Albert-Ludwigs-Universität, Inst. für Informatik
Prof. Dr. Fabian Kuhn
H. Ghodselahi, Y. Maus

February 2, 2017

# Algorithm Theory, Winter Term 2016/17
# Problem Set 7

**Sample Solution**

## Exercise 1: Randomized Independent Set Algorithm (10+10 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 $d \geqslant 1$).

(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) (10 points) 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) (10 points) Assume that the above algorithm has running time $T(n)$ and that it computes an independent set of size $\frac{n}{5d}$ 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?

## 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 2: Randomized partial 3-coloring (10 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?

## 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 3: Random Max Cut Computation (10 points)

In the lecture, we discussed the random contraction algorithm to obtain a minimum edge cut. One could try to use the same algorithm to also find a maximum edge cut (partition $A \subset V, B = V \backslash A$ of the nodes so that the number of edges connecting nodes in $A$ and $B$ is maximized).

Show that for some graphs, the probability that the contraction algorithm returns a maximum cut is 0.

## Solution:

Let $G = (L \cup R, E)$ be the complete bipartite graph with $n/2$ nodes on either side.

The maximum cut of this graph is $S = (L, R)$ and has size $(n/2)^2$.

But as soon as the contraction algorithm contracts a single edge in this graph, it destroys at least one edge. Hence, it will always return a cut which is smaller than $S$, thus the probability that it returns the maximum cut in this graph is zero.

*Remark: It is not crucial that we have the complete bipartite graph. Actually most bipartite graphs work as a counter example.*