l'_e.

4. A k-heap is a heap where each non-leaf node has exactly k children. It has the same properties as those of a binary heap. What is the worst case running time of **insert** and **delete-min** operations (let n denote the number of elements in the heap).

   (a) $O(\log_k n)$ and $O(k \log_k n)$.

   (b) $O(k \log_k n)$ and $O(\log_k n)$.

   (c) $O(k \log_k n)$ and $O(k \log_k n)$.

   (d) $O(\log_k n)$ and $O(\log_k n)$.

5. The inorder traversal for a binary tree is DEACBFGIKJOLPHMNRQS. The preorder and postorder traversals are:

   (a) IGFEDCABHJKLOPMNQRS and DABCEFGKOPLJRSQNMHI

   (b) IGFEDCBAHJKLOPMNQRS and DBACEFGKOPLJRSQNMHI

   (c) KIGFEDCABHJLPOMNQRS and DABCEFGIPOLJRSQNMHK

   (d) KIGFEDCABHJLOPMNQRS and DABCEFGIOPLJRSQNMHK

6. Suppose that while your computer is sorting an array of objects, its memory is struck by a cosmic ray that changes exactly one of the keys to something completely different.For which of these sorting algorithms will the final array have just one or two keys out of place in the worst-case scenario ?

   (a) Quick sort

   (b) Merge sort

   (c) Radix sort

   (d) Heap sort

7. Suppose that we are using double hashing and have selected the table size m to be 1200. When we do an insertion, the first hash function determines the point at which we initially probe the table, and the second hash function determines the amount by which we jump (mod 1200) as we search for an empty table position. Which of the values below for the second hash function would guarantee that we will find an empty position if there is one in the table?

   (a) 95

   (b) 74

   (c) 53

   (d) 39

   (e) 27

8. Arrange the following functions in order of their asymptotic grwoth rates, i.e., if $f(n)$ appears to the left of $g(n)$, then $f(n)$ is $O(g(n))$:

$$2^n, n!, n^n, n^{\log^3 n}$$

   (a) $n^{\log^3 n}, 2^n, n^n, n!$

   (b) $n^{\log^3 n}, 2^n, n!, n^n$

(c) $2^n, n^{\log^3 n}, n^n, n!$

(d) $2^n, n^{\log^3 n}, n!, n^n$

9. Suppose we start with an AVL tree T . We do an insertion of a new key in the tree just using the binary search tree algorithm without yet doing any rotations; the result is a tree T'. It turns out that the deepest node in the tree with a balance outside of $\{1, 0, -1\}$ is is the root, so we do an appropriate rotation at the root to produce the final AVL tree T''. If the height of T'' is 10, what were the heights of T and T' respectively ?

   (a) 10 and 11

   (b) 11 and 10

   (c) 11 and 11

   (d) 10 and 10

   (e) We do not have enough information to determine the answer.

10. Which of the following sorting algorithms cannot be implemented as a stable sorting algorithm.

    (a) Quick sort

    (b) Merge sort

    (c) Radix sort

    (d) Heap sort

11. Consider an undirected connected graph G with edge lengths. Assume all edge lengths are positive integers. We would like to find the **length** of shortest path from s to every other vertex in G. We initialize an array $D[\cdot]$ to $D[s] = 0$, $D[v] = \infty$ if $v \neq s$. Consider the following algorithm. Here $l(e)$ denotes the length of edge e.

```
repeat
   find an edge e = (u,v) such that D[u] > D[v] + l(e)
   update D[u] to D[v] + l(e).
until no such edge can be found
```

    Which of the following statements are true about this algorithm.

    (a) The algorithm may not terminate on some graphs.

    (b) The algorithm always terminates. But there exist graphs for which the final values $D[v]$ are not equal to the length of shortest path from s to v for some vertices.

    (c) The algorithm always terminates. Further, on termination the values $D[v]$ are equal to the length of the shortest s to v path for every vertex v.

    (d) The values $D[v]$ are always at least the length of the shortest s to v path.

12. Suppose we want to find shortest path from a source vertex s to a vertex t in an undirected graphs. However some edges in the graph may have negative lengths. In which of the following cases will the Dijkstra's algorithm find the desired shortest path ?

    (a) No shortest path from s to t has an edge of negative length.

    (b) There is at most one edge of negative length

(c) There does not exist a cycle in which the sum of the lengths of the edges in it is negative.

(d) None of the above.

13. Suppose that we have built up a compressed trie for a list of strings. The rightmost path of this trie is as shown in Figure 2, where the numbers to the left of each node specify the total length of the labels on the path from the root to that node. Let v be the lexicographically largest of the strings we have inserted. We now add a new string w which is lexicographically larger than v, and for which the length of the longest common prefix of v and w is 17. In the new compressed trie, how many edges are there on the rightmost path?

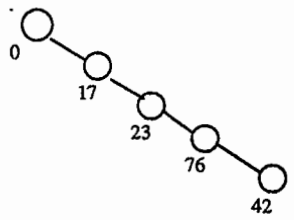


Figure 2:

(a) 5

(b) 4

(c) 3

(d) 2

(e) 1

14. Consider again the 2-4 tree of height 2 given in Figure 2. Which of the following statements are true:

(a) After 27 is deleted the final tree has 20 a leaf node.

(b) After 37 is deleted the final tree has 42 in the root node.

(c) 7 is the min number of keys in a 2-4 tree of height 2.

(d) At least 15 keys have to be deleted from the tree in Figure 2 to decrease its height.

15. Suppose we run the DFS and the BFS algorithms on a connected undirected graph G. It so happens that the DFS tree and the BFS tree obtained from these algorithms are identical. What can we claim about the graph ?

(a) The graph does not have a cycle

(b) The graph can have one cycle

(c) The graph can have several cycles but no two of these cycles share a common vertex.

(d) No such conclusion can be drawn.

16. In a skip list each node is promoted to the next level with probability $1 - \frac{1}{\log n}$. The expected height of a node in such a skip list is

(a) $O(1)$

(b) $\theta(\log n)$

(c) $\theta(1)$

(d) $O(\log \log n)$

17. Consider a connected undirected graph G. Each edge e has **two** lengths, namely $l(e)$ and $w(e)$. Let s be a starting vertex in G. Consider the following modified Dijkstra's algorithm. Here $D[\cdot]$ and $F[\cdot]$ are two arrays initialized to $D[s] = F[s] = 0$ and $D[v] = F[v] = \infty$ if $v \neq s$. Let n denote the number of vertices in G.

```
Initialize a set M to emptyset
While M has less than n elements do
   pick a vertex v which is not in M and for which D[v] is minimum
   add v to M
   for every neighbour u of v such  that u is not in M do {
         if D[u] > D[v] + l((u,v)) {
            update D[u] = D[v} + l((u,v))
                   F[u] = F[v] + w((u,v))
            update parent[u] = v

          }
         else if D[u] == D[v] + l((u,v)) AND F[u] < F[v] + w((u,v)) {
            update F[u] = F[v] + w((u,v))
            update parent[u] = v
         }
   }
```

What does this algorithm do ? For a path P, and a function $f(e)$ on edges of the graph, define $f(P)$ as the sum of $f(e)$ for all edges e in P.

(a) For every vertex v, it finds a path P from s to v for which $f(P)$ is minimum, where $f(e) = l(e) + w(e)$.

(b) For every vertex v, it finds a path P from s to v for which $f(P)$ is minimum, where $f(e) = \min(l(e), w(e))$.

(c) For every vertex v, it finds a path from s to v for which $l(P)$ is minimum. Further if there are several shortest paths from s to v in terms of length $l(e)$, then it outputs the path among these for which $w(P)$ is minimum.

(d) For every vertex v, it finds a path from s to v for which $w(P)$ is minimum. Further if there are several shortest paths from s to v in terms of length $w(e)$, then it outputs the path among these for which $l(P)$ is minimum.

18. Consider a linked list where each node has a **next** pointer. We say that the list has a loop if there are nodes $a_1, a_2, \ldots, a_k$ in the list such that the a_1.next $= a_2$, a_2.next $= a_3, \ldots, a_{k-1}$.next $= a_k$ and a_k.next $= a_1$. Consider the following code to find out if a list L has a loop or not. Here p and q are pointers. What is the worst case running time of this code ? Let n denote the number of nodes.

```
p =  L
q = p.next
loop_exists = FALSE
```

```
while (p!=NULL && loop_exists = FALSE) {
    if (p == q)
        loop_exists = TRUE
    else
        { p = p.next
          q = (q.next).next
        }
}

return loop_exists
```

(a) $O(n^2)$

(b) $O(\log n)$

(c) $O(n \log n)$

(d) $O(n)$

19. Suppose an undirected graph G has 100 vertices and 28 edges. What are the maximum and the minimum number of possible connected components in the graph ?

(a) 93 and 86

(b) 93 and 72

(c) 86 and 72

(d) 93 and 86

20. Define the length of a cycle as the number of edges (or vertices) in the cycle. The **girth** of a graph is the smallest length of a cycle in the graph. Consider the following algorithm find the girth of a connected undirected graph.

```
Pick a vertex u in G.
Starting from u, run the BFS algorithm on G.
Let T be the BFS tree obtained
Define the level of a vertex v as its level in this tree (level of root is
For every edge e=(v,w) which is not in T
   compute a quantity Q(e) = 1 + level(v) + level(w).
Output the smallest Q(e).
```

Which of the following are true about this algorithm ?

(a) The algorithm outputs the girth of the graph.

(b) The algorithm outputs the girth if the starting vertex u lies in a cycle which has length equal to the girth of the graph. Otherwise the graph may give a wrong result.

(c) The algorithm will always give a wrong result if the starting vertex u does not lie on a cycle whose length is equal to the girth of the graph.

(d) There are graphs for which the algorithm will give a wrong answer no matter how we pick the starting vertex u.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEEING
MAJOR : CSL201
(Data Structures)

Max. Time – 2 hrs. **Max. Marks 70** **Date: 28/Nov/06**

Note: Answers of all the questions with all parts in the sequence.

Q1. (24)

a. Write an algorithm to reverse the order of items in a queue Q using stack S. Use standard functions of queue & stack. (4)

b. Suppose that you have to implement MIN-STACK which supports the stack operations **PUSH** and **POP** and a third operation **FindSum** which returns the sum of all elements in the stack, all in O(1) worst case time. Specify the data structures used along with implementation of all these three operations. (1+4)

c. Design an efficient algorithm to find the middle node of a singly linked list. [Hint use multiple pointers and a single loop]. (4)

d. Given a list of 'n' integers along with count field with each key containing the count (number of keys < the key), write an efficient algorithm to sort this list without using additional array. State the complexity of your algorithm. (4+1)

e. You are given two AVL trees T1 and T2. Write an algorithm to merge T1 and T2 to get a height balanced tree. (6)

Q2. (18) Please read parts (a to d) of this question carefully and answer all of them.

Rather than writing a stack and queue implementation from scratch, you decide to use a priority queue that is already implemented. You realize that by assigning the right priorities to data items when they are inserted, you can make data come out in either a LIFO or FIFO order. Your priority queue class has the following prototype:

```
class Pqueue
{
public:
void insert(int priority, dataType D); // Insert D with priority
dataType getmax();      // Get the max element and remove it
int empty();            // Return 1 if Pqueue is empty, 0 otherwise
Pqueue();               // Create an empty priority queue
};
```

and the insert definition says:

```
void Pqueue::insert(int priority, dataType D)
{ // pre: priority is any integer, positive or negative...
```

Both your stack and your queue should have member functions

```
void insert(dataType D);  // Insert data into the stack/queue
dataType getnext(); // Get & remove the next element (in LIFO/FIFO order)
int empty();  // Return 1 if stack/queue is empty, 0 otherwise
```

a. Write the class prototype for STACK and QUEUE which includes private data fields, and use a comment to describe each entry of the class: (2+2)

b. Write the **getnext()** member function for the STACK. Briefly describe how you w change it if you were to implement a queue. (3
c. Write the **insert(dataType D)** member function for the STACK. Briefly describe how would change it if you were to implement a QUEUE. (3
d. If Pqueue class is implemented with a HEAP, then give the tightest big-O bounds you ca
   i. **getnext()** for your QUEUE class,
   ii. **insert()** for your QUEUE class

Assume N elements in the queue. Briefly explain your answer. (2

Q3. (13)

a. Given the following Adjacency Matrix, draw unweighted directed graph labelling n numerically starting at 1. (4)

	1		1		1
		1		1	
1	1		1		1
				1	
1	1		1		
		1	1		

b. Perform a depth first search on the above graph. Clearly show the tree edges. Start the sea at node 1 and always select successors in numerically increasing order. What is maximum size of the stack during this DFS run? (4+2)
c. What is the maximum degree of any vertex in the graph? Does the graph have any direct cycles? (2+1)

Q4. (15)

a. Starting with an empty B-Tree of order 4, insert elements into the B-tree with the followi keys, showing the insertion at each step and each split operation: (5)
   55, 100, 90, 75, 10, 5, 15, 40, 65, 60, 50, 20, 30, 45.
b. Show the result of inserting 2,1,4,5,8,3,6,7 into an empty splay tree.[Show the tree at the e of each insertion] (4)
c. Show the result of deleting node 8 from the splay tree constructed in (b). (2)
d. Suppose we want to use an array to implement a stack. However, we do not know in advan the size to which the stack can grow. Indeed there may not be any apriori limit to the size the stack. Suppose we start with a 'small' array A of size say 10. Now as long as our arr bounds are not exceeded, all we need is a variable "top" that points to top of stack (so A[to is the next free cell). Each of PUSH and POP operations are O(1). However, when the arr is full and we need to push a new element on, we have a problem! In that case we c allocate a new larger array of double the size [say], copy the old one over and then go o from there. This is going to be an expensive operation, so a push that requires us to do th is going to cost a lot. But maybe we can "amortize" the cost over the previous chea operations that got us to this point. (4)

   Prove that with the above implementation of a stack, the cost of an arbitrary sequence of n stack[push/pop] operations to the stack costs at most O(n) and uses at most O(n) memory.