Exact Exponential Algorithms

Some problems can be hard to solve. We will look for good exact exponential algorithms to solve these problems. One problem is the TSP problem, which by bruteforce takes $\mathcal{O}(n!)$ time. Another is the MIS problem that takes $\mathcal{O}(2^n)$ time by bruteforce. We introduce the notion $O^*(g(n))$ for a function $f(n) = O(g(n)poly(n))$.

Dynamic approach to TSP works by calculating the lowest cost path in every subset. The amount of steps required to calculate the optimal route is $O(k^2)$, and we calculate all subsets (without c_1), so our running time is

$$\sum_{k=1}^{n-1} O(\binom{n}{k} k^2) = O^*(2^n)$$

The algorithm for MIS works by picking a vertex v with minimum degree and finds the optimal solution (recursively) by removing a vertex from the neighbourhood $N(v)$.
The worse case running time of this algorithm becomes the largest number of nodes, $T(n)$. We define $d(v)$ as the degree of the vertex we picked. That means we can express the number of nodes in the branching tree as

$$\begin{aligned} T(n) &\leq 1 + T(n - d(v) - 1) + \sum_{i=1}^{d(v)} T(n - d(v_i) - 1) \\ &\leq 1 + T(n - d(v) - 1) + \sum_{i=1}^{d(v)} T(n - d(v) - 1) \qquad \text{(Since v had min degree)} \\ &= 1 + (d(v) + 1) \cdot T(n - d(v) - 1) \end{aligned}$$

Setting $s = (d(v) + 1$, we get

$$\begin{aligned} T(n) &\leq 1 + s \cdot T(n - s) \\ &\leq 1 + s + s^2 + \cdots + s^{n/s} \\ &= \frac{1 - s^{n/s+1}}{1 - s} = \mathcal{O}^*(s^{n/s}) \end{aligned}$$

This has maximum for $s = 3$, which means

$$T(n) = O^*(3^{n/3})$$

We say a problem is fixed parameter tractable if you have an algorithm that runs in $O(f(k)poly(n))$ time for some parameter k. If you know that vertex cover needs some size k, we can reduce the problem so it has $k^2 + k$ vertices and k^2 edges. Do that by removing vertices with degree 0 and 1 (when 1, we keep the neighboor). Also include vertices that has a degree strictly larger than k.