Tutorial on FPT algorithms

1. Polynomial kernel for Feedback Edge set in tournaments
Let $G$ be a directed graph and $F$ be a subset of edges of $T$. We denote by $G \otimes F$ the graph obtained from $G$ after reversing each edge of $F$. $F$ is a feedback edge set if when we delete all the edges in $F$, $G$ becomes a directed acyclic graph.

(a) Let $G$ be a directed graph and $F$ a subset of edges of $G$. Show that $F$ is a minimal feedback edge set if and only if $F$ is a minimal set of edges such that $G \otimes F$ is a directed acyclic graph.

(b) A tournament is a directed graph $T$ such that for each pair of vertices $(u, v)$, exactly one of the edges $uv$ and $vu$ is in $T$. In the Feedback Arc Set in Tournaments problem we are given a tournament $T$ and a non-negative integer $k$. The objective is to decide whether $T$ has a feedback edge set of size at most $k$. Find two simple reduction rules that will permit to get a polynomial kernel.

Solution: Section 2.2.2 of the textbook parametrized algorithms.

2. Closest String Problem
The closest string is an NP-Hard problem which tries to find the geometrical center of a set of input strings.
Given $n$ strings $s_1, \ldots, s_n$ each of length $m$ and an integer $d$, find a string $x$ of length $m$ such that $d_H(x, s_i) \leq d$ for $i = 1, \ldots, m$, where $d_H$ denotes the Hamming distance. Find an FPT algorithm for Closest String Problem running in time $O(nm + nd(d+1)^d)$.

Solution: Section 3.5 of the textbook.

3. Randomized algorithm for Feedback Vertex set
A feedback vertex set is a set of vertices $S$ such that $G \setminus S$ is a forest.

(a) Let $G$ be a multigraph with minimum degree at least 3 and let $S$ be a feedback vertex set of $G$. Prove that more than half of the edges are incident to at least one vertex of $S$.

(b) Prove that there exists a polynomial-time randomized algorithm that, given a Feedback Vertex Set instance $(G, k)$, either reports a failure or finds a feedback vertex set in $G$ of size at most $k$. Moreover, if the algorithm is given a yes-instance, it returns a solution with probability at least $4^{-k}$.

(c) Prove that there exists a randomized algorithm that, given a Feedback Vertex Set instance $(G, k)$, in time $4^k \cdot n^{O(1)}$, either reports a failure or finds a feedback vertex set in $G$ of size at most $k$. Moreover, if the algorithm is given a yes-instance, it returns a solution with constant probability.

Solution: Section 5.1 of the textbook.
4. Triangle Packing
Given a graph $G$ and an integer $k$, the goal is to decide if $G$ contains $k$ vertex-disjoint triangles. Design a randomized FPT algorithm for this problem.

**Sketch of solution:** Choose a random coloring $V(G) \rightarrow [3k]$. Check if there is a colorful solution, where the $3k$ vertices of the $k$ triangles use distinct colors. (Use DP on subsets of size $3^i$, $i = 0, \ldots, k$). Success probability $\frac{(3k)!}{(3k)^k} \geq e^{-3k}$. Hence, we can achieve constant error probability by repeating the algorithm $e^{3k}$ times.

**Derandomization:** $3k$-perfect family of functions instead of random coloring.

5. Polynomial kernel for the Connected Vertex Cover Problem
In the Connected Vertex Cover Problem, we are given a graph $G$ and an integer $k$, and the objective is to decide if $G$ contains a vertex cover $S$ of size at most $k$ and such that $G[S]$ is connected (where $G[S]$ is the subgraph of $G$ induced by $S$).

(a) Give a simple graph for which the kernelization procedure seen in class for Vertex-Cover fails for Connected Vertex Cover.

(b) Show that Connected Vertex Cover admits a kernel with at most $2^k + O(k^2)$ vertices.

**Solution:**

(a) The kernelization consists in applying these two rules as much as possible:

(R1) If a vertex has degree 0, delete it.
(R2) If a vertex has degree at least $k + 1$, delete it (and put it in the solution) and decrease $k$ by 1.

Recall that after applying these two reduction rules, Vertex Cover has a kernel with $k^2$ edges and $k^2 + k$ vertices, that is a YES instance has at most $k^2$ edges and $k^2 + k$ vertices.

The problem is with the first rule that might delete some vertex useful to make the solution connected. For example: take two vertices $a$ and $b$ both adjacent to $k + 1$ vertices and add a vertex $c$ adjacent to $a$ and $b$. Applying the second rules leads to the deletion of $a$ and $b$, and then the first rule leads to the deletion of $c$, while $\{a, b, c\}$ is a solution.

(b) Let $G$ be a YES-instance.
Delete vertices of degree 0 in $G$, they are useless.
Let $X$ be the set of vertices of degree at least $k + 1$. Each vertex of $X$ must belong to any solution, so $|X| \leq k$.
Let $Y$ be the set of vertices of degree 0 in $G - X$, and let $Z = G - (X \cup Y)$. Observe that $Y$ is the set of vertices that would have been deleted by (R1).
$Z$ must have a vertex cover of size at most $k$ (actually of size at most $k - |X|$),
and vertices in $Z$ has degree at most $k$. So $Z$ has at most $k^2$ edges, and since vertices in $Z$ have degree at least 1, $|Z| \leq 2k^2$.

Now, as explained in the previous question, the only purpose for adding a vertex of $Y$ in the solution, is to make it connected. Hence, if $u$ and $v$ are two vertices in $Y$ with the same neighborhood (which is included in $X$), we can delete one of them. Hence, we keep at most $2^{|X|} \leq 2^k$ vertices of $Y$: at most one for each subset of $X$.

All together, and after reductions (delete the vertices of degree 0 in the original graph, then delete vertices in $Y$ that have the same neighborhood), either we return NO, or the graphs has at most $2^k + k + 2k^2$ vertices.

6. **Iterative compression for Feedback Vertex Set in tournaments**

*Feedback Vertex Set* in tournament is the following problem:

*Input*: A tournament $T$ and an integer $k$

*Question*: Is there $S \subseteq V(T)$ such that $T \setminus S$ is acyclic?

The goal of the exercise is to use iterative compression to get a $2^k \cdot n^{O(1)}$-time algorithm for FVS in tournaments.

(a) Design a $3^k \cdot n^{O(1)}$-time algorithm using a simple branching.

(b) Define the problems **FVS Compression** and **Disjoint FVS** in tournaments.

(c) Prove that if we can solve **Disjoint FVS** in tournaments in time $n^{O(1)}$, then we can solve FVS in tournaments in time $2^k \cdot n^{O(1)}$.

(d) Design a polynomial-time algorithm to solve **Disjoint FVS** in tournament.

**Solution**: Section 4.2 of the textbook.