University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn
P. Bamberger, P. Schneider

Algorithms Theory
Sample Solution Exercise Sheet 10

Due: Tuesday, 26th of January 2021, 4 pm

Exercise 1: Max Cut *(10 Points)*

Let $G = (V, E)$ with $n = |V|, m = |E|$ be an undirected, unweighted graph. Consider the following randomized algorithm: Every node $v \in V$ joins the set S with probability $\frac{1}{2}$. The output is $(S, V \setminus S)$.

(a) What is the probability to obtain a cut? *(1 Point)*

(b) For $e \in E$ let random variable $X_e = 1$ if e crosses the cut, and $X_e = 0$, else. Let $X = \sum_{e \in E} X_e$. Compute the expectation $\mathbb{E}[X]$ of X. *(2 Points)*

(c) Show that with probability at least 1/3 this algorithm outputs a cut of size at least $m/4$ (that is a cut of maximum possible size can be at most 4 times as large). *(4 Points)*

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

(d) Show how to use the above algorithm to obtain a cut with at least $m/4$ edges with probability at least $1 - \left(\frac{2}{3}\right)^k$ for $k \in \mathbb{N}$. *(3 Points)*

Remark: If you did not succeed in (c) you can use the result as a black box for (d).

Sample Solution

(a) We obtain no cut if no node joins S or if all nodes join S, because in either case one of the sets S or $V \setminus S$ is empty. The probability that one of these events happens is $2(\frac{1}{2})^n = (\frac{1}{2})^{n-1}$.

(b) Each node joins either side of the cut with equal probability. For an edge $e = \{u, v\} \in E$ the probability that its endpoints u, v join different sides is two out of four equally probable outcomes ($u \in S$ and $v \notin S$ or $u \notin S$ and $v \in S$ or $u, v \in S$ or $u, v \notin S$). Hence $\mathbb{P}(X_e = 1) = \frac{1}{2}$. We obtain

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

(c) Let $\mathcal{E}$ be the event that the algorithm produces a cut of size less than $\frac{m}{4}$. Then $\mathbb{P}(\mathcal{E}) = \mathbb{P}(X < \frac{m}{4})$. Define random variable Y as $Y = m - X$. Then $\mathbb{E}[Y] = m - \mathbb{E}[X] = m - \frac{m}{2} = \frac{m}{2}$. We get

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

Hence the probability that the algorithm produces a cut of size less or equal $\frac{m}{4}$ is at most $\frac{2}{3}$, which means with probability at least $1-\frac{2}{3}=\frac{1}{3}$ we get a cut of size more than $\frac{m}{4}$. Obviously the number of edges that cross any cut is at most m. Therefore our algorithm outputs a $\frac{1}{4}$-approximation with probability $\frac{1}{3}$.

(d) In order to guarantee a $\frac{1}{4}$-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 do not get $\frac{m}{4}$ 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 $\frac{m}{4}$ edges is *at least* $1-\left(\frac{2}{3}\right)^k$.

Exercise 2: Ternary Tree *(10 Points)*

Consider a full, complete ternary tree where each *inner* node has exactly three child nodes. Note that since the tree is full and complete, all leaves have the same distance (= height) ℓ from the root. Let n be the number of leaves of the tree.
Each leave is given a boolean value. The value of an inner node is defined recursively as the majority value of its three direct children. 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 values of leaves it reads.

(a) Is there a *deterministic* algorithm to determine the value of the root, such that for any given input, the algorithm does not need to read the values of *all* leaves? Explain your answer carefully. *(3 Points)*

(b) Give a *recursive, randomized* algorithm (analysis in part (c)) that *always* determines the value of the root but reads at most a^ℓ leaves *in expectation* for $a < 3$. *(4 Points)*

Remark: You can also show that only a q^ℓ-fraction of all leaves is read in expectation for $q < 1$.

(c) Based on the algorithm of (b) give a *tight* upper bound (as a function of the number of leaves n) for the *expected* number of leaves that are read by the algorithm. *(3 Points)*

Sample Solution

(a) To disprove the existence let us consider an arbitrary deterministic algorithm. We construct an input instance where it fails: a simple tree of height 1 that consists only of a root and three leaves. Our algorithm has to process at least *two* leaves to decide what value the root gets.

Consider the two leaves that the algorithm reads first. These are *fixed* since the algorithm is deterministic. We construct our input tree such that the values of these first two leaves *varies.* Therefore the algorithm has to read the third leaf to determine the value of the root. Consequently, it has to read *all* the leaves of this input and the answer is no.

(b) Let us first outline some crucial facts:

(i) For any node, at least two out of its three children have the same value, since we have three nodes and two values (pigeonhole principle).

(ii) If we know that two children have equal value we know the value of the node and we do not have to evaluate the third child.

We can use these facts to design an algorithm that returns the value of a given node N as follows. If node N is a leaf, we just return its value. Otherwise we randomly pick two children. We recursively compute the values of those children. If the values are equal, we return this value (ii). Otherwise we recursively compute the value of the third child and return the definite value.

Remark: We expected only an algorithm with said property, and not an actual computation of a.

(c) Let random variable X_h represent the number of visited leaves, while computing the value of a node N at height h. The height h of a node is its distance to the leaves. Let $\mathcal{E}$ denote the event that the two randomly picked children of N have different values.

Due to (i) at least two of three children of N have the same value. If we fix two children of same value, then the probability to randomly pick one of those two out of a total of three is $\frac{2}{3}$. The probability to pick another of the same value from the two remaining nodes is at least $\frac{1}{2}$ (if the third child also has the same value the probability is 1). Hence $\mathbb{P}(\mathcal{E}) \leq 1 - \frac{2}{3} \cdot \frac{1}{2} = \frac{2}{3}$. Note that the inequality becomes an equality if *exactly* two out of three children of N have the same value.

Our algorithm visits two children of N and with $\mathbb{P}(\mathcal{E})$ visits the third one. For $h \geq 2$ we obtain the following recursion for the expected visited number of leaves

$$\begin{aligned}\mathbb{E}[X_h] &= 2 \cdot \mathbb{E}[X_{h-1}] + \mathbb{P}(\mathcal{E}) \cdot \mathbb{E}[X_{h-1}] \\ &\leq 2 \cdot \mathbb{E}[X_{h-1}] + \frac{2}{3} \cdot \mathbb{E}[X_{h-1}] = \frac{8}{3} \cdot \mathbb{E}[X_{h-1}].\end{aligned}$$

Again we have equality if we have *exactly* two children with the same value. In the base case we have $\mathbb{E}[X_1] \leq 2 + \frac{2}{3} = \frac{8}{3}$ and we conclude that $\mathbb{E}[X_h] \leq \left(\frac{8}{3}\right)^h$. Considering the height $\log_3 n$ of the root, the expected number of leaves visited by the algorithm is

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

The upper bound is tight since we have equality for the above expectation when the algorithm is given a tree where each node that is not a leaf has exactly two children with the same boolean value while the third has the other value.

Remark: The expected proportion of visited leaves is at most $\mathbb{E}[X_\ell]/3^\ell = \left(\frac{8}{9}\right)^\ell$ *(i.e.* $q = \frac{8}{9}$*).*