به نام خدا

دانشکده علوم ریاضی

آنالیز الگوریتم ها
نیم‌سال دوم ۰۱-۰۲

تمرین ۳- امیرحسین محمودی -۹۸۱۰۸۷۷۹

1

We must adjust the binary tree structure to accommodate three children instead of two. As a result, each internal node of the tree will have either two, or three children. The process for constructing a ternary Huffman tree is similar to that of the binary version. Initially, we need to determine the frequency of each symbol in the input data. After this, we create a leaf node for each symbol with its frequency serving as the weight. Then, we sort the leaf nodes in ascending order of frequency. Next, we repeat the following process until we are left with a single node: a) select the two nodes with the lowest frequency, b) combine them into a new internal node whose weight equals the sum of their weights, and c) designate this node as a child of the next node with the lowest frequency. Finally, the last internal node created in last step serves as the root node of the ternary tree. Time complexity:

$$T(priorityqueuecreation) + T(treeconstruction) = O(nlogn)$$

The analysis of optimality is almost identical to the binary case. We are placing the symbols of lowest frequency lower down in the final tree and so they will have longer codes than the more frequently occurring symbols. We show that this tree is makes minimum weighted path length among all ternary trees with the same weights. This can be proved using a greedy argument: at each step, we merge the tree smallest subtrees into a single node and update the weight of the parent node accordingly. This ensures that the weight of the root node (the sum of all symbol frequencies) is minimized.

suppose T is not optimal and T' is an optimal tree.

$$L(T') < L(T)$$

now we make tree T" by removing 3 minimum frequent nodes of T' and summing them and replacing a new node.

$$L(T') < L(T'')butL(T') = L(T'') + f_y > L(T') + f_y = L(T)$$

Which is contradiction.

2

odify Kruskal's algorithm.Sort the edges in decreasing order of weights and then start from an edge continue adding edges to the tree if they do not create a cycle. stop until got |V|-1 edges.

Correctness:

Let G = (V, E) be the input graph, and let T the maximum weight spanning tree. We prove it by contradiction. Suppose T is not a maximum weight spanning tree of G. There exists a tree T which :

$$weights(T) > weights(T)$$

T includes all vertices of G so it has an edge (u, v) which is not in T . Since the edges are sorted in descending order, weight of (u, v) is greater than or equal to the weight of any edge in T . so we can replace an edge from T with this edge and get a new Tree which has maximum weight of edges. so we can get a maximum weight spanning tree by repeat this, contradicting our assumption that T is not a maximum weight spanning tree. cut-property will justify this algorithm correctness.

3

To solve the puzzle with the fewest messages possible, follow these steps: first, assign a unique number to each person in the room, ranging from 1 to n. Now, the greedy algorithm is, people first tell their secret to only one person. For this, n-1 messages are needed. Then this person, who knows all the secrets, sends them to all the people, who in this case also understand all the secrets, and for this, n- 1 other messages are needed, which is equal to the minimum amount of sending messages. That is, it will be 2n-2. At least 2n− 2 messages are required. Consider the first person who succeeded in solving the puzzle. This person knows all the secrets, as a result, each secret has been shared at least once. Now, other people for whom the mystery has not been solved, at least do not know the secret. As a result, the secret must be told at least n− 1 times to know it. which in total 2n− 2 messages are exchanged.

4

4.1

We use cut property the heaviest edge as e=(u,v). We also consider the lightest edge in the same cycle e'=(q,w). Now we consider the cut S , V-S such a way that S includes the vertices of the path v to w and V-S includes the vertices of the path u to q. Based on cut−property, we can replace edge e with edge e' to have less weight.

4.2

According to part A, we know that the minimum tree does not contain the heaviest edge of any cycle. As a result, by removing these edges, there is no change in the MST, and no problem occurs. Also, due to the algorithm that removes every edge in the cycle, there will be no cycle in the graph when the algorithm is finished. Also, after removing cycle from graph, it is still remain connected. As a result, the graph that was initially connected will remain connected at the exit of the algorithm. As a result, at the end of the algorithm, the output tree contains all the edges that should exist in it because all the other edges have been removed, the tree is connected and has no cycle, and the algorithm works correctly.

4.3

Run the BFS algorithm from one vertex to another without considering the edge (u, v). if there is a path from one vertex to another without this edge, so this path along with the edge (u, v) is a cycle in the graph. Its time Complexity is equal to O(|E|+|V|).

4.4

The order of the algorithm is equal to the time of sorting the edges and also the detection that the edge is in a cycle, which is equal to:

$$O(ElogE) + O(E(E+V)) = O(E^2 + EV)$$

5

We present a greedy algorithm for this which guarantees that if the optimal solution is , the solution of this algorithm is less than 2 . The algorithm works in such a way that it selects the farthest point from the set every time. At first randomly select a center point. Then, if the vertices are $c_1, c_2, c...c_i$ we choose the next center in such a way that for each p calculate expression $Min(dist(p, c1_1).dist(p, c_2), ...dist(p, c_i))$ and get maximum of them. This algorithm guarantees that the answer is less than 2 . we prove it with the a contradiction. Suppose that this distance is greater than 2 . As a result, there is a point that is more than 2 away from all the centers, which, with the process of choosing the centers in the algorithm, results that all the centers are more than 2 away from each other. As a result, we have k+1 points whose distance are less than 2 from each other. According to the Pigeonhole Principle, two of these k+1 points fall in same center and since their distance is more than 2 , none of them can be the center. both These points are less than   away from the center, and as a result, the distance between them is less than 2 according to the triangle inequality, which is a contradiction. As a result, the answer of the greedy algorithm in the worst case is twice the optimal answer.