2MMD30: Graphs and Algorithms Lecture 11 by Jesper Nederlof, 18/03/2016

Solutions to exercises of lecture 11

Note: In all the exercises below your algorithms may have one-sided constant error probability.

Excercise 11.1. A triangle in an undirected graph $G = (V, E)$ is a triple of vertices (u, v, w) with a $(u, v), (v, w), (u, w) \in E$. A k-triangle-packing is a collection of triangles $T_1, \ldots, T_k$ that are mutually disjoint.

- Give an $O^*(k^{O(k)})$ time algorithm that given (G, k) determines whether G has a k-triangle packing. Hint: assign numbers $c(v) \in \{1, \ldots, 3k\}$ uniformly and independently at random to every vertex v and look for triangle packings where all triangles (u, v, w) satisfy $c(w) = c(v) + 1 = c(u) + 2 = 3i$ for different integers i.

- Give an $O^*(2^{O(k)})$ time algorithm that given (G, k) determines whether G has a k-triangle packing. Hint: assign numbers $c(v) \in \{1, \ldots, 3k\}$ uniformly and independently at random and look for colorful triangle packing using dynamic programming or Inclusion-Exclusion.

Solution: Assign the $c(v)$ as stated. Now we are looking for a sequence $(v_1, \ldots, v_{3k}) \in V$ of distinct vertices such that $v_{3i-2}, v_{3i-1}, v_{3i}$ is a triangle for every $i = 1, \ldots, k$ and $c(v_i) = i$. Such a sequence exists if and only if for every $i = 1, \ldots, k$ there is a triangle $(v_{3i-2}, v_{3i-1}, v_{3i})$ with $c(v_{3i-2}) = 3i - 2$, $c(v_{3i-1}) = 3i - 1$, $c(v_{3i}) = 3i$. Thus we can detect whether such sequence exists, by determining the latter condition by checking all triples of vertices with the correct colors and seeing if any forms a triangle, for every k. This is clearly polynomial time. Let $T_1, \ldots, T_k$ be a triangle packing and let $v_1, \ldots, v_{3k}$ be an ordering of the vertices in the triangles such that vertices in the same triangle are ordered consecutively. Then $\Pr[\forall i : c(v_i) = v_i] = \frac{1}{(3k)^{3k}}$. Thus if we try $(3k)^{3k} = O(k^{O(k)})$ coloring $c(v)$, one will satisfy $\forall i : c(v_i) = v_i$ and we will detect a triangle packing.

For the second part, the approach is the same as above as in the lecture except that we need to determine whether a colorful triangle triangle packing exists. The observation is again that the colorfulness automatically implies disjointness, so a colorful triangle packing is the same as a color triangle tuple There are two ways of doing this

Dynamic Programming: For every subset $X \subseteq \{1, \ldots, 3k\}$ and integer $1 \leq i \leq k$ define $T[i, X]$ to be **true** iff there exist i triangles $T_1, \ldots, T_i$ such that $X = \{c(v) : v \in \bigcup_{j=1}^{i} T_j\}$ i.e. X

is the set of colors occurring in the triangles. Then we see that

$$T[i,X] = \begin{cases} \textbf{false} & \text{if } i = 0, X \neq \emptyset, \\ \textbf{true} & \text{if } i = 0, X = \emptyset, \\ \bigvee_{(u,v,w)\in V^3} \Big((u,v,w) \text{ is a triangle of } G\Big) \wedge T[i-1, X \setminus \{c(u),c(v),c(w)\}] & \text{if } i > 1, \end{cases}$$

and we can compute $T[k, 1, \ldots, 3k]$ in the usual way in $O^*(2^{3k})$ time and this by definition tells us whether a colorful triangle packing exists.

Inclusion Exclusion: Define $U = \{(T_1, \ldots, T_k) : T_i \text{ triangle of } G, \text{ for every } i\}$, for $j = 1, \ldots, 3k$ define $P_j = \{(T_1, \ldots, T_k) \in U : \exists v \in \bigcup_{i=1}^{k} T_i \text{ such that c(v)=j}\}$. Then $|\bigcap_{i=1}^{3k} P_i|$ is the number of colorful triangle tuples. For $F \subseteq \{1, \ldots, 3k\}$ we have that $|\bigcap_{i\in F} \overline{P_i}|$ is the number of triangle tuples without vertices with a color in F. This is exactly $TR(F)^k$ where TR is the number of triangle in $G[\{v \in V : c(v) \notin F\}]$ and thus can be computed in polynomial time.

Excercise 11.2. Find an algorithm that, given a directed acyclic graph, finds a topological ordering in polynomial time.

Solution: In a directed acylic graph there must always be a vertex without in-edges since otherwise we can keep walking on the edges forever and thus must visit a vertex twice at some moment implying a cycle. We can find such a vertex without in-edges in linear time and safely put it in the beginning of the ordering and remove it from the graph. If we keep iterating this until now vertices are left we spend at most linear time per iteration (and thus per vertex since we remove one vertex in every iteration) so this runs in quadratic time.

Excercise 11.3. Change Line 1-3 in Algorithm `kpath1` to get an algorithm running in time $O^*(k!)$.

Solution: The new algorithm is as follows:

Algorithm `kpath1`(G, k) G is directed

Output: Whether G has a simple path on k-vertices, with constant one-sided error probability.

1: **for** $i = 1 \ldots k!$ **do**
2: Pick a permutation $c : V \to \{1, \ldots, |V|\}$ uniformly at random.
3: Let $G' = (V, E')$ where $E' = \{(u, v) \in E : c(u) < c(v)\}$
4: **if** `kpathDAG`(G', k) **then return true**
5: **return false**

Then G' is still a directed acyclic graph since c already gives the topological ordering. And if $p_1, \ldots, p_k$ is the k-path, the probability that $c(p_1) < c(p_2) < \ldots < c(p_k)$ is $1/(k!)$ (to see this, note that the exact values assigned to these vertices does not matter, only their order matters).

Excercise 11.4. Graphs $G = (V, E), P = (W, F)$ are isomorphic if there exists a bijection $\pi : V \leftrightarrow W$ such that $(u, v) \in E$ if and only if $(\pi(u), \pi(v)) \in F$. In this exercise we assume P is connected.

- Suppose that $|P| = k$. Show that we can determine in $O^*(k!)$ time whether P is isomorphic to one of the connected components of G.

In the Subgraph Isomorphism problem, we are given graphs $G = (V, E), P = (W, F)$ and are asked whether P (the 'pattern') is isomorphic to subgraph of G (recall that a subgraph is obtained by removing vertices and edges).

- Show that the Subgraph Isomorphism problem is NP-complete by picking a specific n-vertex graph P for every n.

Suppose G has maximum degree d.

- Suppose that P is isomorphic to a subgraph of G and G' is obtained from G by removing every edge with probability $1/2$. Lower bound the probability that P is isomorphic to a connected component of G'.

- Show how to solve the Subgraph Isomorphism problem in $O^*(2^{kd}k!)$ time, if $|P| = k$ and G has maximum degree d.

Solution: For every connected component, Simply try all bijections π and check if any satisfies the condition.

If P is a cycle on n vertices, this is the Hamiltonian cycle problem.

If P is isomorphic to a subgraph of G, say (X, E'), P will be isomorphic to a connected component of G' if all edges in E' are kept, all edges in $((X \times X) \cap E) \setminus E'$ are removed and all edges in $E \cap (X \times (V \setminus X))$ are removed. These three edge sets are disjoint, and the sum of their sizes equals the number of edges incident to X, which is upper bounded by $|X|d = kd$. Since all edges are removed or kept, we do the correct thing for each edge with probability $1/2$ and we get probability $(1/2)^{kd}$.

Obtain G' from G as stated and check whether G' contains P as connected component as stated. Repeat 2^{kd} to boost the probability as usual.

Excercise 11.5. Suppose we would like to solve an instance $(G = (V, E), k)$ of FVS where $|V| = n$ and $k = n/2$. We can directly run the $O^*(4^k)$ time algorithm but that is not too impressive. In this question we try to use the $O^*(4^k)$ time algorithm in this setting in a more clever way.

- Why is running the $O^*(4^k)$ time algorithm not too impressive?

- Consider the following problem: given an instance $(G = (V, E), k)$ of FVS and a set W, does G have a FVS X such that $|X| \leq k$ and $W \subseteq X$. Use the $O^*(4^k)$ time algorithm to solve this problem in $O^*(4^{k-|W|})$ time.

- Suppose W is picked uniformly at random from the set of all $(n/4)$-sized subsets of V [1]. If X is a FVS of size $n/2$, lower bound the probability that $W \subseteq X$. Use that $\binom{x}{x/2} \geq 2^x/x$

- Give an algorithm that determines whether a FVS of size $k = n/2$ exists in time $O^*(\binom{n}{n/4}) \leq O^*(2^{.82n})$.

[1] this can be easily done in polynomial time, but we will not concern ourselves with this technical issue here

Solution: $O^*(4^{n/2}) = O^*(2^n)$ which we can also get by trying all subsets of V.

We simply need to determine whether $G[V \setminus W]$ has a FVS of size $k - |W|$

$$\Pr[W \subseteq X] = \frac{\binom{n/2}{n/4}}{\binom{n}{n/4}} \geq \frac{2^{n/2}/n}{\binom{n}{n/4}}.$$

Pick a random W of size $n/4$ and determine whether a FVS containing W exists in time $O^*(4^{k-|W|}) = O^*(4^{n/4})$. Repeat $\frac{\binom{n}{n/4}}{2^{n/2}/n} =$ times to boost the probability computed above. The running time will be

$$O^*\left(4^{n/4}\frac{\binom{n}{n/4}}{2^{n/2}/n}\right) = O^*\left(\binom{n}{n/4}\right)$$

Excercise 11.6. Solve the k-path problem in $O^*((2e)^k)$ time and polynomial space using Inclusion-Exclusion. Hint: To look for a colorful k-path, use as universe the set of all walks with $k-1$ edges (e.g. on k vertices).

Solution: As stated, use for U the set of all walks in G with $k-1$ edges. For every $i \in \{1, \ldots, k\}$ define P_i to be the set of all walks in G with $k-1$ edges that visit a vertex v with $c(v) = i$. Then $|\bigcap_{i=1}^{k} P_i|$ is the number of colorful walks (= the number of colorful paths) and $|\bigcap_{i \in F} \overline{P_i}|$ is the number of walks on $k-1$ edges in $G[\{v \in V : c(v) \notin F\}]$ which can be computed in polynomial time as $\sum_{s,t \in V} w_F(s,t,k)$, where $w_F(s,t,k)$ was computed in polynomial time in Lecture 6, Section 6.