Homework 17

Brian Knotten, Brett Schreiber, Brian Falkenstein

October 4, 2018

24

Define S_1 to be the set of boxes on the left side of the scale, and S_2 to be the boxes on the right side of the scale. The algorithm begins by sorting the boxes in order from least to most heavy. This can be done in $nlogn$ time using any fast sorting method (Merge, quick, etc). The decision tree can be defined as:

- The root of the tree represent the scale with all n boxes in S_1
- For a node v at depth i, take $v's$ left child to be moving $B_i + 1$ to S_2, and $v's$ right child to leave $B_i + 1$ in S_1.

The size of this tree will be 2^n. The depth will be n, as each depth i represents deciding over box B_i, and each node has a branching factor of 2^n.
The tree can be pruned with the following rules:

1. If a node v has a configuration of S_1 and S_2 such that $|S_1| < |S_2|$ (that is, the scale is tipped to the right), prune v.
2. If 2 nodes u and v at the same depth have the properties of $|S_1|_u = |S_1|_v$ and $|S_2|_u = |S_2|_v$, prune v.

The first pruning rule works because if we are deciding over boxes from least to most heavy, and never considering a box twice, a scale tipped to the right can never be reconciled, as we will only have heavier boxes to decide over in the future, and adding any boxes to S_2 will only make that side heavier. Thus a scale tipped to the right will always stay tipped to the right. If at any point we reach a state of equilibrium, where $|S_1| = |S_2|$, the algorithm terminates and returns true.