PSO #7 Solutions Sketch (Week 9)

Week of 2021-10-18

1 Greedy Proof

1. Consider m conference rooms with capacities $D_1, D_2, \ldots, D_m$. You receive n requests for rooms of size $k_1, k_2, \ldots, k_n$. Once a room is allocated additional requests cannot be put there.

   Develop a greedy algorithm to handle the requests and prove its correctness and runtime.

   Solution: Sort the requests and room capacities and then simply start allocating requests based largest capacity room to largest request and keep going from there.

   Proof is to construct a basic swap argument, i.e., if there is a different assignment of disks that is known to be optimal, then at the first point at which the two assignments differ one can recognize that the optimal assignment has less of a capacity remaining in the disk that was assigned the memory request than the greedy, and will continue to have less capacity remaining as an aggregate for all requests going forward.

   Runtime is $O(\max(m, n) \log \max(m, n))$, as we are sorting both lists and then traversing through them each once.

2. Consider a graph G. Devise a scheme to assign colors to the nodes of G s.t. no node has the same color as a node adjacent to it (Graph coloring heuristic). You are given a list of colors to use that is guaranteed to be in length at least greater than or equal to one more than the maximum degree of any vertex in G, as well as a corresponding "dictionary" structure which lists the index of the color in the list.

   Solution: Take a node, and give it a color. Then, using either a BFS or DFS strategy, traverse the rest of the graph, and for each node, give it the first color in the list that does not appear among the nodes it is adjacent to. This will take the same runtime as the BFS or DFS strategy that the user chooses.

2 Graphs

1. Given an undirected graph G represented with adjacency lists, design and analyze an algorithm that determines if G is bipartite.

   Note: we say an undirected graph G is bipartite if the vertices can be divided into two disjoint sets U, V such that each edge connects a vertex in U to a vertex in V

Solution:

Start at an arbitrary node u, run BFS. Add all elements an even distance away fto U, and elements an odd distance to V. If there exists an edge between two odd nodes or edge between two even nodes, it is not bipartite. Otherwise, it must be bipartite.

2. Prove that an m-ary tree of height h has at most m^h leaves

   A rooted tree is m-ary if each non-leaf node has no more than m children. If m is 2, we call it a binary tree.

   Solution:

   Prove by (strong) induction.

   For our base case, let $h = 1$. An m-ary tree of height 1 has m at most leaves, so the statement holds for $h = 1$

   Inductive Hypothesis: Assume m-ary trees of height k, where $1 \leq k < h$, have at most m^k leaves.

   An m-ary tree of height h can be broken into m subtrees rooted at the m children of the root. These trees have at most height $h - 1$.

   Applying the IH, we have that each subtree has at most m^{h-1} leaves. Since there are m subtrees, combined they have at most $m \times m^{h-1} = m^h$ leaves, which are precisely the leaves of our original tree

3. Let G be an undirected graph with n vertices, where each vertex has a degree of at least $\frac{n}{2}$. Prove that G is connected.

   Solution:

   Suppose to the contrary that G is not connected. Then, there exists some component of G, call it G_1, that has at most $\frac{n}{2}$ vertices.

   Consider a vertex $v \in G_1$. By the information given, $\deg(v) \geq \frac{n}{2}$. But all the edges are in G_1, which is impossible since G_1 has at most $\frac{n}{2}$ vertices. This is a contradiction, thus G must be connected.