Albert-Ludwigs-Universität, Inst. für Informatik
Prof. Dr. Fabian Kuhn
M. Ahmadi, A. R. Molla, O. Saukh January 26, 2016

Algorithm Theory, Winter Term 2015/16
Problem Set 11 - Sample Solution

Exercise 1: (7 points)

Assume that we are given a rooted tree with n leaves. All leaves have the same distance ℓ from the root and all nodes except for the leaves have three children. A problem instance consists of such a tree and a boolean value for each leaf.
The value of each inner node is defined as the majority value of its children. Initially, the values of inner nodes are not given. The objective is to compute the value of the root.
The performance of an algorithm to solve this problem is measured by the number of leaves whose values are read by the algorithm.

(a) (2 points) Is there a deterministic algorithm that can solve the problem such that for every input, the algorithm does not need to read the values of all leaves?

(b) (1 points) Design a recursive randomized algorithm to determine the value of the root with certainty but without necessarily reading all the leaves values.

(c) (4 points) What is the expected number of leaves that your randomized algorithm needs to read? Give an upper bound for this expected number of leaves.

Solution

(a) Assume existence of such deterministic algorithm. Let us consider an input instance - a simple tree of height 1 that consists only of a root and three leaves. Our algorithm has to process at most *two* leaves and decide what value does the root get. W.l.o.g. assume that leaves are processed in the left-to-right order, and according to our assumptions, we can only take a look at only the first two leaves and can correctly compute the value for the root, only if the processed values are the same(in this case, the value of the third leave is not important). In case, if the values of the processed leaves vary, we can not compute the value of the root in a deterministic way. So, such algorithm can not exist.

(b) Let us first outline some crucial facts:

   (a) For any node, at least two out of its three children have the same value. (Dirichlet's box principle).

   (b) If we know the values of two arbitrary children of a node and these values are the same, the node gets this value assigned, no matter which value the third child has.

   We can use these facts to design an algorithm which is as follows. For any node N we randomly pick two of its children. We recursively compute the values of those children. If the values are equal, we return this value as the answer. If values are not equal, we recursively compute the value of the third child and return it as the answer. If node N is a leaf, we just return its value.

(c) Here, we would like to calculate the expected number of leaves visited by the algorithm while computing the root's value. Let us define a random variable X_h that represents the number of visited leaves, while computing the value for an arbitrary node N at height h. The height of a node is its distance to the leaves. Also consider an event S to denote that the two randomly picked children of N have different values. Without knowing the value assignment of N's children, we claim that $\Pr(S) \leq \frac{2}{3}$. The reason is that in any case at least two of its three children have the same value. Therefore the algorithm, for evaluating the value of the node N, with probability 1 visits two of N's children and with $\Pr(S)$ visits the third one. Hence,

$$\begin{aligned}\mathbb{E}[X_h] &= 2 \cdot \mathbb{E}[X_{h-1}] + \Pr(S) \cdot \mathbb{E}[X_{h-1}] \\ &\leq 2 \cdot \mathbb{E}[X_{h-1}] + \frac{2}{3} \cdot \mathbb{E}[X_{h-1}].\end{aligned}$$

This gives us the following recurrence relation.

$$\mathbb{E}[X_h] \leq \frac{8}{3} \cdot \mathbb{E}[X_{h-1}] \tag{1}$$

Let us consider X_1 as the base case for the above recurrence relation. X_1 represents the number of leaves that are visited by a node of height 1. In the same as before we can argue that each node of height 1 has to read two of its children values with probability 1 and to additionally read the third child's value with probability at most $\frac{2}{3}$. Therefore,

$$\mathbb{E}[X_1] = 2 + \Pr(\text{reading the third child's value}) \cdot 1 \leq 2 + \frac{2}{3} = \frac{8}{3}. \tag{2}$$

From (1) and (2), we can conclude that

$$\mathbb{E}[X_h] \leq \left(\frac{8}{3}\right)^h. \tag{3}$$

Considering the height of the root which equals $\log_3 n$, the expected number of leaves visited by the algorithm is as follows.

$$\mathbb{E}[X_\ell] \leq \left(\frac{8}{3}\right)^\ell = \left(\frac{8}{3}\right)^{\log_3 n} = n^{\log_3\left(\frac{8}{3}\right)} = n^{\log_3 8 - \log_3 3} \approx n^{0.89}$$

Therefore, $n^{0.89}$ is an upper bound on the expected number of leaves values the algorithm reads while computing the value of the root.

Exercise 2: Max-Cut (5 points)

Let $G = (V, E)$ be an undirected graph. Consider the following randomized algorithm: Every node $v \in V$ joins the set S with probability 1/2. The algorithm's output is the cut $(S, V \setminus S)$. You can assume that $(S, V \setminus S)$ actually is a cut, i.e., $\emptyset \neq S \neq V$.

(a) (3 points) Show that with probability at least 1/3 this algorithm outputs a cut which is a 4-approximation to a maximum cut (i.e., a cut of maximum possible size).

Remark: *For a non-negative random variable X, the Markov inequality states that for all $t > 0$ we have $Pr(X \geq t) \leq \frac{E[X]}{t}$.*

Hint: *Apply the Markov inequality to the number of edges* **not** *crossing the cut.*

(b) (2 points) How can you use the above algorithm to devise a 4-approximation of a maximum cut with probability at least $1 - \left(\frac{2}{3}\right)^k$ for $k \in \mathbb{N}$. What is the success probability of your idea.

Remark: *If you could not solve a), you can still use the result as a black box for solving b).*

Solution

(a) Let us assume that the graph has m edges. For each edge e we define a $0-1$ random variable X_e as follows:

$$X_e = \begin{cases} 1 & \text{if the end points of } e \text{ are in different partitions i.e., } e \text{ crosses the cut} \\ 0 & \text{otherwise} \end{cases}$$

In addition, we define X as the sum of the above random variables. Then the number of edges crossing the cut will be given by:

$$X = \sum_{e \in E} X_e$$

The expected number of edges crossing the cut is given by: (using linearity of expectation)

$$\mathbb{E}[X] = \mathbb{E}\Big[\sum_{e \in E} X_e\Big] = \sum_{e \in E} \mathbb{E}[X_e] = \sum_{e \in E} \Pr(X_e = 1)$$

Since each node joins the set S with probability $1/2$, the probability that the end nodes of e are in different partitions is $1/2$. Hence,

$$\Pr(X_e = 1) = \frac{1}{2}$$

Therefore,

$$\mathbb{E}[X] = \sum_{e \in E} \Pr(X_e = 1) = \sum_{e \in E} \frac{1}{2} = \frac{m}{2}$$

It is easy to see that an upper bound on the number of edges that cross the cut is m. Hence the size of the max-cut $\leq m$. Let $\mathcal{E}$ be the event that the algorithm produces a cut of size less than $m/4$. Then $\Pr(\mathcal{E}) = \Pr(X \leq m/4)$.

Let us define a random variable Y as $Y = m - X$. Then $\mathbb{E}[Y] = m - \mathbb{E}[X] = m - m/2 = m/2$. Then

$$\begin{aligned} \Pr(\mathcal{E}) &= \Pr(X \leq \frac{m}{4}) \\ &= \Pr(Y \geq \frac{3m}{4}) \\ &\leq \frac{\mathbb{E}[Y]}{(3m/4)} \quad \text{[from Markov inequality]} \\ &= \frac{m/2}{3m/4} = \frac{2}{3} \end{aligned}$$

This shows that with probability at most $\frac{2}{3}$ the algorithm produces a cut of size less than $m/4$. This means with probability at least $1 - \frac{2}{3} = \frac{1}{3}$ the algorithm produces a cut of size more than $m/4$. Hence, the given algorithm outputs 4-approximation of the maximum cut (since the size of the max-cut is $\leq m$).

Note that, alternatively we can say that the probability the algorithm *fails* to produce such a cut is at most $\frac{2}{3}$.

(b) In order to guarantee the construction of 4-approximation of the maximum cut with probability $1 - \left(\frac{2}{3}\right)^k$, we repeat the above construction of max-cut algorithm k times and take the largest cut we find. Then the probability that we don't get $m/4$ edges or more is at most $(2/3)^k$, since all the repetitions are independent and the probability of failure of each repetition is at most $2/3$. In other words, the probability that we get at least $m/4$ edges is *at least* $1 - \left(\frac{2}{3}\right)^k$. Therefore, we can succeed in constructing the 4-approximation with *at least* $1 - \left(\frac{2}{3}\right)^k$ probability.

Notice that if we choose $k = c \log_{3/2} n$, then the probability of success is at least $\left(1 - \frac{1}{n^c}\right)$.