# Fine-grained Parameterized Algorithms

FPA · SoSe-2024 · tcs.uni-frankfurt.de/parameterized/ · 2024-07-03 · 2d88037

## Problem set 2: Bounded search trees

**Overview:** In this problem set, you develop FPT algorithms using the „trivial“ method as well as using the bounded search tree method.

**Instructions:** For each skill, select **exactly one** problem below and submit your solution in Moodle; in your submission, make sure to repeat the problem that you are solving. The problems are roughly ordered by difficulty, choose a problem that you find non-trivial and interesting. (You are encouraged to try the other problems as well and ask us for feedback.)

---

!! **Skill-2a. „Trivial“ FPT:** *I can design FPT algorithms by arguing that the optimal solution is large.* (This is similar to Skill-1b.)

**2.1 Independent set in regular graphs.** Recall that the independent set problem asks whether a given graph contains an independent set of size at least $k$. Let $r$ be a constant positive integer. Show that the independent set problem, when parameterized by $k$, is fixed-parameter tractable on $r$-regular graphs. *Bonus:* Show that this remains true if $r$ is part of the input and we parameterize by $k + r$.

**2.2 Feedback vertex set in regular graphs.** Given an undirected graph $G = (V, E)$, a subset of vertices $X \subseteq V$ is a *feedback vertex set* if removing $X$ and all edges incident to $X$ from $G$ yields a forest. Show that the problem of deciding whether a graph has a feedback vertex set of size at most $k$ is fixed-parameter tractable on regular, undirected graphs. (Note: Unlike 2.1, the degree is not assumed to be constant here!)

---

!! **Skill-2b. Bounded search trees:** *I can design FPT algorithms using bounded search trees.* (For more context, see Section 3.1 in Cygan et al.)

**2.3 Clique in regular graphs.** Recall that the clique problem asks whether a given graph contains a clique of size at least $k$. Let $r$ be a constant positive integer. Show that the clique problem, when parameterized by $k$, is fixed-parameter tractable on $r$-regular graphs. *Bonus:* Show that this remains true if $r$ is part of the input and we parameterize by $k + r$. (Note: This problem looks similar to 2.1, but the solution is quite different.)

**2.4 Feedback vertex set.** Show that every graph on $n$ vertices of minimum degree at least 3 contains a cycle of length at most $2\lceil \log n \rceil$. Use this to design a $(\log n)^{O(k)} \cdot n^{O(1)}$-time algorithm for the feedback vertex set problem on undirected graphs. Does this runtime bound satisfy the conditions for an fpt-algorithm, that is, can it be bounded by $f(k) \cdot n^{O(1)}$ for some computable $f$?

🌰🌰🌰 **2.5 Chordal completion.** A graph is *chordal* if it does not contain a cycle on at least four vertices as an induced subgraph. That is, for every cycle of length at least four, there is at least one edge in the graph between two vertices of the cycle that is not in the cycle. Such an edge is a *chord*. Hence the name.

A *triangulation* of a graph $G = (V, E)$ is a set of edges $E' \subseteq \binom{V}{2}$ such that $(V, E \cup E')$ is chordal.

Consider the chordal completion problem: Given a graph and an integer $k$, decide whether $G$ has a triangulation of size at most $k$.

a) Give a $k^{O(k)}n^{O(1)}$-time algorithm for the problem.

b) Show that there is a bijection between inclusion-wise minimal triangulations of a cycle of length $\ell$ and binary trees with $\ell - 2$ internal nodes. From this, conclude that a cycle on $\ell$ vertices has at most $4^{\ell-2}$ inclusion-wise minimal triangulations.

c) Use the previous point to design an algorithm for the problem running in time $2^{O(k)} \cdot n^{O(1)}$.

**2.6 Max leaf spanning tree.** Consider the following problem: Given a connected graph $G$ and an integer $k$, decide whether there is a spanning tree of $G$ with at least $k$ leaves. Design an algorithm that solves this problem in time $4^k \cdot n^{O(1)}$.

---

**Additional problems.** Solving these problems is required to fully understand the course material. However, you do not need to submit the solution in writing.

easy **2.7 Vertex-cover in low-degree graphs.** Prove that the vertex-cover problem can be solved optimally in polynomial time on graphs of maximum degree at most 2.