Algorithm & Data Structures

June 9, 2018

1 Array and adhoc techniques

1.1 Window Sliding technique

This technique shows how a nested loop in a few problems can be converted to a single loop and hence reducing complexity, only requirement being usually the array elements need to have total order(each element is comparable to each other element).

e.g. Given an array of integers of size n. Our aim is to calculate the maximum sum of k consecutive elements in the array.

2 Stacks and Queues

Even though stacks(LIFO) and queue(FIFO) have opposite rules, a stack can be implemented using two queues and a queue can be implemented using two stacks.

3 Sorts

3.1 QuickSort

There are two kinds of implementation of partition in quicksort, One which moves both the first and the last pointer, and the other which only moves the first pointer up to the end(easier to understand). Here is the example which moves only a single pointer.

```
void qsort(int l, int u) {
   int i, m;
   if (l >= u)
      return;
   m = l; // invariant: m always represents value < or equal to l
   for (i = l + 1; i <= u; i++) {// i is the only moving ptr
      if (x[i] < x[l]) // compare running ptr with pivot value
         swap(++m, i); // move m, then swap contents a[m] <-> a[i]
   }
   swap(l, m); // m is where the pivot value is supposed to go
   qsort(l, m - 1);
   qsort(m + 1, u);
}
```

4 Binary Heap

Stored in an array.

Invariant We study a max-heap, i.e parent's keys are always larger then children keys.

keep index 0 empty.
So arr[1] is largest element i.e root

Child index: child of node n is n*2, n*2 + 1
Parent index: parent of node n is n/2

Sink op: keep swapping values with children until settled.
Swim op: keep swapping values with parent until settled.

Public Operations - creation:

- **heapify**: Create heap out of array elements
- **union/merge**: mix existing heaps to create a new valid heap

Public Operations - manipulation & inspection:

- **extract-max/pop** - remove and return max value from the heap.
- **find-max/peek** - just get the max value on the heap without modifying it.
- **insert/push** - add a new item to the heap.
- **size** - returns number of items in the heap.

Internal Operations:

- **decrease-key/increase-key** change value of a particular node in heap
- **trickle-up/sift-up/swim** bring a node upwards, until heap property is satisfied
- **trickle-down/sift-down/sink** bring a node downwards until heap property is satisfied.

5 Binary Tree

For a given Binary Search Tree, only pre-order traversal(first root, then leftchild recursively, then right child recursively), can preserve the structure of the tree. Hence it is used in serialization, and preserve tree information.

The proof of above case is interesting, since it derives from the general fact that you would serialize/store a structure of graph using an adjacency list, which is assentially the node followed by list of stuff it points to.

5.1 Binary Search Trees

Binary Search Tree have a special invariant, i.e all the items on the left subtree are smaller than current item, and all the items on the right subtree are larger than the current item. In a sense, we assume there is a total order possible for all the elements added to the BST. Also by induction, the left and right subtrees are also binary search trees. Following regular operations without modifying tree height or any facility for balancing tree to improve its performance

- **insert**: find the place to insert by using comparisions and add it.
- **search/find**: take left and right guided by comparisions from the root till you reach the desired node.
- **delete**: 3 cases.
    1. **Leaf Node**: Just remove node
    2. **One child**: remove node and make parent point directly to grand child.
    3. **Two children**: remove node and replace by successor or predecessor

5.1.1 Balanced Binary Search Trees

There are usually some techniques involved in manipulation of binary search trees, to maintain balancing invariant (e.g height balancing or weight balancing), in order to prevent greater skewing of tree shape and directly or indirectly affecting complexity of opearations in Binary search trees.

5.1.2 AVL Tree - Height Balanced Binary search Tree

Height is an important property held by every node in such a tree. The definition of height of a node is inductive:

- Height of a null node is 0 $H_{null} = 0$
- Height of a node is $H = max(H_l, H_r) + 1$

Also the height of a node is only affected by the height of children subtrees, and has no bearing on parent nodes.

Balance Factor of a node : The balance factor is defined as the height difference between the child nodes. $B = |H_r - H_l|$

Core Invariant BalanceFactor is at most 1 : $|H_l - H_r| \leq 1$

How different operations affect balance factor of nodes?

- **search/find** The search process is same as that of a regular binary search tree, and since it does not modify the tree, there is no effect of search operation on balance factor of nodes.

- **insert** The insert process is done by first doing the search process, followed by inserting at the desired position. Insertion of node may result in an balance factor change. So to maintain the invariant of balance factor, rotations are performed on unbalanced node to rebalance the tree.
- **delete** Similar role as insert.

6 Graph

The vertices be numbered from 1 to N, for the reason of simply storing mapping in vector/array since keys are number.

For quick and easy represntation of graphs, use a VxV matrix, in which item (i,j) is 1, if there is a connection from i to j. Space complxity is $O(|V^2|)$

In practice, the graphs are really large and sparse, so one needs to use an adjacency list representation. In this represntation, each vertex stores list of its adjacent vertices. Complexity is $O(|V| + |E|)$

One of the most important concepts that most graph algorithms rely on is that of a cut/fringe. A cut essentially partitions graph into two sets of vertices, and has a bunch of crossing edges along the cut

A crossing edge is an edge that goes from one side of the cut to the other.

6.1 Minimum Spanning Trees

Cayley's formula: The number of labeled trees of n vertices is n^{n-2}

Simple MST algorithms like kruskal and prim's algorithm assume that you are working on a undirected weighted graph.

Main Invariant: Given any cut, the minimum weight crossing edge is always in MST.

7 Strings

A string is a sequence of charachters.

7.1 SubStrings

A substring is a list of consecutive charachters in a given string that has a start and an end index, e.g. $A[i..j]$. For a given string of length N, the full string $A[1..N]$ is also a substring of A.

No of Substrings All possible combinations of start indexes with end indexes would be $n + (n-1) + (n-2) + \cdots + 1$ i.e. $\frac{n(n-1)}{2}$.

7.2 SubSequence

A subsequence is a list of charachters picked from the original string preserving the relative order, but not necessarily consecutive charachters. one can list subsequence of original string using a list of indexes. e.g. $A[1, 2, 4, 6, 7]("abcdefghi") =$ "$abdfg$" is a valid subsequence.