CS217 – Algorithm Design and Analysis

Homework 3

Not Strong Enough

March 22, 2020

1

Given an array A of n items (numbers), we can find the maximum with $n-1$ comparisons (this is simple). Show that this is optimal: that is, any algorithm that does $n-2$ or fewer comparisons will fail to find the maximum on some inputs.

Proof. Consider that the process of finding the maximum forms a "comparision tree". Now we explain how the "comparison tree" is formed:

1. At the beginning, each item of A is regarded as a node, and all nodes are separated. Each node has a value, from $A[0]$ to $A[n-1]$.
2. Each time we pick two nodes a and b such that both a and b have **no parent** in the tree.
3. Then we make a comparison between the values of a and b, and get the larger one.
4. We add a new node as the parent of both a and b. The value of the new node is set to the larger value of a and b.
5. Repeat step 2 to step 4. Stop when there is only one node which has no parent, i.e., all nodes constitute a tree. Finally the value of the root is the maximum, which is what we want.

From the process, we know that only the origin n nodes are leaves of the tree. And it is clear that every internal node, i.e., node that is not a leaf, has exactly two children, since the internal node is added after comparing its two children.

It is known that such kind of trees (whose internal nodes have exactly two children) have exactly $2n-1$ nodes: n leaves and $n-1$ internal nodes. So at least $n-1$ comparisons are needed. Otherwise we will get more than one tree in the end, and thus we can't determine the maximum. □