THE HONG KONG UNIVERSITY OF SCIENCE & TECHNOLOGY

Department of Computer Science and Engineering

COMP171: Data Structures and Algorithms

Spring 2008

Solution

Final Examination

Time: 28 May, 2008, 16:30 – 19:30

L1: Albert C. S. CHUNG

L2: Bo LI

L3: Zhen ZHOU

1. This is a closed-book, closed-notes examination.

2. Before starting, first check that you have all 18 pages (including this cover page).

3. Write your name and student ID on this page above.

4. Answer all questions in the space provided.

Student Name: _____ **Student ID:** _____

Email Address: _____ **Lecture Session:** _____

Question	Marks
1	/10
2	/10
3	/10
4	/15
5	/15
6	/15
7	/10
8	/15
Total	/100

Question 1 **Analysis of algorithms** **(10 marks)**

a) Rank the following twelve functions in their increasing order of growth rate.
Write them down on the lines provided below, where the slowest-growing function will be placed on the first line, and so on so forth. If two functions are having the same growth rate, then they should be placed on the same line.

2^n, $\frac{3}{8}n$, $2n^5+n^3$, $\log(n)$, $\log(n^3)$, $\log(\log n)$, e^{10}, $\frac{n}{\log n}$, $15n+60\log n$, $\ln(n)$, $n!$, $2^{\log n}$

Answer:

Solution:
e^{10}
$\log(\log n)$
$\ln(n)$, $\log(n)$, $\log(n^3)$
$\frac{n}{\log n}$
$15n+60\log n$, $2^{\log n}$, $\frac{3}{8}n$
$2n^5+n^3$
2^n
$n!$

b) Suppose $T(0) = T(1) = 1$ and $n \geq 2$. Which of the following recurrence function $T(.)$ CANNOT be upper-bounded by a polynomial function of n? (Note: there may be more than one choice. Each incorrect answer will receive deducted marks.)

(A). $T(n) = 3T(\lfloor \frac{n}{2} \rfloor) + n^2$
(B). $T(n) = 4T(\lfloor \frac{n}{2} \rfloor) + n$
(C). $T(n) = 3T(\lfloor \frac{7n}{8} \rfloor) + 8n + 1$
(D). $T(n) = 2T(n-6) + n$
(E). $T(n) = T(n-1) + n^2$

Answer:

Solution: (D).

c) Given the following function:

```
int Fun( int n )
{
     if((n==0) || (n==1)) { return 1; }

     else { return 2 * Fun(n-2) + 1; }
}
```

Derive the time complexity $T(n)$ of executing `Fun(n)` in Big−Θ form, assuming that n is a non-negative even number. Answer without derivation receives no mark.

Answer:

$$
\begin{aligned}
T(n) &= T(n-2) + c \\
&= T(n-4) + 2c \\
&: \\
&: \\
&= T(0) + (n/2)c \\
&= \Theta(n)
\end{aligned}
$$

Question 2 **Sorting** **(10 marks)**

a) Give the tightest bounds on the worst case running time of the following sorting algorithms in Big-O notation, assuming that the input size is n.

Algorithm	Running Time (worst)
Insertion Sort	$O(n^2)$
Merge Sort	O(nlogn)
Heap Sort	O(nlogn)
Quick Sort	$O(n^2)$

b) What is the minimum number of comparisons needed to guarantee to sort *any* 5 elements?

Answer:

ceiling($\log_2(5!)$) = 7

c) In a Quicksort implementation, suppose that we always select the left-most element of the array as the pivot.

i) In ONE sentence, state under which condition the best case of Quicksort occurs?

Answer:

Each partition is balanced. / Pivot selected is always median of sorting array.

ii) Analyze the time complexity of the best case of Quicksort. Show all steps.

Answer:

$$T(n)\begin{cases} O(1) & n=1 \\ 2T(\frac{n}{2})+O(n) & n>1 \end{cases}$$

Let c be a constant,

$$T(n)=2T(\frac{n}{2})+cn$$
$$=2[2T(\frac{n}{2^2})+\frac{cn}{2}]+cn$$
$$=\dots$$
$$=2^{\log_2 n}T(\frac{n}{2^{\log_2 n}})+(\log_2 n)cn$$
$$=nO(1)+cn\log_2 n$$
$$=O(n\log n)$$

Question 3 **Trees** **(10 marks)**

a) Let $G = (V, E)$ be an undirected graph. Prove that:
If G is a connected graph and $|E| = |V| - 1$, then G is tree.

(Hint: by definition, tree is a connected, acyclic, undirected graph.)

Proof:

G **is connected.** We want to prove that an acyclic graph G with n vertices and $n-1$ edges is connected. The proof is by induction on the number of vertices of G.

Base case: For $n=1$ and $n=2$ the claim holds, since in both cases, a graph with $n-1$ edges is connected.

Inductive step: Assume that the claim is true for all graphs up to size n. Consider an acyclic graph G with $n+1$ vertices and n edges. At least one of the vertices must have degree 1 (see note [1]). Now take that vertex of degree 1 and remove it, along with the associated edge. The graph G^- that remains has n vertices and $n-1$ edges and is connected according to our induction hypothesis.

b) Which statement(s) about AVL trees is(are) correct? There is only one correct choice.

I. The difference of the depth levels (from the root) of any two leaves is at most one.

II. During any insertion, if some rotation has been performed on an unbalanced node, then we need not do any rotation for its parent.

III. If the height of an AVL tree increases after an insertion, then it implies that before the insertion, all nodes on the path from the root to the leaf to which the newly added node is attached are completely balanced (i.e., height difference between left and right children is 0).

IV. Every heap is an AVL tree

A. Only II. B. I, II. C. II, III. D. I, II, III. E. I, III, IV. F. All.

Answer:

Answer: C

c) Let $N(h)$ be the minimum number of nodes in an AVL tree of height h, where $h = 0$ means a tree with a single node.

i) What are the values of $N(0)$, $N(1)$, $N(2)$, $N(3)$ and $N(4)$?

$N(0)$	$N(1)$	$N(2)$	$N(3)$	$N(4)$
1	2	4	7	12

2.5 marks

ii) Give a general recursive formula for calculating $N(h)$, together with the boundary conditions.

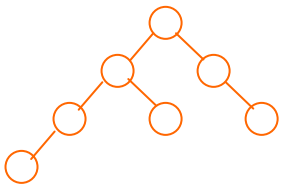
Answer:

$N(h) = 1 + N(h-1) + N(h-2)$ for h>=2

$N(0) = 1, N(1) = 2$

iii) Draw a tree with $h = 3$ and $N(3)$ nodes (no need to fill in node values).

Answer:



Question 4 **B$^+$ trees** **(15 marks)**

a) Given a B$^+$-tree with $M = 5$ and $L = 4$,

i) What is the minimum number of data stored in each leaf node?

Answer:

2

ii) What is the minimum number of children that an internal node may have?

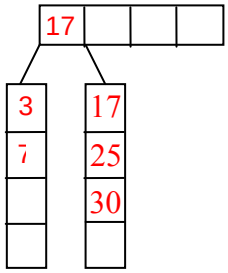
Answer:

3

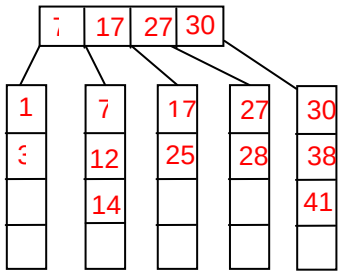
b) Insert the following number sequence into an empty B$^+$-tree with $M = 5$, $L = 4$, according to the given order. Apply the insertion procedure described in the lecture notes. Draw the intermediate B$^+$-trees after inserting 3, 38, 20, 10 in the following sequence:

7 30 17 25 3 41 14 12 27 1 28 38 15 29 23 6 20 21 10

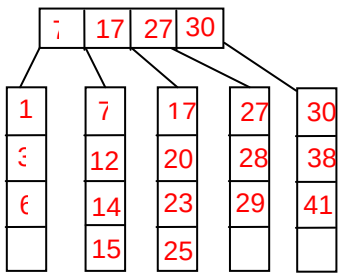
i) After inserting 3



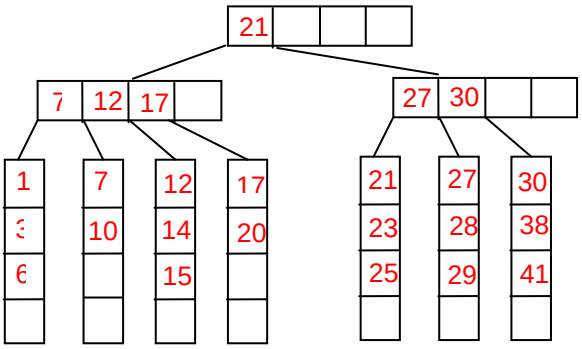
ii) After inserting 38



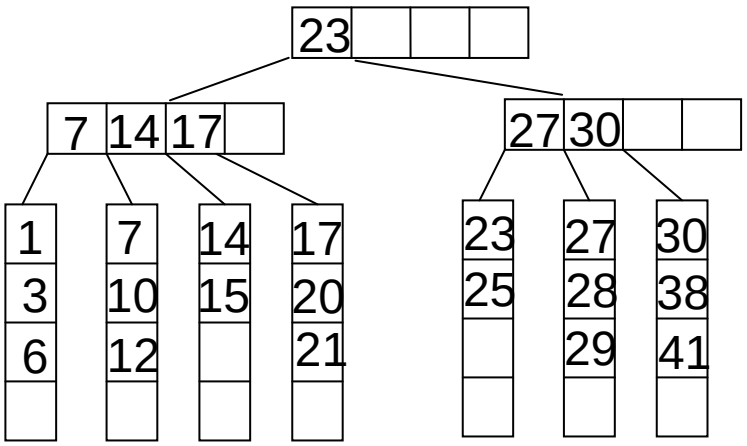
iii) After inserting 20



iv) After inserting 10

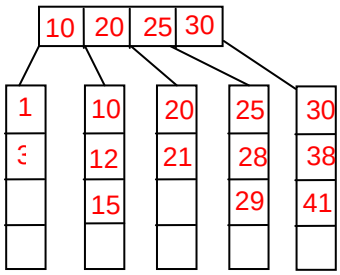


c) Given the above B^+-tree with $M = 5$ and $L = 4$.



Apply the deletion procedure described in the lecture notes to the B^+-tree above. Draw the resulting B^+-tree after deleting all the following numbers in the given order.

17 14 7 27 23 6



Question 5 **BFS and DFS** **(15 marks)**

a) Suppose graph *G* is a binary tree, which tree traversal method(s) will give the same order of visited vertices as running BFS (in lecture notes) on *G* starting from the root?

A. Preorder traversal
B. Postorder traversal
C. Inorder traversal
D. None of above

Answer:

Ans: D

b) Suppose graph *G* is a binary tree, which tree traversal method(s) will give the same order of visited vertices as running DFS (recursive version in lecture notes) on *G* starting from the root?

A. Preorder traversal
B. Postorder traversal
C. Inorder traversal
D. None of above

Answer:

Ans: A

c) Given the following pseudo-code on DFS algorithm.

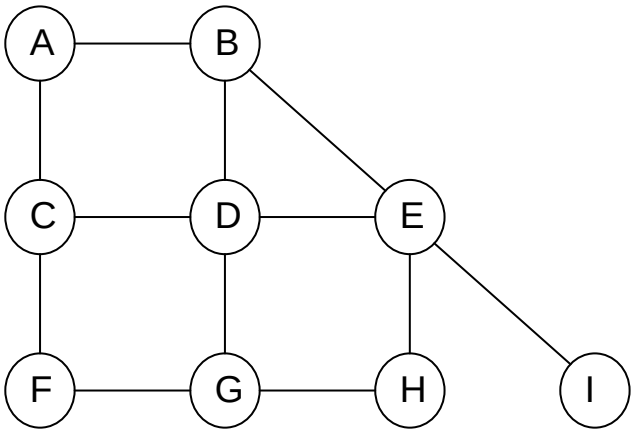
```
1    Boolean visited[V.size];

2    dfs(graph G, vertex x)
3    {
4      for each vertex u in V
5           visited[u] = false
6      list L = empty
7      search(x)
8      while(L nonempty)
9          remove w from end of L
10         if visited[w] == false
11         {
12             search(w)
13         }
14    }

15    search(vertex v)
16    {
17     print(v);
18     visited[v] = true
19     for each w in Adj[v]
20         add w to end of L
21    }
```

(Note: in line 19, *w* is picked according to the order of occurrence in the Adjacency List.)

Graph *G* :



Adjacency List :

A:	B	C		
B:	A	D	E	
C:	A	D	F	
D:	B	C	E	G
E:	B	D	H	I
F:	C	G		
G:	D	F	H	
H:	E	G		
I:	E			

Write down the printing output (i.e., output from line 17) after calling the `dfs(G,D)` function.

Answer:

Ans : D G H E I B A C F

Write down the list of vertices in *L* after each time `search(v)` is called.

Answer:

Vertices in L

1st time	B C E G	
2nd time	B C E D F H	
3rd time	B C E D F E G	
4th time	B C E D F B D H I	
5th time	B C E D F B D H E	
6th time	B C E D F A D E	
7th time	B C E D F B C	
8th time	B C E D F B A D F	
9th time	B C E D F B A D C G	

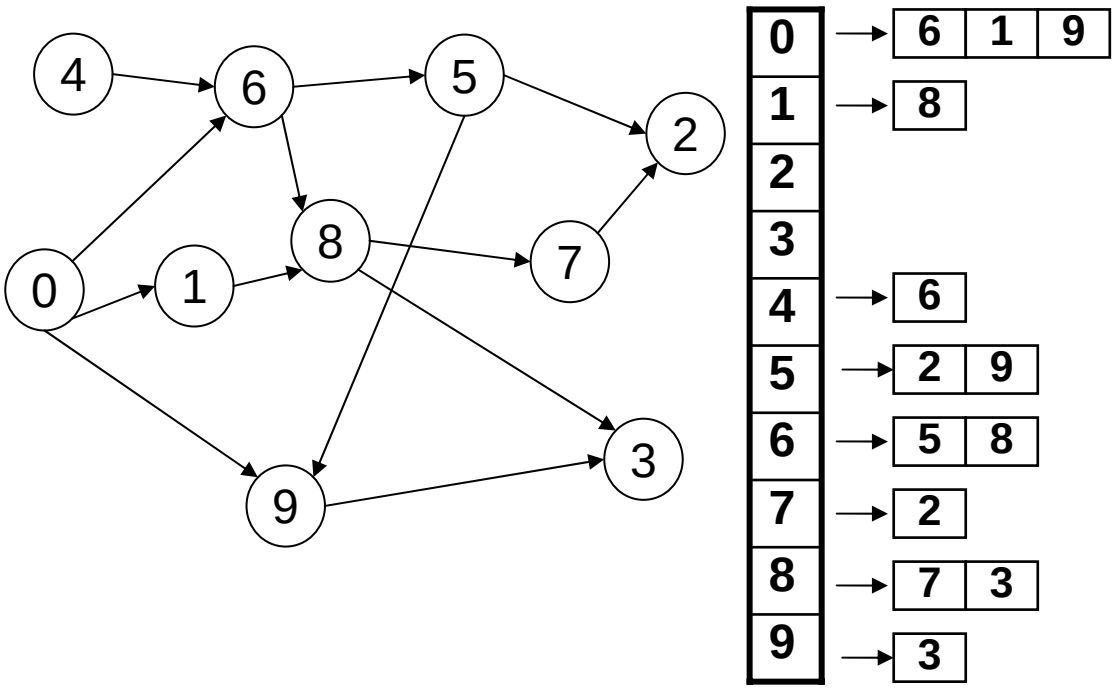
d) By changing one line of code in part c, the algorithm will become BFS. Write down the line number and show how to change the code.

Answer:

Ans: line 9, remove w from start of L or
line 20,add w to start of L

Question 6 **Topological sort** **(15 marks)**

Given the following graph *G* and its adjacent list representation:



Apply the Topological Sort algorithm described in the lecture notes, where an auxiliary data structure is used to store the neighboring vertices during the traversal.

a) Write down the topologically sorted list of the vertices using a queue (with enqueue and dequeue operations).

Answer:

0 4 1 6 5 8 9 7 3 2

b) Write down the topologically sorted list of the vertices using a stack instead (with push and pop operations).

Answer:

4 0 1 6 8 7 5 9 3 2

c) Write down the topologically sorted list of the vertices using a min-heap (with insert and deleteMin operations).

Answer:

0 1 4 6 5 8 7 2 9 3

d) By adding some condition checking code at the end of the Topological Sort algorithm described in the lecture notes, it is possible to determine whether the input directed graph has cycles or not.

Write down the complete pseudo-code for Topological Sort, with some extra code at the end, such that it returns TRUE if the graph is cyclic, and FALSE otherwise.
Assume that the indegree table has been given as an array and can be accessed by `indegree[v]`, where *v* is the vertex index of the directed graph.

Answer:

```
1. Initialize Q to be an empty queue;
2. For each vertex v
3.     do if indegree[v] = 0
4.           then enqueue(Q,v);
5. While Q is non-empty
6.     do v := dequeue(Q);
7.         output v;
8.         for each arc(v,w)
9.             do indegree[w] = indegree[w] -1;
10.                 if indegree[w] = 0
11.                     then enqueue(w);
12. for each vertex v
13.    do if indegree(v) > 0
14.           then return TRUE;
15. return FALSE;
```

Question 7 **Hashing** **(10 marks)**

a) Choose one correct answer from the five choices.

Hash tables can contribute to an efficient average-case solution for all of the following problems EXCEPT:

(i) Counting distinct values: given a set of n keys, determine the number of distinct key values.

(ii) Sorting data: printing of the entire table in sorted order in linear time.

(iii) Dynamic dictionary: support the operations of insert, delete, and search in a dictionary.

(iv) Symbol table lookup: given a program identifier, find its type and address.

(v) Range search: given values a and b, find all the records whose key values is in the range $[a, b]$.

(vi) Finding intersections: given two sets of keys, find all key values in common to both sets.

A. (vi) B. (iii) and (vi) C. (i) and (iv) D. (v) and (vi) E. (ii) and (v)

Answer:

Ans: E

b) Insert keys 67, 56, 77 one by one to the following hash tables by using *linear probing* to resolve collision, where the primary hash function is $h(k) = k \bmod 13$.

	Initial hash table	**After inserting 67**	**After inserting 56**	**After inserting 77**
0	26	26	26	26
1	14	14	14	14
2		67	67	67
3				77
4	17	17	17	17
5	44	44	44	44
6			56	56
7	33	33	33	33
8				
9				
10				
11				
12	25	25	25	25

c) Insert keys 67, 56, 77 one by one to the following hash tables by using *double hashing* to resolve collision.

The primary hash function is $h(k) = k \bmod 13$,

the secondary hash function is $h_2(k) = 11 - (k \bmod 11)$,

and the collision resolution strategy is $f(i, k) = i * h_2(k)$.

	Initial hash table	**After inserting 67**	**After inserting 56**	**After inserting 77**
0	26	26	26	26
1	14	14	14	14
2		67	67	67
3				
4	17	17	17	17
5	44	44	44	44
6				
7	33	33	33	33
8				
9				
10				77
11			56	56
12	25	25	25	25

d) State one advantage for linear probing

Answer:

Linear probing:

- easily implemented

e) State one advantage for double hashing

Answer:

Double hashing:

- eliminates the problem of clustering

Question 8 **Pattern matching** **(15 marks)**

a) What is the worst case time complexity of the Naïve, Rabin-Karp (RK) and Boyer-Moore (BM) algorithm, given *m* the length of pattern and *n* the length of the text?

Algorithm	*Naïve (Brute force)*	*Rabin-Karp*	*Boyer-Moore*
Worst-case time complexity	O(mn)	O(mn)	O(mn)

O((n-m+1)m) is also acceptable. O(mn+A) for BM is also correct.

b) Suppose the pattern is

1	6	3	4	7	9

and the text is

9	2	0	5	3	2	5	1	6	3	4	7	9	1	2	3	9	1	2	7	3

By RK algorithm, we use the following function to transform the pattern into an integer *p*.

$$p = P[m-1]+2*(P[m-2]+2*(P[m-3]+\ldots 2*(P[1]+2*P[0])\ldots).$$

And the prime number *q* that we choose for modulo arithmetic is 11.

Describe step by step how RK algorithm works for this pattern matching problem.

Answer:

Hash(163479)=4

Hash(920532)=7

Hash(205325)=4

…… =2

=10

=0

=10

=9

=4

Character comparison for each equality in the hash value.

c) Suppose the pattern is

AT-THAT

and the text is

WHICH-FINALLY-HALTS . - -AT-THAT-POINT

Describe step by step how BM algorithm (described in lecture notes) works for this pattern matching problem, for example the first step should be the following:

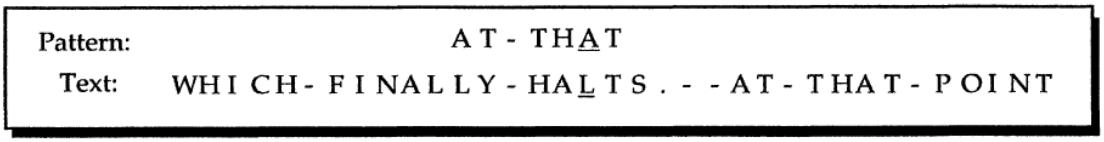
Please underline the mismatched characters.

Note: the BM algorithm halts after it finds the first successful matching.

Answer:

(6pts) Step 2

Step 3



Step 4

AT-THAT

WHICH-FINALLY-HALTS . - -AT-THAT-POINT

Step 5

AT-THAT

WHICH-FINALLY-HALTS . - -AT-THAT-POINT

Step 6

If one of the step is wrong, the following steps are all considered wrong. Please be noticed that there is a character '.' After 'HALTS'.

This page is for rough work ONLY.
No marks will be given for anything written here.

This page is for rough work ONLY.
No marks will be given for anything written here.