UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

COS 212 Tutorial 5: Version A

- 23/03/2012
- 50minutes
- 2 questions for a total of 27 marks.

Name:_____

Student/staff Nr:_____

Marker (office use):_____

Question 1 Heaps ..... (13 marks)

1.1 Consider an array representation for both heaps and binary search trees. As for heaps, in binary search trees the same calculations can be used to find children or parents. With this representation, what advantage does the heap structure property yield that cannot be guaranteed for binary search trees. (2)

Answer:

> **Solution:** Array is compact, no open holes, space saved, etc.

1.2 Considering a max-heap, why is searching for any element other that the largest an $O(n)$ operation? (2)

Answer:

Solution: No assumption can be made of the relation between elements in sibling subtrees.

OR

Duplicate elements may exist.

1.3 Why is a linked list an inappropriate structure to use when implementing heaps? (2)

Answer:

Solution: Finding children, parents, last position etc. is an order N operation, have to traverse the entire list the whole time.

1.4 Consider the given max-heap and answer the questions that follow:

`[100,52,51,20,31,23,20,1,13,16,30,1,22,1,15]`

a) Perform two consecutive `heap dequeue` operations on this heap and only give the final `array` representation. Answer: (4)

Solution: Correct element in the root

Last two leaf nodes missing

15 is where it is supposed to be

1 (the original **last** 1 in the list)is where it is supposed to be.

b) Assuming the originally given heap (unchanged by the previous question), add the element 200 to the heap. Only draw the final `array` representation of the heap after this insertion. (2)

Answer:

Solution: 200 is the root

All elements in their correct position i.e. 100 is left of 200, 52 is left of 100, 20 is left of 52, 1 is left of 20

1.5 Heaps need not be binary. In fact, heaps may be created where each node is allowed any number of children decided on beforehand. The general case for heaps are called `d-Heaps` where `d` indicates the maximum number (1)

of children each node is allowed to have. A binary heap can then in essence be referred to as a 2-Heap. What is the formula for calculating the index of the sixth child for a node at index `i` in a 10-heap?

Answer:

Solution: $10i + 6$

Question 2 B-Trees ..... (14 marks)

2.1 Not taking maximal block usage into account, the value of `M` in B-Trees needs to be chosen carefully. There are values for `M` that will cause errors in B-Tree operations. Which values for `M`, $M > 0$, are inappropriate? Also list and briefly discuss two problems that arise from an incorrectly chosen `M` (3)

Solution: 1 Mark for M cannot be even (or cannot be 1, seeing as the questions has $M > 0$)

1 Mark for each of the following problems:

- For splits, one of the siblings will always underflow if keys are divided.
- There will be one more or one less child (depending on split or merge) that cannot be incorporated into the tree.

Answer:

2.2 Insert the following elements, in the given order, into an initially empty B-Tree with $M = 7$. For your answer give only the final tree. (5)

`1,2,3,4,5,6,7,8,50,49,48,47,46,45,44,43`

Solution: 1 mark tree is of height 2

1 mark, root is 4,8,47

1 mark for first leaf 1,2,3

1 mark second leaf 5,6,7

1 mark third leaf 43,44,45,46

//If the above holds the last leaf will be correct anyway

Answer:

2.3 Assume the following partially given `BTreeNode` class:

```
class BTreeNode<T extends Comparable<? super T>>
{
        BTreeNode(T firstKey)
        {
                keys[0] = firstKey;
                for(int i = 0; i < M; ++i)
                        children[i] = null;
                numberOfKeys = 1;
        }

        int numberOfKeys;
        Comparable<T> keys[] = new Comparable[M-1];
        BTreeNode<T> children = new BTreeNode<T>[M];
}
```

a) Without making any assumptions about any additional methods and fields, how can one test if a `BTreeNode` is a leaf node? (1)

> **Solution:** Test if first element in children is null.

Answer:

b) Write a recursive method with the signature `void ascending(BTreeNode<T> n)` which will output the keys in a B-Tree in ascending order. (5)

> **Solution:** check null base case for n
>
> loop through keys
>
> do child before key first (i.e. do child[i]...).
>
> then key output (...and then keys[i])
>
> take care of last child (i.e. child [numKeys+1]).

Answer: