March 01 2006
Duration : 75 min - Total : 25pts

Exercise 1 : Algorithm Analysis [6pts]

1. Give an analysis of the running time (Big-Oh notation) for each of the following 4 program fragments. Note that the running time corresponds here to the number of times the operation `sum++` is executed. `sqrt` is the function that returns the square root of a given number.

   (a)
   ```
   sum = 0;
   for(i=0;i<sqrt(n)/2;i++)
    sum++;
   for(j=0 ;j<sqrt(n)/4;j++)
    sum++;
   for(k=0;k<8+j;k++)
    sum++;
   ```

   (b)
   ```
   sum = 0;
   for(i=0;i<sqrt(n)/2;i++)
     for(j=i;8+i;j++)
      for(k=j;k<8+j;k++)
       sum++;
   ```

   (c)
   ```
   sum = 0;
   for(i=1;i<2*n;i++)
    for(j=1;j<i*i;j++)
      for(k=1;k<j;k++)
        if (j % i == 1)
          sum++;
   ```

   (d)
   ```
   sum = 0;
   for(i=1;i<2*n;i++)
    for(j=1;j<i*i;j++)
      for(k=1;k<j;k++)
        if (j % i)
          sum++;
   ```

2. If it takes 10ms to run program (b) for n=100, how long will it take to run for n=400 ?
3. If it takes 10ms to run program (a) for n=100, how large a problem can be solved in 40ms ?

Solution

1. **(a) and (b)** $O(\sqrt{N})$**, (c)** $O(N^4)$ **and (d)** $O(N^5)$
2. **20ms**
3. **1600**

Exercise 2 : B-Trees [6pts]

Given a B-Tree of order `M` and containing `N` data items, what is the worst and best case number of disk access (big-Oh notation) of the following operations :

1. Find a given data item.
2. Insert a given data item.
3. Delete a given data item.

Solution

1. Worst case : $O(\log_{\frac{M}{2}} N)$**. Best case :** $O(\log_M N)$**.**

Exercise 3 : Binary Search Trees [0.5x6+0.5x3+0.5x3+2+1 = 9pts]

1. Given a Binary Search Tree, *BST*, of height `H`. In terms of `H`, what is the complexity in time (big-Oh notation) of the following operations.
   (a) FindMin.
   (b) FindMax.
   (c) Insert a new element.
   (d) Delete a leaf node.
   (e) Delete a one child node.
   (f) Delete a two children node.
2. Assuming that *BST* is a perfect binary tree (each non leaf node has exactly two children). In terms of `H`, what is the complexity in time (big-Oh notation) of the following operations.
   (a) Printing *BST* using the in order traversal.
   (b) Printing *BST* using the post order traversal.
   (c) Printing *BST* using the pre order traversal.
3. Redo the previous question if we assume that BST is a degenerated binary tree (has exactly one branch or in other words every non leaf node has exactly one child).
4. Describes briefly the method that prevents a BST from degenerating into a linear structure. Explain why there is no need of such method for binary Heaps.
5. Which of the above traversal algorithms allows you to print the elements of BST in increasing order ?

Solution

1. $O(H)$

2. $O(2^H)$

3. $O(H)$

4. **AVL method. A heap is a** *complete binary tree* **(it is completely filled with the exeption of the bottom level) since it adds elements in a breath first manner (top down and from left to right for each level).**

5. **Inorder traversal**

Exercise 4 : Priority Queues [4x1=4pts]

`Insert` and `DeleteMin` are two fundamental operations of priority queues. What is the worst case time cost of these two operations in the case where the priority queue is implemented using :

1. a linked list
2. a sorted linked list
3. a binary search tree
4. a binary heap

Solution

1. **Insert :** $O(1)$**, DeleteMin :** $O(N)$**.**
2. **Insert :** $O(N)$**, DeleteMin :** $O(1)$**.**
3. **Insert and DeleteMin :** $O(H)$ **such that** H **can be** $\log N$ **or** N **(in the worst case : degenerated BST).**
4. **Insert and DeleteMin :** $O(\log N)$**.**