Exam: Randomized Algorithms and Probabilistic Methods

Exercise 1
(6 points)
We generate a random graph on \( n \) vertices \( V = \{v_1, \ldots, v_n\} \) as follows. Every vertex \( v_i \in V \) chooses a color \( c(v_i) \in \{1, 2, 3\} \) uniformly at random and independently of all other vertices. We then add an edge between two vertices if and only if the sum of their colors is 3. Let \( X \) denote the number of edges in this graph.

Calculate \( \mathbb{E}[X] \) and \( \text{Var}[X] \) and show that for every constant \( \varepsilon > 0 \) we have \( \Pr[(1 - \varepsilon) \mathbb{E}[X] \leq X \leq (1 + \varepsilon) \mathbb{E}[X]] = 1 - o(1) \).

Remark. You only need to calculate expectation and variance up to when it can easily be evaluated by a computer for a given value of \( n \). That is, your final form may contain binomial coefficients, fractions, products, sums, but no random variables or terms of the form \( \Pr[\ldots] \).

Exercise 2
(6 points)
We roll a six-sided dice \( n \) times. We do not start with a regular dice though, but with one that has 1 dot on one face and 0 dots on all other faces. Every time we roll the dice we first count the number of dots on the side that faces up and then paint an additional dot on this side. Let \( X_i \) denote the number of dots we count in the \( i \)-th dice roll (again: before painting an additional dot). Moreover, let \( X = \sum_{i=1}^{n} X_i \).

Calculate \( \mathbb{E}[X] \) and prove that for every constant \( \varepsilon > 0 \) we have \( \Pr[|X - \mathbb{E}[X]| \geq \varepsilon \mathbb{E}[X]] \leq e^{-\Omega(n)} \).

Exercise 3
(6 points)
Let \([n] = \{1, 2, \ldots, n\}\) for every \( n \in \mathbb{N} \). By \([n]_{\frac{1}{2}}\) we denote a random subset of \([n]\) in which every element is contained with probability \( \frac{1}{2} \) independently of all other elements. Let \( \varepsilon > 0 \) be a constant. Prove that \([n]_{\frac{1}{2}}\) contains at least one interval of length \((1 - \varepsilon) \log_{2} n\) with probability \( 1 - e^{-\Omega(n)} \).

Exercise 4
(6 points)
In this exercise we want to calculate the chances of \( n \) drunk men and women to successfully cross a narrow bridge.

Let \( \varepsilon > 0 \) be a fixed constant. Consider \( n \) persons and a bridge of length \( n \) meters and width \( w = w(n) \) meters. Each of the \( n \) drunk persons starts at one end of the bridge, centered between the left and right edge. In each step he or she moves one meter forward and, with probability \( \frac{1}{2} \) each and independently of previous steps, one meter to the left or one meter to the right. A person successfully crosses the bridge if he or she performs \( n \) steps without ever drifting off by more than \( w(n)/2 \) meters to the left or right.

Prove that there exists a constant \( C > 0 \) such that for \( w = C \sqrt{n \ln n} \) we have with probability \( 1 - o(1) \) that all persons will successfully cross the bridge.

(please turn over)
**Exercise 5**  
(4+4=8 points)

Given a simple graph $G = (V,E)$ we consider the Markov-chain that describes a fast random walk of a particle on the the graph $G$. Our state space is $S = V$. The particle starts in an arbitrary vertex of the graph. If the location of the particle after $t-1$ steps is $v \in V$, then in step $t$ we choose a neighbor $u \in N(v)$ uniformly at random, and a neighbor $w \in N(u)$ uniformly at random and change the particles location from $v$ to $w$ in the $t$-th step. If $v$ has no neighbors, then we simply stay at $v$.

(a) Prove a simple characterization of the graphs $G$ for which this Markov-chain is ergodic! That is, use at most 4 words to complete the following sentence and prove what your sentence claims: The Markov-chain is ergodic if and only if $G$ is . . .

(b) Consider the Markov-chain when $G$ consists of two vertex-disjoint copies of $K_n$ that are joined by a single edge. Show that in this case the Markov-chain is rapidly-mixing.

**Exercise 6**  
(4+8=12 points)

With their seminal paper from 1960 Erdős and Rényi initiated the study of random graphs. They studied how the component structure of a random graph $G_{n,p}$ looks like depending on the edge probability $p$. In particular they were interested in the size of the largest component of such a graph and showed that it undergoes a phase transition at $p = \frac{1}{\ln n}$. For every graph $G$ let $L(G)$ denote the size of the largest component in $G$. Then Erdős and Rényi showed that for every constant $c \geq 0$ we have that

(i) if $c < 1$ then with high probability $L(G_{n,c/n}) = O(\log n)$, and

(ii) if $c > 1$ then with high probability $L(G_{n,c/n}) = \Omega(n)$.

In this exercise we discuss a simple proof for (i) which was recently discovered by Krivelevich and Sudakov. The proof is based on DFS (Depth First Search). Recall that the DFS is a graph search algorithm that visits all vertices of a (directed or undirected) graph $G = (V,E)$ as follows. (If you prefer, there is a pseudocode version of the algorithm on the next page, which is equivalent to the following textual description.) It maintains three sets of vertices, letting $S$ be the set of vertices whose exploration is complete, $T$ be the set of unvisited vertices, and $U = V \setminus (S \cup T)$, where the vertices of $U$ are kept in a stack (the last in, first out data structure). It is also assumed that some order $\sigma$ on the vertices of $G$ is fixed, and the algorithm prioritizes vertices according to $\sigma$. The algorithm starts with $S = U = \emptyset$ and $T = V$, and runs till $U \cup T = \emptyset$. At each round of the algorithm, if the set $U$ is non-empty, the algorithm queries $T$ for neighbors of the last vertex $v$ that has been added to $U$, scanning $T$ according to $\sigma$. If $v$ has a neighbor $u$ in $T$, the algorithm deletes $u$ from $T$ and inserts it into $U$. If $v$ does not have a neighbor in $T$, then $v$ is popped out of $U$ and is moved to $S$. If $U$ is empty, the algorithm chooses the first vertex of $T$ according to $\sigma$, deletes it from $T$ and pushes it into $U$.

Observe that the DFS algorithm starts revealing a connected component $C$ of $G$ at the moment the first vertex of $C$ gets into (empty beforehand) $U$ and completes discovering all of $C$ when $U$ becomes empty again. We call a period of time between two consecutive emptyings of $U$ an epoch each epoch corresponds to one connected component of $G$.

Let $\varepsilon > 0$, $p = \frac{1-\varepsilon}{n}$, $k = \frac{2}{\varepsilon} \ln n$, and $N = \binom{n}{2}$.

(a) Let $\bar{X} = (X_i)_{i=1}^N$ be a sequence of independent Bernoulli($p$)-distributed random variables. Prove that with high probability $\bar{X}$ does not contain an interval of length $kn$ in which at least $k$ of the random variables $X_i$ take value 1.

(b) Show that with high probability all components in $G$ have size at most $k$. (Hint: Use (a) and DFS)

GOOD LUCK!

2
Algorithm 1.1 \textbf{Depth-First-Search}

\textbf{Input}: Graph $G(V, E)$ and an order $\sigma = v_1, v_2, \ldots, v_{|V|}$ on $V$

\textbf{Output}: Collection of connected components $\mathcal{C} = \{C_1, C_2, \ldots, C_t\}$

\begin{itemize}
  \item[] $T \leftarrow V$ // unvisited vertices
  \item[] $U \leftarrow \text{stack()}$ // vertices under exploration (last in, first out data structure)
  \item[] $i \leftarrow 0$
  \item[] \textbf{while} $U \cup T \neq \emptyset$ \textbf{do}
    \item[] \textbf{if} $U = \emptyset$ // the $i$-th component has been explored completely
      \item[] \hspace{1em} $i \leftarrow i + 1$
      \item[] \hspace{1em} $C_i \leftarrow \emptyset$ // vertices of $i$-th comp. that have been explored completely
      \item[] \hspace{1em} $j \leftarrow \min \{\ell : v_{\ell} \in T\}$
      \item[] \hspace{1em} $U \leftarrow \{v_{j}\}$ // move first such vertex from $T$ to $U$
      \item[] \hspace{1em} $T \leftarrow T \setminus \{v_{j}\}$
    \item[] \textbf{else} // there are still vertices under exploration in the $i$-th component
      \item[] \hspace{1em} $v \leftarrow \text{top}(U)$ // vertex last added to $U$ (without removing it from the stack)
      \item[] \hspace{1em} \textbf{if} $N(v) \cap T \neq \emptyset$ \textbf{then} // $v$ has neighbors in $T$
        \item[] \hspace{2em} $j \leftarrow \min \{\ell : v_{\ell} \in N(v) \cap T\}$
        \item[] \hspace{2em} $T \leftarrow T \setminus \{v_{j}\}$ // move first such neighbor from $T$ to $U$
        \item[] \hspace{2em} $U.\text{push}(v_{j})$
      \item[] \hspace{1em} \textbf{else} // $v$ has no neighbors, i.e. $v$ has been explored completely
        \item[] \hspace{2em} $U.\text{pop}(v)$ // move that vertex from $U$ to $C_i$
        \item[] \hspace{2em} $C_i \leftarrow C_i \cup \{v\}$
  \item[] \textbf{return} $\{C_1, C_2, \ldots, C_t\}$
\end{itemize}