RWTH Aachen **SS 2014**
Lehrgebiet Theoretische Informatik Problem Set 1
Reidl—Rossmanith—Sánchez—Sikdar 16.10.2014

Parameterized Algorithms Tutorial

Tutorial Exercise T1

The Independent Set problem is defined as follows. Given a graph $G = (V, E)$ and an integer k, is ther a set S of size k such that for all $u, v \in S$ where $u \neq v$ it holds $uv \notin E(G)$? Is Independent Set restricted to graphs of maximal degree d, where d is a constant, fixed paramater tractable parameterized by the size of the solution k?

Solution

The problem is in FPT. One can design an FPT algorithm as follows. Take any node v. It will have degree d or lower. We know that either v or one of its neighbors must be in a maximal independent set. We branch over every possibility by taking a node of $N[v]$ into the independent set, deleting it and all of its neighbors and recursively solving the remaining graph. This gives us an algorithm with running time $O((d + 1)^k \cdot n)$.

Tutorial Exercise T2

Since any planar graph is has a four coloring, any instance of the Planar Independent Set problem is guaranteed to have a solution of at least size $n/4$. Is the above guarantee version of the problem fixed parameter tractable paramertized by the solution size k?

Solution

If k is smaller than $n/4$ we can say yes. If $k > n/4$ it follows that $n < 4k$. This means that using any brute force algorithm on this instance will suffice.

Tutorial Exercise T3

The Cluster Vertex Deletion problem is defined as follows: given a graph $G = (V, E)$ and an integer parameter k, does there exist a set S of size at most k such that $G[V \setminus S]$ consists of a collection of disjoint cliques. The cliques are disjoint in the sense that they do not share vertices and/or edges and there is no edge with one endpoint in one clique and the other in a different clique. Design an algorithm that runs in FPT-time w.r.t. k as parameter.

Solution

A graph is precisely a cluster graph when it does not contain any induced paths of length three. Thus we can solve solve the problem in FPT time as follows. Either we are done, or we find an induced path of length three. One of these three nodes can not be in the solution. We branch over their deletion. This gives us an $O(3^k \cdot n^4)$ algorithm.

Homework H1

The Triangle Vertex Deletion problem is defined as follows. Given a graph $G = (V, E)$ and an integer parameter k, are there k vertices whose deletion results in a graph with no cycles of length three? Show that this problem is fixed-parameter tractable. What is the running time of your algorithm? Is there some easy way to improve the running time?

Solution

The idea now is to branch on the vertices of a triangle. Let (u, v, w, v) be a three cycle in G. We recurse on the instances $(G - u, k - 1)$, $(G - v, k - 1)$ and $(G - w, k - 1)$. The branching vector is $(1, 1, 1)$ and the running time is $3^k \cdot \mathrm{poly}(|G|)$.

Homework H2

This exercise is concerned with the fixed parameter tractable algorithm for the Closest String problem that was discussed in class. Recall that an input to this problem consists of n strings $s_1, \ldots, s_n \in \Sigma^L$ of length L each and an integer parameter k. The question is whether there exists a string $s \in \Sigma^L$ such that $d(s, s_i) \leq k$, for all $1 \leq i \leq n$. Explain this algorithm and analyze its running time. You should probably start by asking yourself the following question: How many differences can there be between two strings s_1 and s_2, such there can exist a string s such that $d(s, s_1) \leq k$ and $d(s, s_2) \leq k$?

Solution

First observe that if there exists a string $s \in \Sigma^L$ such that $d(s, s_i) \leq k$ for all i, then for pairs of distinct strings s_i, s_j, we have

$$d(s_i, s_j) \leq 2k.$$

One can interpret this as follows. Consider the balls B_i and B_j of strings that are at Hamming distance at most k from s_i and s_j, respectively. A string s that is at distance at most k from both s_i and s_j must be in both these balls, meaning that these balls must intersect. This happens if and only if $d(s_i, s_j) \leq 2k$.

One could imagine trying to obtain a closest string s as follows. We first set $s = s_1$. Let s_i with $2 \leq i \leq n$ be the first string such that $d(s_i, s_i) > k$. Then by our observation, these two strings differ in at most $2k$ positions (if we assume that such a string exists in the first place). Given *any* set of $k + 1$ positions where s and s_i disagree, we have to make s and s_i agree in at least one of them. It is important to note that this fact holds irrespective of *which* set of $k + 1$ positions we choose. We therefore choose some set of $k + 1$ positions and create $k + 1$ instances of the problem by modifying s in the appropriate place in each of these instances. How does the parameter change? Every time we modify s, we increase the Hamming distance between s and s_1 by one. This is the crucial observation. Hence in each of these $k + 1$ recursive calls, we can modify at most $k - 1$ additional positions, since $d(s_1, s) = 1$. The size of the recursive tree is clear: each node has degree at most $k + 1$ and the depth of the tree is at most k. The total size is $((k + 1)^{k+1} - 1)/k$. The branching vector of this recursion is $(k - 1, k - 1, \ldots, k - 1)$ ($k + 1$ times).

One should verify that there exists such a string s if and only if there exists at least one branch of the recursion tree which finds such a string.