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

Algorithms Theory
Sample Solution Exercise Sheet 6

Due: Monday, 21st of January, 2019, 14:15 pm

Exercise 1: Contention Resolution *(2+2+4 Points)*

Consider the contention resolution problem explained in the lecture with n processes and a single shared resource. We would like to calculate the expected number of time slots until every process has been successful at least once. For all integers $i \leq n$, let random variable T_i denote the smallest integer such that exactly i different processes are successful to access the resource in the first T_i time slots.

(a) Let t be an arbitrary time slot in $[T_j + 1, T_{j+1}]$ for an arbitrary integer $j < n$. What is the probability that some process becomes successful for the first time in time slot t?

(b) For all $i \leq n$, let random variable X_i be the number of rounds needed for the i^{th} process to succeed after exactly $i-1$ distinct processes have succeeded, i.e., $X_i := T_i - T_{i-1}$. Then, for an arbitrary integer $j \leq n$, what is the expected value of X_j?

(c) What is the expected value of T_n, the time for all processes to succeed at least once?

Hint: The probability that some process is successful in a given time slot is $(1-1/n)^{n-1}$. *We have seen that this probability is approximately* $1/e$. *For simplicity, you can assume that this probability is exactly* $1/e$.

Sample Solution

(a) In every time slot in $[T_j + 1, T_{j+1}]$, there are $n - j$ processes that have not yet been successful. Therefore, considering the given hint, the probability that one of these nodes becomes successful in any such round is $\frac{1}{e} \cdot \frac{n-j}{n}$.

(b) Considering part (a), X_j has geometric distribution with parameter $\frac{n-j+1}{en}$. Therefore, the expected value of X_j is $\frac{en}{n-j+1}$.

(c) Due to the definition of T_i and X_i, considering $T_0 = X_0 = 0$, it holds that $T_n = \sum_{i=1}^{n} X_i$. Therefore, considering the linearity of expectation, it holds that

$$\begin{aligned}
\mathbb{E}[T_n] &= \mathbb{E}[X_1 + X_2 + \cdots + X_n] \\
&= \mathbb{E}[X_1] + \mathbb{E}[X_2] = \ldots \mathbb{E}[X_n] \\
&= \sum_{i=1}^{n} \frac{en}{n-i+1} \\
&= \sum_{j=1}^{n} \frac{en}{j} \\
&= en \cdot H(n) \\
&\approx en \cdot \ln n
\end{aligned}$$

Exercise 2: Randomized Independent Set Algorithm *(6+7 Points)*

Let $G = (V, E)$ be a graph with n vertices and m edges. An independent set in a graph G is a subset $S \subseteq V$ of the nodes such that no two nodes in S are connected by an edge. Let $d := \frac{1}{n}\sum_{v \in V} \deg(v) = \frac{2m}{n}$ be the average node degree and consider the following randomized algorithm to compute an independent set S.

(I) Start with an empty set S. Then independently add each node of V with probability $1/d$ to S (you can assume that $d \geq 1$).

(II) The subgraph induced by S might still contain some edges and we therefore need to remove at least one of the nodes of each of the remaining edges. For this, we use the following deterministic strategy: As long as S is not an independent set, pick an arbitrary node $u \in S$ which has a neighbor in S and remove u from S.

It is clear that the above algorithm computes an independent set S of G.

(a) Find a (best possible) lower bound on the expected size of S at the end of the algorithm. Your lower bound should be expressed as a function of n and d.

Hint: First compute the expected numbers of nodes in S and edges in $G[S]$ after Step (I) of the algorithm.

(b) Assume that the above algorithm has running time $T(n)$ and that it computes an independent set of size at least $\frac{n}{5d}$ with probability at least $\frac{1}{2}$.

Show how to compute an independent set of size at least $\frac{n}{5d}$ with probability $1 - \frac{1}{n}$. What is the running time of your algorithm?

Sample Solution

Let $G = (V, E)$ be the given graph and let $n = |V|$ denote the number of nodes, $m = |E|$ the number of edges and $d = \frac{2m}{n}$ the average degree.

(a) We first compute the expected number of nodes in S after the first step. Let X be the random variable which indicates the size of S. By linearity of expectation we obtain

$$E[X] = \sum_{v \in V} \frac{1}{d} = \frac{n}{d}.$$

Let Y be the random variable which denotes the number of edges in $G[S]$ after the first step. Each edge $e \in E$ exists in $G[S]$ if and only if both of its adjacent nodes joined S in the first step, which happens with probability $1/d^2$. Thus we obtain

$$E[Y] = \sum_{e \in E} \frac{1}{d^2} = \frac{m}{d^2}.$$

Now, we use that $m = dn/2$ and obtain $E[Y] = dn/2d^2 = n/2d$.

In step (II) all edges are removed. Therefore, the size of the independent set after step (II) is at least $X - Y$ because we remove at most one node for each edge in $G[S]$. Combining both results and using linearity of expectation we obtain that the expected number of nodes after step (II) is at least

$$E[X - Y] = E[X] - E[Y] = \frac{n}{2d}.$$

(b) Let $\mathcal{A}$ denote the above algorithm which finds an independent set of size at least $n/5d$ with probability $1/2$. We amplify the probability by executing algorithm $\mathcal{A}$, k times (with independent

probabilities), where we determine the proper value of k later. We return the largest independent set of all k invocations of the algorithm.

If we have k invocations, the probability that we do not return an independent set of size at least $n/5d$ after the k invocations is the same as the probability that none of the independent invocations returns an independent set of size at least $n/5d$. This probability can be bounded by

$$(1 - \frac{1}{2})^k = \frac{1}{2^k}.$$

To solve the question we need that this probability is at most $1/n$. Thus, setting $k = \lceil \log_2(n) \rceil$ is sufficient. Then the runtime of the algorithm will be $k \cdot T(n) = \lceil \log_2(n) \rceil \cdot T(n)$.

Exercise 3: Randomized Partial 3-Coloring *(7 Points)*

The maximum 3-coloring problem asks for assigning one of the colors $\{1, 2, 3\}$ to each node $v \in V$ of a graph $G = (V, E)$ such that the number of edges $\{u, v\} \in E$ for which u and v get different colors is maximized. A simple randomized algorithm for the problem would be to (independently) assign a uniform random color to each node.
What is the expected approximation ratio of this algorithm?

Hint: Consider the approximation ratio to be the minimum ratio of the algorithm solution to the optimal solution over all input instances.

Sample Solution

Let $G = (V, E)$ be the given graph with $n = |V|$ and $m = |E|$. Let X denote the random variable which indicates the number of edges where both endpoints have different colors. For a single edge $e \in E$ the probability that both endpoints have different colors is $2/3$. Thus we obtain

$$E[X] = \sum_{e \in E} \frac{2}{3} = \frac{2}{3} \cdot m.$$

Because any optimal algorithm can at most color the endpoints of all edges ($= m$) differently, we can ensure an expected approximation ratio of $2/3$.
Note: In the literature, the approximation ratio is often defined via $\frac{\text{Opt. Solution}}{\text{Alg. Solution}}$ instead of $\frac{\text{Alg. Solution}}{\text{Opt. Solution}}$. With this definition we could ensure an approximation ratio of $3/2$.
Remark: One can also show that this approximation factor is tight: In a ring an optimal 3-coloring really colors the endpoints of all m edges differently and our algorithm (in expectation) only colors the endpoints of $2/3$ of the edges differently.

Exercise 4: Max Cut *(1+3+5+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 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 $\Pr(X_e = 1) = \frac{1}{2}$. We obtain

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

(c) Let $\mathcal{E}$ be the event that the algorithm produces a cut of size less than $\frac{m}{4}$. Then $\Pr(\mathcal{E}) = \Pr(X \leq \frac{m}{4})$. 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

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

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* $1-\left(\frac{2}{3}\right)^k$.