ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institute of Theoretical Computer Science
Bernd Gärtner, Rasmus Kyng, Angelika Steger, David Steurer, Emo Welzl

Algorithms, Probability, and Computing — Exercises KW48 — HS23

General rules for solving exercises

- When handing in your solutions, please write your exercise group on the front sheet:

  Group A: Wed 14–16 CAB G 56

  Group B: Wed 14–16 CAB G 57

  Group C: Wed 16–18 CAB G 56

  Group D: Wed 16–18 CAB G 57

- This is a theory course, which means: if an exercise does not explicitly say "you do not need to prove your answer", then a formal proof is **always** required.

The following exercises will be discussed in the exercise class on November 29, 2023. These are "in-class" exercises, which means that we do not expect you to solve them before the exercise session. Instead, your teaching assistant will solve them with you in class.

Exercise 1

Let A be an $n \times n$ matrix with 0/1-entries. For $1 \leq i, j \leq n$ let $\epsilon_{i,j}$ be independent random variables, $\epsilon_{i,j} \in_{\text{u.a.r.}} \{-1, +1\}$. Let B be the random matrix with $b_{i,j} = \epsilon_{i,j} \cdot a_{i,j}$. In other words, to get B from A we randomly assign signs to the entries of A.

(a) Show that $\mathbb{E}[\det B] = 0$.

(b) Show that $\mathbb{E}[(\det B)^2] = \text{per}(A)$.

Exercise 2

Suppose that we have an algorithm for testing the existence of a perfect matching in a given graph, with running time at most $T(n)$ for any n-vertex graph.

(a) Explain how repeated calls to the algorithm can be used to find a perfect matching if one exists. Estimate the running time of the resulting algorithm.

(b) How can the algorithm be used for finding a maximum matching in a given graph?

Exercise 3

There is a close connection between counting algorithms and sampling algorithms. We have seen in the lecture how to count the number of perfect matchings in a graph (not very efficiently for general graphs) and here your task is to develop algorithms to sample a perfect matching uniformly at random. All the randomness you are allowed to use in this exercise is given by a stream of random bits and extracting one bit from the stream takes unit time.

Throughout, we let n denote the number of vertices in a graph. We assume access to a counting oracle that counts the number of perfect matchings in a graph in time $T(n)$.

(a) Given a positive integer N, how to efficiently sample a uniformly random number from the set $\{1, \dots, N\}$ by using the given stream of random bits? You should give a bound in big O notation on the number of random bits used in expectation.

(b) Show how to sample a uniformly random perfect matching in a given graph by using $O(n^2)$ calls to the counting oracle. You should use $O(n^2 \log n)$ random bits in expectation and your algorithm should run in expected time $O(T(n) \cdot \text{poly}(n))$.

(c) Show how to sample a uniformly random perfect matching in a given planar graph by using $O(n)$ calls to the counting oracle. You should use $O(n^2)$ random bits in expectation and your algorithm should run in expected time $O(nT(n))$. You can assume that $T(n) \in \Omega(n)$.