Extra Practice Exercises Solutions

Excercise 1. Given undirected graph $G = (V, E)$, find a pair $u, v \in V$ with $u \neq v$ maximizing $N(u) \cap N(v)$ in $O(n^\omega)$ time, where ω is the matrix multiplication constant. Recall that $N(v)$ denotes the set of all neighbors of v (so not v itself).

Solution. If $v_1, \ldots, v_n$, A is the adjacency matrix (so $a_{ij} = 1$ if and only if $(v_i, v_j) \in E$) and $B = A^2$, then recall that $b_{ij} = \sum_{k=1}^{n} a_{ik}a_{kj}$. Thus $|N(v_i) \cap N(v_j)| = b_{ij}$. Therefore we can compute B in $O(n^\omega)$ time and return $i < j$ such that b_{ij} is maximum.

Excercise 2.

- Give an algorithm taking as input an undirected graph G and a Feedback Vertex Set (FVS) F of G, and outputs a tree decomposition of G of width $|F| + O(1)$ in polynomial time.
- Suppose you have an algorithm `fvstw`$(G, (X, T))$ that given an undirected graph G and a tree decomposition (X, T) of G of width w, computes a minimum size FVS in time $O^*(3^w)$. Give an algorithm that uses `fvstw`$(G, (X, T))$ as a blackbox, takes as input a graph G and integer k, and determines whether G has a FVS of size at most k in $O^*(3^k)$. Hint: use iterative compression.

Solution. Recall that in Exercise 7.2 we constructed a tree decomposition of a tree. Use the same tree decomposition here for every connected component of $G[V \setminus F]$, which needs to be a tree since F is a FVS. This gives tree decompositions $(X^1, T^1), \ldots, (X^l, T^l)$ for each of the l connected components of $G[V \setminus F]$. Now we add to every bag the set F and we connect all trees $T^1, \ldots, T^l$ in an arbitrary way to a tree T. This gives a tree decomposition of G because : (i) every vertex occurs in some bag (ii) every edge occurs in some bag (the edges of $G[V \setminus F]$ are already contained in one of the tree decompositions (X^1, T^l), the edges incident to F are contained in the tree decomposition since F is in every bag) (iii) for every $v \in V$ the set of bags containing v induces a connected subtree of T since if $v \in F$ it is the whole tree and if $v \notin F$ it only occurs in T_i for some i and since T_i is a tree decomposition of the connected component of $G[V \setminus F]$ containing v, the set of bags containing v must induce a connected subtree. Note that the maximum bag size of this decomposition if $|F| + 2$ (so it has width $|F| + 1$).

For the iterated compression we use the following algorithm. Note that it is the same as Algorithm 7 from Lecture 5 except that we have a different compression step.

Algorithm $\texttt{FVS}(G = (V, E), k)$
Output: Whether G has a feedback vertex set of size at most k
Let $V = \{v_1, \dots, v_n\}$
Let $X = \{v_1, \dots, v_k\}$
for $i = k+1, \dots, n$ **do**
  $X \leftarrow X \cup v_i$ — X is a FVS of $G[\{v_1, \dots, v_i\}]$ of size at most $k+1$
  start compression
  Construct a TD $(\hat{X}, T)$ of $G[\{v_1, \dots, v_i\}]$ of width at most $|F| + O(1)$ as outline above.
  $X = \texttt{fvstw}(G, (\hat{X}, T))$
  end compression
  if $|X| = k+1$ **then return false** — check whether the compression was successful
return true

Algorithm 1: Algorithm for Feedback Vertex Set.

Excercise 3. A connected vertex cover (CVC) of a graph $G = (V, E)$ is a vertex cover $X \subseteq V$ such that $G[X]$ is connected. Give an algorithm that takes as input a graph $G = (V, E)$ and integer k and determines in $O^*(c^k)$ time whether G has a connected vertex cover of size at most k, for some constant c. You may use as blackbox an algorithm $\texttt{st}(G = (V, E), T, l)$ that solves the Steiner Tree problem in $O^*(2^{|T|})$ time, i.e. it determines whether there exists $T \subseteq Y \subseteq V$ with $|Y| \leq l$ with $G[Y]$ connected.

Solution. Use the following algorithm:

Algorithm $\texttt{cvc}(G = (V, E), X, k)$
Output: Whether G has a connected vertex cover of size at most k containing the set X.
if X is a vertex cover of G **then**
  return $\texttt{st}(G, X, k)$
if $k \leq 0$ **then return false**
Let $(u, v) \in E$ such that $u \notin X$ and $v \notin X$.
return $\texttt{cvc}(G[V \setminus u], k-1) \vee \texttt{cvc}(G[V \setminus v], k-1)$

Algorithm 2: $O(4^k k(n+m))$ time algorithm for detecting connected vertex covers of size at most k.

If X is a vertex cover of G, then we only need to worry about the requirement that $G[X]$ is connected. Then $\texttt{st}(G, X, k)$ determines whether there exists a superset Y of X such that such that $|Y| \leq k$ and $G[Y]$ is connected, which is exactly the set of connected vertex covers that are a superset of X.

If $(u, v) \in E$ such that $u \notin X$ and $v \notin X$, then we know that for every vertex cover Y either $u \in Y$ and $v \in Y$ (or both).

Excercise 4. Give an $O(n^2)$ time algorithm that takes as input three matrices $A, B, C \in \mathbb{Z}_2^{n \times n}$ with the following properties:

- If $AB = C$, i.e. $\forall i, j : c_{ij} \equiv_2 \sum_k a_{ik} b_{kj}$, the algorithm always outputs **true** ,

- If $AB \neq C$, the algorithm outputs **false** with constant probability.

Hint: Pick $x \in \mathbb{Z}_2^n$ uniformly at random and study Cx and ABx, use the rank-nullity theorem to bound the probability of false positives.

Solution. Pick $x \in \mathbb{Z}_2^k$ uniformly at random. Compute Cx and $A(Bx)$ (which is easily naïvely done in $O(n^2)$ time by following the definition of matrix vector multiplication). We have that if $AB = C$ then $ABx = C$. On the other hand, if $AB \neq C$ then $AB - C$ has rank at least 1, and $\mathsf{nul}(AB - B) \leq n - 1$. Then we see that

$$\begin{aligned}
\Pr_x[ABx = Cx] &= \Pr_x[(AB - C)x = 0] \\
&= \Pr_x[x \text{ in null space of } AB - C] \\
&\leq 2^{n-1}/2^n \leq 1/2.
\end{aligned}$$