Quiz 2 Solution

Question-1 Discuss time complexity of the quick sort algorithm.

Solution

Quicksort means first pick the random element then fix its position in sorted array, then divide array as making it middle point. Then pick another random element and then fix its position sorted subarray and so on. Some possible cases in fixing picked points are:

1. If all picked points placed in last or start of the array or subarray. In this case sorted elements (picking random elements and then placing in sorted array or subarray), will increment by one, in every pass of remaining elements (n-1, n-2, n-3, n-4……...).
   Worst case: n^2
2. If all picked points placed in ½ of array or subarray. Beauty of this case i3, sorted elements (picking random elements and then placing in sorted array or subarray), will increase two time, in every pass of remaining elements (n-1, n-3, n-7, n-15……...).
   Best case: n (logn)
3. If all picked points placed in ¾ or ¼ of the array or subarray.
   Average case: n (logn)

Marking Scheme:

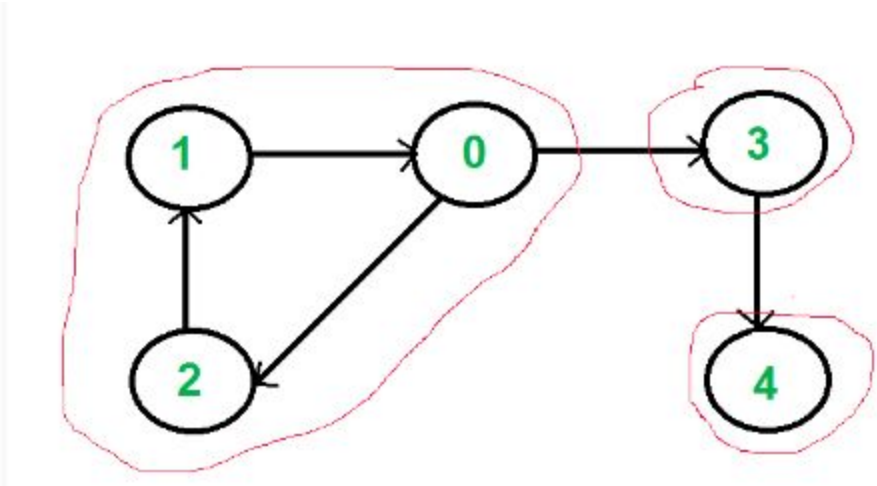
If discuss all cases of Quicksort (best, worst, average) [3]
If discuss any two cases [2]
If only complexity without discussion [1]
If Nothing or wrong [0]

Q2 - Define Strongly Connected Components(SCC). Write a note on the correctness of Kosaraju's algorithm for finding (SCC).

Solution

SCC

Let G = (V, E) be a directed graph. A strongly connected component (or SCC) of G is a set C ⊆ V such that C is not empty. For any u, v ∈ C: u and v are strongly connected. For any u ∈ C and v ∈ V – C: u and v are not strongly connected.



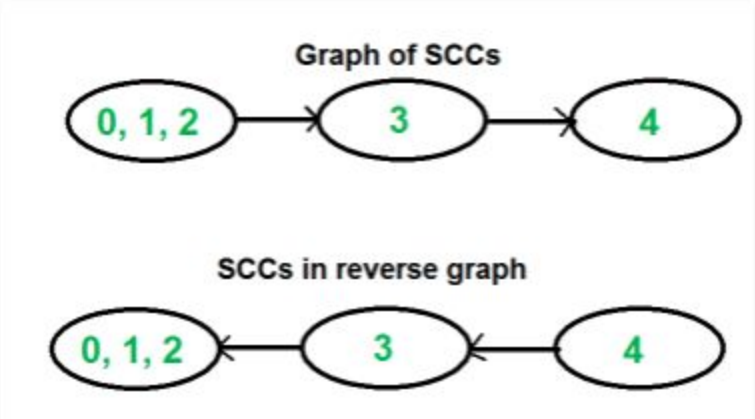


Image source : geeksforgeeks.org

Intuition behind the proof of Correctness of Kosaraju's algorithm for finding SCC.

The condensation of a directed graph G is the directed graph G_{SCC} whose nodes are the SCCs of G and whose edges are defined as follows: (C_1, C_2) is an edge in G_{SCC} iff $\exists u \in C_1, v \in C_2$. (u, v) is an edge in G.

G_{SCC} is a DAG for any graph G.

Also, the last nodes in each SCC will be returned in reverse topological order. Each time we do a DFS in the reverse graph starting from some node, we only reach nodes in the same SCC or in ancestor SCCs. Since we process the SCCs in topological order, at each point the only unvisited nodes reachable are nodes in the same SCC.

A more involved proof : (source : CLRS)

We argue by induction on the number of depth-first trees found in the depth-first search of G^T, the vertices of each tree form a strongly connected component. The inductive hypothesis is that the first k trees produced are strongly connected components. The basis for the induction, when k = 0, is trivial.

In the inductive step, we assume that each of the first k depth-first trees produced is a strongly connected component, and we consider the (k + 1)st tree produced. Let the root of this tree be

vertex u, and let u be in strongly connected component C. Because of how we choose roots in the depth-first search, f [u] = f (C) > f (C) for any strongly connected component C other than C that has yet to be visited. By the inductive hypothesis, at the time that the search visits u, all other vertices of C are white. By the white-path theorem, therefore, all
other vertices of C are descendants of u in its depth-first tree. Moreover, any edges in G^T that leave C must be to strongly connected components that have already been visited. Thus, no vertex in any strongly connected component other than C will be a descendant of u during the depth-first search of G^T . Thus, the vertices of the depth-first tree in G^T that is rooted at u form exactly one strongly connected component, which completes the inductive step and the proof.

Marking Scheme:
Definition of SCC : 2 marks
Proof of Kosaraju : 5 marks

Q-3 Ford fulkerson algorithm should be used.

How to create graph:-Create bipartite graph of job applicants ,jobs and add edges from job applicants to the jobs in which they are interested in.Now add two more vertices source and sink in which source is connected with all the vertices of job applicants and all the vertices for jobs are connected to sink.Now find max flow for the graph using ford fulkerson algorithm.

Pseudocode:-

```
initialize flow to 0

path = findAugmentingPath(G, s, t)

while path exists:

    augment flow along path

    G_f = createResidualGraph()

    path = findAugmentingPath(G_f, s, t)

return flow

Time Complexity:- O(E*f) where f is the max flow

                  O(E*V^2)
```

Resource:-https://www.geeksforgeeks.org/maximum-bipartite-matching/

Rubric:-Complete Algorithm(5 marks)

Time complexity(2 marks)

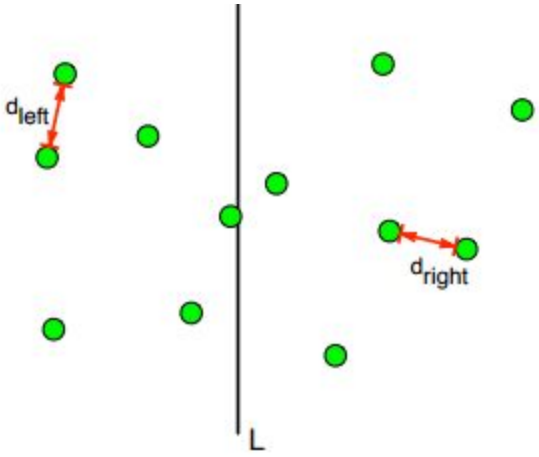
Question 4 : Discuss an O(n) conquer algorithm for finding the closest pair of points. Prove its correctness.
Solution :
Algorithm is a follows :

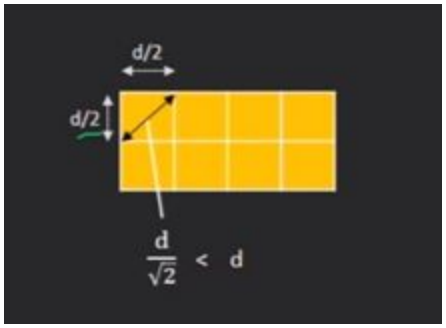
1. Sort points according to their x-coordinates.
2. Split the set of points into two equal-sized subsets by a vertical line x=xmid.
3. Solve the problem recursively in the left and right subsets. This yields the left-side and right-side minimum distances dLmin and dRmin, respectively.
4. Find the minimal distance dLRmin among the set of pairs of points in which one point lies on the left of the dividing vertical and the other point lies to the right.
5. The final answer is the minimum among dLmin, dRmin, and dLRmin.

It turns out that step 4 may be accomplished in linear time. Again, a naive approach would require the calculation of distances for all left-right pairs, i.e., in quadratic time. The key observation is based on the following sparsity property of the point set. We already know that the closest pair of points is no further apart than dist= min(dLmin, dRmin). Therefore, for each point p to the left of the dividing line we have to compare the distances to the points that lie in the rectangle of dimensions (dist, 2 · dist) to the right of the dividing line.And what is more, this rectangle can contain at most six points with pairwise distances at least dRmin. Therefore, it is sufficient to compute at most 6n left-right distances in step 4.

The complexity is thus O(7*n) = O(n).

Correctness : When we form a grid of d x 2d in the middle there cannot be more than 6 elements to be compared with as each grid takes one point atmost . The cell of grid is of length d/2 x d/2 as it's length and breadth. The diagonal is the longest which is d/sqrt(2) < d. And we know d is the smallest distance as computed. Thus maximum 6 comparisons and thus algorithm is correct.





Rubric : Mentioning the complete algorithm with time complexity analysis - 3 marks

Correctly explaining why there cannot be more than one point in a cell of the grid as d/sqrt() < d with the diagram - 3 marks

Q5) Use Convex Hull and apply any of its algorithm like QuickHull or Graham Scan. For correctness, try to prove by using intersection property of line with cops and robbers and points inside the triangle formed.

Rubric :-

3 marks for convex hull

3 marks for full correctness

2 marks for other method and 2 marks for other correctness