ETH Zürich
Institute of Theoretical Computer Science
Prof. Rasmus Kyng
Prof. Johannes Lengler
Prof. Vera Traub

FS 2025

Algorithmen und Wahrscheinlichkeit
Peer Grading Exercise 4

Submit by Moodle () until 16:00 on 15.05.2025.

Exercise 1 – *Finding k disjoint triangles*

Provide a randomized algorithm running in time $O(2^{O(k)}\text{poly}(|V|)\log(1/\delta))$, for the following problem: given a graph $G = (V, E)$ and an integer k, decide whether a graph G has k vertex-disjoint triangles. When G does not contain k vertex-disjoint triangles, the algorithm should always output NO, if the graph G contains k such triangles, the algorithm should output YES with probability at least $1 - \delta$.

Formally, we want to check if there is k (disjoint) set of vertices $\{v_{1,1}, v_{1,2}, v_{1,3}\}, \ldots \{v_{k,1}, v_{k,2}, v_{k_3}\}$ s.t. for all $i \in [k]$ the vertices $\{v_{i,1}, v_{i,2}, v_{i,3}\}$ form a triangle in G.

(*Hint:* Ideas similar to ones used for the LONG-PATH problem in lecture can be used here.)

Solution 1

As in the color-coding based solution for the LONG-PATH problem, we start by coloring all vertices of the graph at random into $3k$ colors (each vertex gets assigned each of the $3k$ colors with the same probability, independently). For the positive instance, if the graph indeed contains at least k disjoint triangle, the probability that all $3k$ vertices of those triangles are assigned different colors is at least $(3k)!/(3k)^{3k}$. Since $e^{-n} \leq n!/n^n$ for any n, this probability is at least $\exp(-3k)$.

If we happen to have succesfully colored the $3k$ vertices in k disjoint triangles into different colors, we can find those $3k$ vertices by a simple dynamic programming. Specifically, we keep a DP table $F[S]$, such that for all subsets $S \subset [3k]$ of colors, with $|S|$ divisible by 3, $F[S]$ is 1 if there are $|S|/3$ triangles using colors from S, such that all vertices of those $|S|/3$ triangles have different colors.

To fill in the table $F[S]$ we can start by preparing a table $H[\kappa_1, \kappa_2, \kappa_3]$, which, for all triples of colors $\{\kappa_1, \kappa_2, \kappa_3\}$ is equal to 1 whenever there is a triangle with one vertex of color κ_1, another vertex of color κ_2 and the last vertex of color κ_3. We can fill in this table in time $O(n^3)$ by iterating over all triples of vertices.

Using this, we can fill the F table. We start by setting $F[\emptyset] = 1$, and to compute $F[S]$ for a set $S \neq \emptyset$, assuming that we already have computed $F[S']$ for all $|S'| < |S|$, we just iterate over all triples $\{\kappa_1, \kappa_2, \kappa_3\} \subset S$, and check if $F[S \setminus \{\kappa_1, \kappa_2, \kappa_3\}] = 1$ and if there is a triangle with vertices of colors $\kappa_1, \kappa_2, \kappa_3$ (which is a look-up in the H table).

The total running time of this dynamic-programming algorithm can be roughly upper bounded by a constant times

$$\sum_{S \subset [3k]} 2^{|S|} = \sum_{j \leq 3k} \binom{3k}{j} 2^j = 3^{3k}.$$

This gives a total running time for the sub-problem of finding a colorful k-triangles as $O(2^{O(k)} + n^3)$. This algorithm succeeds with probability at least e^{-3k} when the graph contains k disjoint triangles, always correctly says "NO" when the graph contains no k disjoint triangles.

We can repeat the algorithm now $m = O(\exp(3k)\ln(1/\delta))$ times independently, and output "YES" only if at least one of the runs outputs "YES". Clearly, when the graph contains no k-disjoint triangles, this algorithm is still always correct. On the other hand, if the graph contains the

required configuration, the probability that the algorithm fails all m times is, by independence, at most

$$(1 - \exp(-3k))^m \leq \exp(-e^{-3k})^m = \exp(-e^{-3k}m) \leq \exp(-\ln(1/\delta)) \leq \delta.$$