DM582 Exercises - Sheet 5

Mads Anker Nielsen Tobias Samsøe Sørensen

February 19, 2025

This document contains exercises from the course DM582 (spring 2025). Most exercises are from the book *Introduction to Algorithms, 4th edition* by Cormen, Leiserson, Rivest, and Stein (CLRS), the book *Algorithm Design, 1st edition* by J. Kleinberg and E. Tardos (KT), and the book *Discrete Mathematics and its Applications, 8th edition* by K. Rosen.

References to CLRS refer to the book *Introduction to Algorithms, 4th edition* by Cormen, Leiserson, Rivest, and Stein.

References to KT refer to the book *Algorithm Design, 1st edition* by J. Kleinberg and E. Tardos.

References to Rosen refer to the book *Discrete Mathematics and its Applications, 8th edition* by K. Rosen.

This document will inevitably contain mistakes. If you find some, please report them to your TA so that we can correct them.

Sheet 5

KT, Exercise 6

Exercise

One of the (many) hard problems that arises in genome mapping can be formulated in the following abstract way. We are given a set of n markers $\{\mu_1, \ldots, \mu_n\}$ — these are positions on a chromosome that we are trying to map — and our goal is to output a linear ordering of these markers. The output should be consistent with a set of constraints, each specified by a triple (μ_i, μ_j, μ_k), requiring that μ_j lie between μ_i and μ_k in the total ordering that we produce. (Note that this constraint does not specify which of μ_i or μ_k should come first in the ordering, only that μ_j should come between them.)

Now it is not always possible to satisfy all constraints simultaneously, so we wish to produce an ordering that satisfies as many as possible. Unfortunately, deciding whether there is an ordering that satisfies at least k' of the k constraints is an NP-complete problem (you don't have to prove this).

Give a constant $\alpha > 0$ (independent of n) and an algorithm with the following property. If it is possible to satisfy k^* of the constraints, then the algorithm produces an ordering of markers satisfying at least αk^* of the constraints. Your algorithm may be randomized; in this case, it should produce an ordering for which the expected number of satisfied constraints is at least αk^*.

KT, Exercise 7

Exercise

In Section 13.4, we designed an approximation algorithm to within a factor of 7/8 for the MAX 3-SAT Problem, where we assumed that each clause has terms associated with three different variables. In this problem, we will consider the analogous MAX SAT Problem: Given a set of clauses $C_1, ..., C_k$ over a set of variables $X = \{x_1, \ldots, x_n\}$,find a truth assignment satisfying as many of the clauses as possible. Each clause has at least one term in it, and all the variables in a single clause are distinct, but otherwise we do not make any assumptions on the length of the clauses: There may be clauses that have a lot of variables, and others may have just a single variable.

(a) First consider the randomized approximation algorithm we used for MAX 3-SAT, setting each variable independently to true or false with probability 1/2 each. Show that the expected number of clauses satisfied by this random assignment is at least $k/2$, that is, at least half of the clauses are satisfied in expectation. Give an example to show that there are MAX SAT instances such that no assignment satisfies more than half of the clauses.

(b) If we have a clause that consists only of a single term (e.g., a clause consisting just of x_1, or just of x_2), then there is only a single way to satisfy it: We need to set the corresponding variable in the appropriate way. If we have two clauses such that one consists of just the term x_i, and the other consists of just the negated term x_i, then this is a pretty direct contradiction. Assume that our instance has no such pair of "conflicting clauses"; that is, for no variable x_i do we have both a clause $C = \{x_i\}$ and a clause $C = \{\overline{x_i}\}$. Modify the randomized procedure above to improve the approximation factor from 1/2 to at least .6. That is, change the algorithm so that the expected number of clauses satisfied by the process is at least .6k.

(c) Give a randomized polynomial-time algorithm for the general MAX SAT Problem, so that the expected number of clauses satisfied by the algorithm is at least a .6 fraction of the maximum possible.

Exercise from course webpage

Exercise

This problem concerns a randomized algorithm for coloring graphs. Assume we have a graph, $G = (V, E)$, and colors, $\{1, 2, 3, 4\}$. Then $f : V \to \{1, 2, 3, 4\}$ is called a 4-coloring of G. We say that an edge $uv \in E$ is *good* with respect to f if $f(u) \neq f(v)$.

1. Suppose that each vertex independently gets color i with probability 1/4 for $i \in \{1, 2, 3, 4\}$. Let the random variable X be the number of edges in E that are good with respect to the coloring f. Show that $E[X] = \frac{3|E|}{4}$.

2. Use the result above and the probabilistic method to conclude that every graph has a 4-coloring f such that at least $\frac{3|E|}{4}$ of its edges

are good with respect to f.

3. Define a randomized algorithm that, for a given graph $G = (V, E)$, finds a 4-coloring f^* of V such that at least $\frac{3|E|}{4}$ of the edges of G are good with respect to f^*.

4. Follow the construction from MAX 3-SAT to compute a bound on the expected running time of your algorithm?

KT, Exercise 2

Exercise

Revisit this exercise to also find some upper bound on the probability of at least 1000 Democrats voting for the R candidate.

KT, Exercise 8

Exercise

Let $G = (V, E)$ be an undirected graph with n nodes and m edges. For a subset $X \subseteq V$, we use $G[X]$ to denote the subgraph induced on X — that is, the graph whose node set is X and whose edge set consists of all edges of G for which both ends lie in X.

We are given a natural number $k \leq n$ and are interested in finding a set of k nodes that induces a "dense" subgraph of G; we'll phrase this concretely as follows. Give a polynomial-time algorithm that produces, for a given natural number $k \leq n$, a set $X \subseteq V$ of k nodes with the property that the induced subgraph $G[X]$ has at least $\frac{mk(k-1)}{n(n-1)}$ edges. You may give either (a) a deterministic algorithm, or (b) a randomized algorithm that has an expected running time that is polynomial, and that only outputs correct answers.

Exercise from course webpage

Exercise

For a graph, $G = (V, E)$, a *spanning bipartite subgraph* G' of G is defined by a partition (V_1, V_2) of V, and the edges $E' \subseteq E$ that have an endpoint in both parts.

Consider the following randomized algorithm for finding a spanning bipartite subgraph of an arbitrary graph: Independently, for each vertex $v \in V$, decide uniformly at random if vertex v is in V_1 or V_2.

1. Give a lower bound on the expected number of edges m' in E' as a function of $m = |E|$.

2. How can you use your result to conclude that any graph has a spanning bipartite subgraph with $m' \geq m/2$?

3. Design a deterministic, polynomial-time algorithm for this problem, finding a spanning bipartite subgraph G' of any graph G, where $m' \geq m/2$.