Problem 1 (0.6 points)

Given a graph where each node has a unique integer id between 0 and numNodes - 1, and all edges are of equal weight, find the shortest path between two nodes, start and dest, in O(n) time. Each node has a maximum number of connections k that does not scale with the number of vertices n. You may write psudocode or a detailed description.

Solution

Think of the graph as a tree with the start node as the root, and apply a breadth-first traversal until the destination is reached. Since we visit each node at most once, the complexity is O(n). Note that we don't even need the assumption that k doesn't scale with n.

Problem 2 (0.7 points)

Non_Recursive_QuickSort(A, Pivot, hi, lo) //performs 1 iteration of pivot sort on an array for a given pivot, and returns the correct index of the pivot.

```
swap A[hi] with A[index of Pivot]
pivot := A[hi]
i := lo       // place for swapping
for j := lo to hi – 1 do
   if A[j] ≤ pivot then
      swap A[i] with A[j]
      i := i + 1
swap A[i] with A[hi]
return i //returns correct index of pivot
```

```
Bad_bogo(A)
  for 1 to 100 do
        shuffle A
  return
```

Array Nums //an array of all integers 1 to 100

Using the above functions and predefined array implement a function to generate a random integer between two bounds min and max exclusively.
Given the limits min and max will always be within 1 to 100.

int Randint(min,max, Nums) //returns a random integer. Nums is given array above.

Solution

```
int Randint(min,Max, Nums)
        Bad_bogo(Nums)
        Non_Recursive_QuickSort(Nums, Min, Nums.length()/or 100, 0)
        Non_Recursive_QuickSort(Nums, Max, Nums.length()/or 100, 0/or Min)
        return Nums[(Min + Max)/2]
```

Problem 3 (0.7 points)

Proof of Prim's Algorithm by induction

Base case: Consider graph M with m vertices and graph N with n vertices. m = 1, n = 1. A graph with only 1 vertex is a minimum spanning tree of itself, so we have 2 minimum spanning trees that span graphs of m vertices and n vertices.

Inductive Step: Assume we found the minimum spanning trees for each of graph M and graph N.

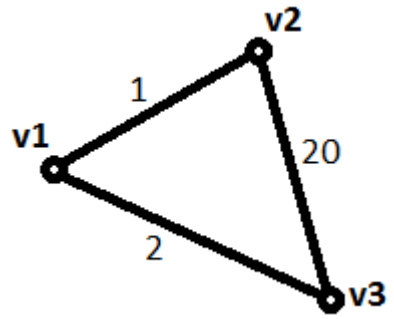
Conclusion: If we choose the edge with least weight that connects a vertex in graph M to a vertex in graph N, this edge has to be in the minimum spanning tree of a graph that consists of all the m+n vertices. Because we have already found the minimum spanning trees for each of the two subgraphs, adding this edge connects the two minimum spanning trees and thus form the minimum spanning tree for the new graph of vertices m+n.

When finding the minimum spanning tree with Prim's Algorithm, the vertices already connected form the set of m vertices and the new vertex to be added forms a graph of n=1 vertex. Hence, Prim's Algorithm is proved.

Is the proof right or wrong? If it is wrong, what is the mistake in the proof?

Solution

It is wrong. Adding an edge with least weight to connect minimum spanning trees of m vertices and n vertices does not guarantee to form a minimum spanning tree of m+n vertices. Consider this case:



If v1 is in Graph M and v2 and v3 are in Graph N, with this algorithm, a spanning tree will be found to have edges v1-v2 and v2-v3; however, the minimum spanning tree of all three vertices is formed by choosing edges v1-v2 and v1-v3.