Algorithm & Data Structures

April 7, 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.

Insertion: Insertion usually happens at last index N, followed by a swim op.

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.

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

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.