NP-complete problems seem to only have exponential-time algorithms, but how exponential? e.g., we saw pseudo-polynomial time for knapsack—poly-time if profits have log(n) bits. More generally, consider other parameters. Small parameter $\Rightarrow$ poly-time.

Vertex Cover: Given graph $G = (V, E)$ and $k \in \mathbb{N}$, is there a vertex cover of size $\leq k$? A set of vertices hitting all edges.

What if we use $k$ itself as a parameter? $k = 1, 2, 3$. Try all possible $k$-sets $\binom{n}{k}$, which is $O(n^k)$. Checking a subset takes $O(n \cdot k)$ so total time is $O(k \cdot n^{k+1})$. This is poly-time for constant $k$.

Same idea works for clique or independent set, but not for graph coloring—given a graph, is it 3-colourable? is NP-complete.

$O(n^k)$ is still pretty bad.

We want $O(f(k) \cdot n^c)$ for some $f(k) \text{ ind. of } n$ and some constant $c \text{ ind. of } k$.

or even $O(f(k) + n^c)$

We can achieve this for Vertex Cover (but not for Ind. Set unless $P = NP$).

For each edge $(u, v)$ we need $u$ or $v$ in the Vertex Cover.

We can do exhaustive search, branching on this choice.

At each tree node, we pick an uncovered edge. $|C| \leq k \Rightarrow$ stop at depth $k$. 

Alg. VC2 \((G, k)\) — return True or False
if \(E = \emptyset\) return true
if \(k = 0\) return false
pick \(e = (u, v) \in E\)
return VC2(G \(- u, k-1\)) or VC(G \(- v, k-1\))

remove \(u\) and incident edges

Time \(O(2^k \cdot n)\)   \(\leftarrow\) work for each node
\# internal tree nodes

EX: find the vertex cover \(\leq 20\).

We can improve to \(O(f(k) + m+n)\)

technique called "kernelization".

Claim If \(G\) has a vertex \(v\) with \(\deg(v) > k\) then \(v\) must be in \(C\).
\(C\) vertex cover of size \(k\)

Pf if \(v \notin C\), we need all \(> k\) neighbours.

Alg. VC3 \((G, k)\) — return True/False
\(C'\) ← all vertices of \(\deg \geq k\)
\(k' \leftarrow k - |C'|\)
\(G' \leftarrow G \setminus (C'\text{ and all incident edges})\)

and remove isolated vertices

if \(G'\) has \(> 2k^2\) vertices
return False
else return VC2 \((G', k')\)

previous alg.

Note: \(G'\) is not very big:
max degree in \(G'\) is \(\leq k\)
if \(G'\) has VC of size \(\leq k\)
then \(G'\) has \(\leq k^2\) edges
\(\Rightarrow\) \(G'\) has \(\leq 2k^2\) vertices.
the actual vertex cover is \( C^1 \cup \text{vertex cover of } G^1, k^1 \)

idea due to Prof. J. Buss '93

Analysis: call to \( VC2 \) takes \( O(2^k \cdot \# \text{vertices of } G^1) \)

finding \( C^1 \) takes \( O(m+n) \)

total run time \( O(2^k \cdot 2^k \cdot m + n) \)

This 3rd alg. is even practical sometimes (much better than other two)

Comparison

\[
\begin{array}{|c|c|c|}
\hline
\text{Time} & n = 10^4 & k = 10 \\
\hline
\text{brute force} & kn^{k+1} & 10^{45} & \times \\
\hline
\text{VC2} & 2^k n & 10^9 & \{\text{possible}\} \\
\hline
\text{VC3} & 2^k k^2 + n + m & 2 \cdot 10^5 + m & \text{\{possible\}} \\
\hline
\end{array}
\]

DEF: A problem is **fixed parameter tractable** in parameter \( k \)
if it has an alg. with run time \( O(f(k) \cdot n^c) \)
where \( f(k) \) is a \( \mathcal{f} \alpha \) of \( k \) (ind. of \( n \))
and \( c \) is a constant (ind. of \( k \))

Examples of parameters

- value of opt. e.g. Ind. set of size \( k \)
- max degree of graph
- dimension for geometric problems
- genus of graph genus 0 = planar
  1 = on torus
Some examples of FPT algs:
- Vertex cover of size \( k \)
- Simple path of length \( k \)
- Finding \( k \) disjoint triangles in graph
- Drawing graph in plane with \( k \) edge crossings
- Finding disjoint paths connecting \( k \) pairs of nodes

Hardness - some parameterized problems have no FPT alg.,
unless \( P = \text{NP} \) e.g. Ind. Set.

The kernelization method generalizes

\[ \text{Thm. If a problem with parameter } k \text{ is FPT.} \]
\[ \text{i.e. there's an alg. with run time } O(f(k) \cdot n^c) \]
\[ \text{then there exists an alg. with run time } O(f'(k) + n^{c'}) \]

But proof is not constructive. (It's not hard either).
**k-path**: Given graph $G$, $k \in \mathbb{N}$, vertices $s$, $t$, find a simple $s$-$t$ path with exactly $k$ internal vertices. (don't repeat vertices (⇒ don't repeat edges)

A shortest $s$-$t$ path will be simple.

But if $k$ is larger, this problem is NP-hard — it generalizes Hamiltonian Path

**Ex.**

A randomized FPT algorithm

Colour vertices $V \setminus \{s,t\}$ randomly with $k$ colours

Test for colourful $s$-$t$ path — each colour appears exactly once.

We'll see how to test later.

**Error analysis:**

no simple $s$-$t$ path $\Rightarrow$ no colourful $s$-$t$ path $\Rightarrow$ alg. returns NO (for any colouring)

3 simple $s$-$t$ path $P$

$$\text{Prob} \ P \text{ is colourful} = \frac{2^{k!}}{k^k} \geq \frac{(\frac{k}{e})^k}{k^k} = \frac{1}{e^k}$$

Stirling $\lim_{k \to \infty} \frac{(\frac{k}{e})^k}{k^k} = \frac{1}{e}$

so Alg. outputs YES (i.e. is correct)

with prob. $\geq \frac{1}{e^k}$

If prob. success $\geq p$ then

prob. failure after $\frac{1}{p}$ repetitions is

$$1 - p + \frac{p^2}{2} \ldots \text{ Taylor expansion}$$

So in our case, repeating $e^k$ times gives prob. error $\leq \frac{1}{e}$
Finding a colourful $s$-$t$ path.
The colours on such a path are a permutation of $1 \ldots k$.
Try all $k!$ permutations.

The problem is easy for fixed permutation.

Run time $O(k! \cdot m)$, $m = \#\text{edges}$
Fact: can be improved to $O(2^k \cdot m)$
So we have a FPT alg. with small prob. of error, $\leq \frac{1}{e}$

and with run time $O(e^k 2^k \cdot m)$
Fact: can be derandomized (using $k$-perfect hash functions)