Albert-Ludwigs-Universität, Institut für Informatik
Prof. Dr. Fabian Kuhn
M. Ahmadi, P. Schneider, J. Uitto

February 6, 2018

# Algorithm Theory - Winter Term 2017/2018
# Exercise Sheet 6

**Hand in by Thursday 10:15, January 25, 2017**

## Exercise 1: Trinary Tree (2+5+3 Points)

Consider a complete, full trinary tree with $n$ leaves, i.e. all leaves have the same distance (= height) $\ell$ from the root and all nodes except for the leaves have three children. Each leave is given a boolean value. According to the values of the leaves we can determine values of the inner nodes. The value of an inner node is defined 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 leaves that need to be read by the algorithm.

(a) Is there a deterministic algorithm that can 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.

(b) Design a recursive, randomized algorithm to determine the value of the root with certainty but reading only a proportion of $q^\ell$ leaves in expectation (for $q < 1$).

(c) Based on the result of (b) give a tight upper bound (in the number of leaves $n$) for the expected number of leaves that are read by the algorithm.

## Sample Solution

(a) To disprove the existence of such an algorithm let us consider 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. W.l.o.g. we first take a look at the first two leaves. However, if the values of the first two leaves vary, we have to read the third leaf to determine the value of the root. Therefore such an algorithm can not exist.

(b) Let us first outline some crucial facts:

(i) For any node, at least two out of its three children have the same value (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 $q$.*

(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

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

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.* $q = \frac{8}{9}$*, however we did not expect you to actually compute* $q$*, though).*

## Exercise 2: Compare Polynomials (2+3+5 Points)

You are given two polynomials $p$ and $q$ of degree $n$. However, you are not given the polynomials in an explicit form. You can only evaluate them at some value $x \in \{1, \ldots, 2n\}$ (i.e., you can compute $p(x)$ and $q(x)$ for values $x \in \{1, \ldots, 2n\}$). You want to find out whether the two polynomials are identical.

(a) Pick $x \in \{1, \ldots, 2n\}$ uniformly at random. What is the minimum probability that $p(x) \neq q(x)$ in case $p \neq q$?

(b) Give an efficient[1] randomized *Monte Carlo* algorithm that tests whether the two polynomials are identical in $\mathcal{O}(\log \frac{1}{\varepsilon})$. In specific, if $p = q$, your algorithm should always return "yes", if $p \neq q$, your algorithm is allowed to err with probability at most $\varepsilon \in (0, 1)$.

(c) Prove the properties of your algorithm.

## Sample Solution

(a) If $p(x) = q(x)$ for *more than* $n$ values $x \in \mathbb{R}$ then this implies $p = q$. The contraposition is that $p \neq q$ implies $p(x) = q(x)$ for *at most* $n$ values $x \in \mathbb{R}$. Assuming that $p \neq q$ is the case then $p(x) = q(x)$ for at most half the possible values $x \in \{1, \ldots, 2n\}$. Hence, if we pick $x \in \{1, \ldots, 2n\}$ uniformly at random we have $\mathbb{P}\big(p(x) \neq q(x) \mid p \neq q\big) \geq \frac{1}{2}$.

We have equality $\mathbb{P}\big(p(x) \neq q(x) \mid p \neq q\big) = \frac{1}{2}$ for two polynomials $p \neq q$ with $p(x) = q(x)$ for exactly $n$ distinct choices of $x \in \{1, \ldots, 2n\}$ (such two polynomials $p, q$ exist). Therefore $\frac{1}{2}$ is in fact the minimum of $\mathbb{P}\big(p(x) \neq q(x) \mid p \neq q\big)$.

[1] Measured by the number of polynomial evaluations it needs to perform.

(b)

**Algorithm 1:** `TestEquality`$(p, q)$

```
for ⌈log_2 1/ε⌉ iterations do
    Pick x ∈ {1, ..., 2n} uniformly at random
    if p(x) ≠ q(x) then
        return "No"
return "Yes"
```

(c) It's obvious that our algorithm performs in $\mathcal{O}(\log \frac{1}{\varepsilon})$[1]. For the correctness first assume that the input are two equal polynomials $p = q$. Then it doesn't matter which $x$ is picked we always have $p(x) = q(x)$ and after the loop we definitely return "yes".

On the other hand, if $p \neq q$ we must find a witness value $x$ with $p(x) \neq q(x)$ for the output "No". The probability *not* to pick a witness, i.e. $p(x) = q(x)$ during *every* iteration is at most

$$\Big(1 - \mathbb{P}\big(p(x) \neq q(x) \mid p \neq q\big)\Big)^{\lceil \log_2 \frac{1}{\varepsilon} \rceil} \leq \left(\frac{1}{2}\right)^{\log_2 \frac{1}{\varepsilon}} = \frac{1}{1/\varepsilon} = \varepsilon.$$

## Exercise 3: Max Cut (1+2+4+3 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 actually obtain a cut?

(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 $X$.

(c) Show that with probability at least $1/3$ this algorithm outputs a cut which is a $\frac{1}{4}$-approximation to a maximum cut (i.e. a cut of maximum possible size is at most 4 times as large).

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

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

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

*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 $\mathbb{P}(\mathcal{E}) = \mathbb{P}(X \leq \frac{m}{4})$.

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

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

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 the construction of $\frac{1}{4}$-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 don't 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 4: Graph Connectivity (2+4+4 Points)

Let $G=(V,E)$ be a graph with $n$ nodes and edge connectivity[2] $\lambda \geq \frac{16 \ln n}{\varepsilon^2}$ (where $0<\varepsilon<1$). Now every edge of $G$ is removed with probability $\frac{1}{2}$. We want to show that the resulting graph $G'=(V,E')$ has connectivity $\lambda' \geq \frac{\lambda}{2}(1-\varepsilon)$ with probability at least $1-\frac{1}{n}$. This exercise will guide you to this result.
*Remark: If you don't succeed in a step you can use the result as a black box for the next step.*

(a) Assume you have a cut of $G$ with size $k \geq \lambda$. Show that the probability that the same cut in $G'$ has size *strictly smaller* than $\frac{k}{2}(1-\varepsilon)$ is at most $e^{-\frac{\varepsilon^2 k}{4}}$.

(b) Let $k \geq \lambda$ be fixed. Show that the probability that at least one cut of $G$ with size $k$ becomes a cut of size *strictly smaller* than $\frac{k}{2}(1-\varepsilon)$ in $G'$ is at most $e^{-\frac{\varepsilon^2 k}{8}}$.
*Hint: You can use that for every $\alpha \geq 1$, the number of cuts of size at most $\alpha\lambda$ is at most $n^{2\alpha}$.*

(c) Show that for large $n$ the probability that at least one cut of $G$ with *any* size $k \geq \lambda$ becomes a cut of size *strictly smaller* than $\frac{k}{2}(1-\varepsilon)$ in $G'$, is at most $\frac{1}{n}$.
*Hint: Use another union bound.*

## Sample Solution

(a) Let $C$ be the edges of a cut of size $k$. For $e \in C$ let $X_e = 1$ if $e \in G'$ and else $X_e = 0$. Let $X = \sum_{e \in C} X_e$. The expectation is $\mathbb{E}[X] = \sum_{e \in C} \mathbb{E}[X_e] = \frac{k}{2}$. We use a Chernoff bound

$$\mathbb{P}\big(X < \mathbb{E}[X](1-\varepsilon)\big) \leq \exp\left(-\frac{\varepsilon^2 \mathbb{E}[X]}{2}\right) = \exp\left(-\frac{\varepsilon^2 k}{4}\right).$$

[2] The connectivity of a graph is the size of the smallest cut $(S, V \setminus S)$ in $G$.

(b) According to the hint we have at most $n^{\frac{2k}{\lambda}}$ many cuts of size $k$. For an arbitrary cut of $C$ of size $k$ let $C' := C \cap E'$. With a union bound we obtain

$$\begin{aligned}
\mathbb{P}\Big(\bigcup_{\substack{C \text{ cut}\\|C|=k}} \big(|C'| < \frac{k}{2}(1-\varepsilon)\big)\Big) &\le \sum_{\substack{C \text{ cut}\\|C|=k}} \mathbb{P}\Big(|C'| < \frac{k}{2}(1-\varepsilon)\Big) && \text{(union bound)}\\
&\le \sum_{\substack{C \text{ cut}\\|C|=k}} \exp\Big(-\frac{\varepsilon^2 k}{4}\Big) && \text{(a)}\\
&\le n^{\frac{2k}{\lambda}} \exp\Big(-\frac{\varepsilon^2 k}{4}\Big) && \text{(Hint)}\\
&= \exp\Big(\frac{2k\ln n}{\lambda}\Big)\exp\Big(-\frac{\varepsilon^2 k}{4}\Big) && (\lambda \ge \tfrac{16\ln n}{\varepsilon^2})\\
&\le \exp\Big(\frac{\varepsilon^2 k}{8} - \frac{\varepsilon^2 k}{4}\Big) = \exp\Big(-\frac{\varepsilon^2 k}{8}\Big).
\end{aligned}$$

(c) For brevity let $\mathcal{E}(k)$ (for $k \ge \lambda$) be the event $\bigcup_{\substack{C \text{ cut}\\|C|=k}} \big(|C'| < \frac{k}{2}(1-\varepsilon)\big)$ (recall $C' := C \cap G'$). Then

$$\begin{aligned}
\mathbb{P}\Big(\bigcup_{k=\lambda}^{n-1} \mathcal{E}(k)\Big) &\le \sum_{k=\lambda}^{n-1} \mathbb{P}\big(\mathcal{E}(k)\big) \overset{(b)}{\le} \sum_{k=\lambda}^{n-1} \exp\Big(-\frac{\varepsilon^2 k}{8}\Big) \le \sum_{k=\lambda}^{n-1} \exp\Big(-\frac{\varepsilon^2 \lambda}{8}\Big)\\
&\le \sum_{k=\lambda}^{n-1} \exp\Big(-2\ln n\Big) = \sum_{k=\lambda}^{n-1} n^{-2} \le \frac{1}{n}. && (\lambda \ge \tfrac{16\ln n}{\varepsilon^2})
\end{aligned}$$

Or another solution with a geometric series which holds for large $n$

$$\begin{aligned}
\mathbb{P}\Big(\bigcup_{k=\lambda}^{n-1} \mathcal{E}(k)\Big) \le \mathbb{P}\Big(\bigcup_{k=\lambda}^{\infty} \mathcal{E}(k)\Big) &\le \sum_{k=\lambda}^{\infty} \mathbb{P}\big(\mathcal{E}(k)\big) \overset{(b)}{\le} \sum_{k=\lambda}^{\infty} \exp\Big(-\frac{\varepsilon^2 k}{8}\Big) && \text{(geometric series)}\\
&= \frac{e^{-\frac{\varepsilon^2\lambda}{8}}}{1-e^{-\frac{\varepsilon^2}{8}}} = \frac{n^{-2}}{1-e^{-\frac{\varepsilon^2}{8}}} = \frac{1}{n}\cdot\underbrace{\frac{1}{n\cdot(1-e^{-\frac{\varepsilon^2}{8}})}}_{\text{fraction } \le 1 \text{ for large } n} \le \frac{1}{n}.
\end{aligned}$$