Jan 5, 2017

1 Course Overview

Abstract Data Type (ADT) a set of objects and operations on them.

Example. Object: integers; Operations: +, -, x, /

Example. Object: sequences of other objects; Operations: PUSH, POP, ISEMPTY along with description and effects of each operation

Remark. ADT describes **what** are the objects. Data structure describes **how** ADT's are implemented.

- array is an implementation of stack
- linked list implementation of stack

Complexity Analysis

Remark. Used to compare different data structures. There are several categories of complexity, i.e. running time complexity and space complexity.

Time Complexity

Remark. Represents the *abstract running time*, independent of hardware, programming language, etc. Time complexity is a function with the number of steps depending on the size of input. (Note that size is measured informally here as size of input. The true measure of input size would involve the number of bits stored in memory)

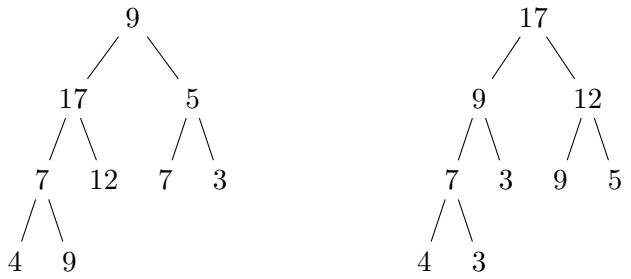
Example. LinkedSearch(L, k)

Algorithm 1: LinkedSearch

```
1 Function LinkedSearch (L, k)
      Input: A is an array of objects
      Output: Returns the index of k in L
2     z = l.head
3     while z ≠ None ∧ z.key ≠ k do
4         z = z.next
5     return z
```

This algorithm runs in $\mathcal{O}(n)$. For any input n, the number of steps taken depends on where k is in L. We are primarily intersted in the worst case scenario. We can also calculate the average running time.

Remark. Difference between worst-case / best-case and upper-bound $\mathcal{O}$ / lower-bound Ω.
Asymmetry of bounds Proving upper bound on the worst case tells us that no instance regardless of input takes more step than the upper bound. Proving lower bound on the worst case tells us that at least one instance regardless of input takes longer time than the lower bound in the worst case. This is flipped for best-case.



A regular full binary tree and a heap

Definition. ***Priority Queue*** *(ADT)*
A queue (ADT) where each element has a "priority". Operations include

- Insert(S, x)*: adds x to priority queue S. Assume that x.priority.*
- Max(S)*: return an item with maximum priority in priority queue S*
- ExtractMax(S)*: removes and return an item with maximum priority from S*

Remark. Items with higher priority come out first.

Possible data structure

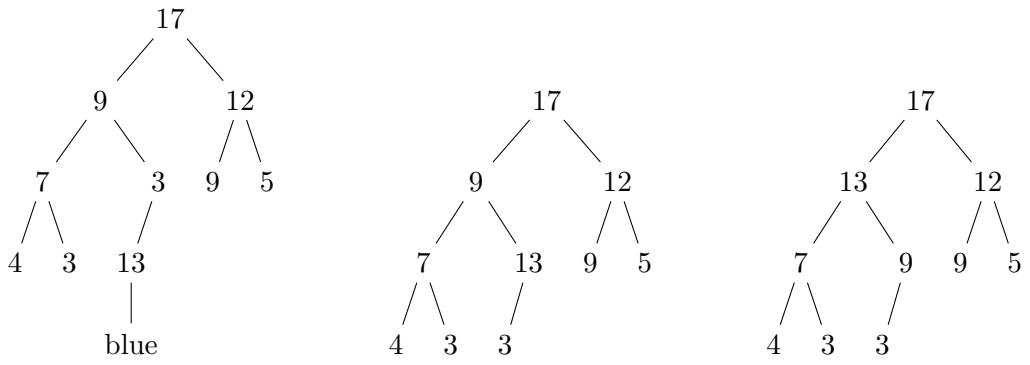
- Unsorted array. Insert(S, x) takes $\mathcal{O}(1)$; However Max(S) and ExtractMax(S) takes $\mathcal{O}(n)$.
- Sorted array. Max(S) and ExtractMax(S) takes $\mathcal{O}(\log n)$ with binary search algorithm; However Insert(S, x) takes $\mathcal{O}(n)$ since we have to shift all subsequent item in array.

Definition. ***Heap*** *(data structure) is a full binary tree following heap order, The heap order indicates that each node's priority is greater than or equal to priorities of its children.*

Remark. Every path from root to leave is in decreasing order. Note that however

Definition. *A full binary tree is a tree in which every node other than the leaves has two children.*

- INSERT(S, x): Insertion has to maintain both shape of the tree and order. Since shape is difficult to maintain. We add x to the correct leaf position for a full tree. Then we swap x's position with its parent as long as x has a higher priority than its parent. We call it *bubble up* the new value to restore order. This procedure takes $\Theta(\log n)$



- MAX(S): takes $\mathcal{O}(1)$ because the root element has the highest priority.
- EXTRACTMAX(S): We first remove the root of the tree and move the rightmost element at the bottom-most level to the root. Now we then swap the element with the child with the larger priority until order is restored. This procedure takes $\Theta(\log n)$

2 Binomial Heaps

Definition. ***Mergeable Heaps*** *(ADT) collection of items with*

UNION(H_1, H_2):