P1:10
P2:9
P3:10
P4:10

6.856
Randomized Algorithms
2004/02/09

Dan Ports
drkp@mit.edu

Problem Set 1

Problem 1:

a) We return 1 with probability p_1 and 2 with probability $p_2 = 1 - p_1$ by generating successive random bits and comparing them to the binary representation of p_1. If the ith random bit differs from the ith bit of p_1, we stop and return 1 if the bit of p_1 is greater and 0 if the random bit is greater.

Because each random bit has probability $1/2$ of differing from the corresponding bit of p_1, in expectation two random bits are required.

b) To sample from n items, we generate a balanced binary tree and augment each node with the sum of the weights in its subtree. If the weight of the root is p_r and the weights of its children are p_a and p_b, we first use the algorithm above to return the root with probability $\frac{p_r}{p_r+p_a+p_b}$. Otherwise, we traverse one of the subtrees. With probability $\frac{p_a}{p_a+p_b}$, we choose subtree a, and b with probability $\frac{p_b}{p_a+p_b}$. We then recursively apply this procedure.

The expected runtime is $O(\log n)$ because each at each node we perform two calls to the algorithm from part a, which requires constant time, and the depth of the tree is $O(\log n)$.

c) Consider the problem of sampling uniformly from $\{1, 2, 3\}$. Each result occurs with probability $1/3$. In the worst case, it is impossible to generate a result with the appropriate distribution with only a finite number of random bits. Note that the binary representation of $1/3$ continues infinitely. But if a finite number of bytes is used, then the precision of output probability is limited: if n bits are used, then the probability of any result must be a multiple of $\frac{1}{2^n}$. This precludes the correct probability $1/3$.

Problem 2:

a) Let n be the size of S, and s be the number of elements smaller than the pivot x. The size of the subset in the recursive call is at most $\max\{s, n-s-1\}$. Since the pivot is selected uniformly at random, the expectation is

$$\leq \frac{1}{n}\sum_{i=0}^{n-1} \max\{s, n-s-1\} \leq \frac{2}{n}\sum_{i=0}^{\lfloor \frac{n-1}{2} \rfloor} s = \frac{1}{n}\left(\frac{(n-1)^2}{4} + \frac{n-1}{2}\right) = \frac{n}{4} + \frac{3}{4n} < bn$$

(-1) close

for, e.g. $b = 1/2$ and large n.

b) Assume for purposes of induction that the expected runtime $\mathrm{E}\left[T(\frac{n}{2})\right] = c\frac{n}{2}$. Then consider solving a problem of size $n+1$. This requires comparing the n other elements to the randomly-selected pivot, then applying the algorithm recursively to one of the two subsets (call it S_i):

$$\mathrm{E}[T(n+1)] = \mathrm{E}[n + T(|S_i|)] \tag{1}$$

$$= n + \mathrm{E}[T(|S_i|)] = n + T(\frac{n}{2}) = n + c\frac{n}{2} \leq cn \tag{2}$$

for $c = 2$.

The base case is trivial: for a set with small constant size, we can simply sort the set and pick the kth smallest element in constant time.

c) Equation 2 uses linearity of expectation for the fact that $\mathrm{E}[n + T(|S_i|)] = n + T(\frac{n}{2})$. If the recurrence is non-linear, this equality does not necessarily hold.

OK, but be more concrete

Problem 3:

The MST-based minimum-cut algorithm is precisely equivalent to the randomized contraction algorithm. Assume for simplicity that we are using Kruskal's MST algorithm. The algorithm repeatedly selects the remaining edge with smallest weight — since the weights are random, this selects a random remaining edge. If the edge connects two distinct trees in the forest, it adds the edge to the minimum spanning tree, or otherwise deletes the edge. Each tree in the forest is equivalent to a (meta)vertex in the contraction algorithm. The process of adding an edge between two vertices to the MST is equivalent to contracting the two vertices, since all other edges between the two trees (internal edges in the contracted metavertices) will be discarded and not considered later in the procedure.

The edge with highest weight in the minimum spanning tree is necessarily the one that was inserted last; it is therefore equivalent to the final remaining edge in the contraction procedure. So with probability $\Omega\left(\frac{1}{n^2}\right)$, the cut it produces will be the minimum cut.

Problem 4:

As in the analysis of the contraction algorithm, let k be the size of the minimum cut. Then every vertex must have degree at least k, so $m \geq \frac{kn}{2}$. Further let $\varkappa$ be the size of the second smallest cut. Every vertex must have degree at least $\varkappa$, except possibly for the one vertex with smallest degree. Thus, $m \geq \frac{\varkappa(n-1)}{2}$.

The remainder of the algorithm proceeds in largely the same way as in the minimum-cut analysis. However, we stop when *three* vertices remain. The probability of not randomly choosing an edge of the second-smallest cut on the ith step is at least

$$1 - \frac{\varkappa}{\frac{\varkappa((n-i+1)-1)}{2}} = 1 - \frac{2}{n-i} = \frac{n-i-2}{n-i}$$

and so the probability of never choosing an edge of the second-smallest cut is at least

$$\prod_{i=1}^{n-3} \frac{n-i-2}{n-i} = \prod_{j=3}^{n-1} \frac{j-2}{j} = \frac{1}{3}\frac{2}{4}\frac{3}{5}\cdots\frac{n-4}{n-2}\frac{n-3}{n-1} = \frac{2}{(n-2)(n-1)}$$

Once three vertices remain, we choose one of the edges between them at random and use it to define the cut. There are at most three such edges, so the probability of returning the minimum cut is $\frac{2}{3(n-2)(n-1)} = \Omega\left(\frac{1}{n^2}\right)$.