10/10/08

6.856
Randomized Algorithms
2004/02/16

Dan Ports
drkp@mit.edu
Collaborators: {sarahl}

Problem Set 2

Problem 1:

a) Consider the graph that consists of a linear chain of $n-2$ vertices, and a source s and sink t, with edges between the nodes in the chain as well as from s and t to each of the vertices in the chain. The minimum cut has size $n-2$: it must cut either all of the edges leaving s or all of the edges entering t.

The graph consists of $n-2$ edges between s and the middle vertices, $n-2$ edges between t and the middle vertices, and $n-3$ edges in the chain. To find a minimum cut, the contraction algorithm must either never select any of the edges between s and the middle vertices, or any of the edges between t and the middle vertices. Considering one of these possibilities (the two are symmetric), there are $n-2$ edges that must survive. Since there are initially $3n-7$ edges and the number of edges only decreases, the probability of selecting such an edge is at least $1/3$. Thus, since the algorithm runs until at least $n-2$ edges have been contracted, the probability that no such edge is ever cut is at most $\left(\frac{2}{3}\right)^{n-2}$. The probability that the algorithm returns a minimum cut is twice this, since there is one minimum cut on each side. Thus, the probability of success is exponentially small. ✓

PICTURES

b) Now consider the graph that consists of s, t, $n-2$ middle vertices, and edges from s and t to each of the middle vertices. (This is the same as the previous graph, except without the chain of edges in the middle.) For any vertex x ($x \neq s, t$), either the edge between s and x or the edge between x and t must be in the minimum cut. There are two choices for each middle vertex, so this graph has 2^{n-2} minimum cuts.

This is the maximum number of cuts possible in a graph, and so the maximum number of minimum cuts. In any graph, each of the $n-2$ vertices other than s and t must be assigned to either the s side of the cut or the t side of the cut. There are 2^{n-2} possible assignments. ✓

Problem 2:

a) As before, we assume the adversarial model where an adversary chooses the value of each leaf as it is queried in response to the previous queries. The adversary chooses the value of each leaf such that no node's value can be determined without querying the value of every leaf in its subtree. We prove this claim by induction on the subtree height h. For $h = 1$, the subtree is simply the leaf, so the claim is clearly true. Now assume that the claim is true for some h. Then we wish to construct a subtree of height $h+1$ whose value cannot be determined without querying all of its leaves. We accomplish this using three subtrees of height h. By the induction hypothesis, we can construct these trees such that none of the $h+1$-height tree's children's values are determined until their subtree's leaves are all queried. Assume without loss of generality that the first child to be determined has value 1. Then we choose the remaining leaves for the other subtrees such that the second child to be determined has value 0. This forces the algorithm to continue evaluating leaves in the third subtree in order to determine the value of the $h+1$-height node. By the induction hypothesis, this requires all 3^{h+1} nodes in the subtree to be evaluated. This proves the claim.

b) If the first two children of a node selected evaluate to the same value, then there is no need to evaluate the third child. A non-deterministic algorithm can guess which two of the three children

at each level will evaluate to the majority value, and query only these two. The cost is $2^h = n^{\log_3 2}$ leaves.

c) If all three children of a node have the same value, the randomized algorithm will always need to query only two of the children. If one differs from the other two, the algorithm will need to query the third if it selected two that differ. There are three possible pairs, of which two have differing values, so with probability $2/3$, the algorithm will need to query three children. So the expected number of children queried is at most $2\,(1/3) + 3\,(2/3) = 8/3$. Thus, the expected number of leaves queried is

$$(8/3)^h = n^{\log_3 \frac{8}{3}} < n^{0.9}$$

Problem 3:

Lemma 1. *On a uniform input distribution of the permutations of n elements, the average number of comparisons required for any deterministic comparison-based sorting algorithm is $\Omega(n \log n)$.*

Proof. Consider the decision tree representation of a sorting algorithm. Each leaf node corresponds to a permutation of the input, so there are $n!$ leaves. The number of comparisons required when sorting any input permutation is the length of the path from the root to the associated leaf. Thus, we must consider the average height of a leaf node in any binary tree with $n!$ elements. Recall (from the analysis of treaps) that the average leaf height of a binary search tree created by inserting N leaves in random order is $\Theta(\log N)$. This is the same situation, with $N = n!$. So the average leaf height is $\Theta(\log n!) = \Theta(n \log n)$. This proves the lemma. □

We use this lemma and Yao's minimax principle to claim that any comparison-based Las Vegas algorithm for sorting has expected runtime $\Omega(n \log n)$. Yao's minimax principle tells us that a lower bound on the average-case runtime over any input distribution (in this case, we choose permutations uniformly at random) of any deterministic algorithm is a lower bound on the expected runtime of a Las Vegas algorithm. From the lemma, we have a lower bound of $\Omega(n \log n)$.

Problem 4:

a) We upper bound the probability of having a contiguous sequence of length $c + \lg n$ heads by counting. There are $n - (c + \lg n) + 1$ choices for the starting location of the sequence, and $2^{n-(c+\lg n)}$ choices for the values of the other coin flips. There are 2^n total outcomes, so the probability is at most

$$\frac{(n - c - \lg n + 1)2^{n-c-\lg n}}{2^n} = \frac{n - c - \lg n + 1}{2^{c+\lg n}} = \frac{n - c - \lg n + 1}{n2^c} \leq (1/2)^c$$

b) Consider the probability of encountering a contiguous sequence of $\lg n - d \lg \lg n$ heads. We wish to show that for some d, this probability is $1 - \frac{1}{n^k}$. Using similar analysis from before, we can lower bound the probability of encountering a contiguous sequence of $c + \lg n$ heads as $\frac{1}{n2^c}$. Setting $c = -d \lg \lg n$, we find that the probability of encountering a contiguous sequence of $\lg n - d \lg \lg n$ heads is at least

$$\frac{1}{n2^{-d \lg \lg n}} = \frac{(\lg n)^d}{n}$$

Setting $d = \frac{\lg(n - n^{-2k+1})}{\lg \lg n}$, the probability becomes

$$\frac{1}{n2^{-\frac{\lg(n-n^{-2k+1})}{\lg \lg n} \lg \lg n}} = \frac{n - n^{-2k+1}}{n} = 1 - \left(\frac{1}{n}\right)^k$$

so a contiguous sequence of $\lg n - d \lg \lg n$ heads occurs with high probability.