Today: Fixed-parameter algorithms
- vertex cover
- fixed-parameter tractability
- kernelization
- connection to approximation

Pick any 2:
1. hard problems
2. fast (poly.-time) algorithms
3. exact solutions

Idea: aim for exact algorithm, but isolate exponential term to a parameter
⇒ get fast solution for instances with small parameter value
   - hope parameter is small in practice

Parameter = nonnegative integer \( k(x) \)
- often a "natural" parameter (\( k \) in input)
- not necessarily efficiently computable (e.g. \( \Delta P \))
Parameterized problem = problem + parameter
  "problem w.r.t. parameter"
(potentially many interesting parameterizations)

Goal: polynomial in problem size n, exponential in parameter k

Example: k-Vertex Cover (NP-hard)
  Given: graph $G=(V,E)$, nonnegative integer $k$
  Q: is there a set $S$ of $\leq k$ vertices that "covers" all edges: $\forall e \in E \exists v \in S : e \ni v$
  Parameter: $k$

Note: can have $k \ll |V|$

Brute-force solution: (BAD)
  - try all $\binom{|V|}{k} + \binom{|V|}{k-1} + \cdots + \binom{|V|}{0}$ sets of $\leq k$ vs. vs.
  - can skip - bigger is better
  - test coverage in $O(m)$ time ($m=\#edges$)
  $\implies O(V^k E)$ time
  - polynomial for fixed $k$
  - but not same polynomial - e.g. not $O(V^{100})$
  - inefficient in most cases
  $\implies$ define $n^{f(k)}$ to be BAD

$\implies$ here $n = |V| + |E|$
**Bounded search-tree algorithm:** (Good)
- pick arbitrary edge $e = (u, v)$
- know that either $u \in S$ or $v \in S$ (or both) but don't know which
- guess: try both possibilities
  1. add $u$ to $S$
     delete $u$ & incident edges from $G$
     recurse with $k' = k - 1$
  2. ditto with $v$ instead of $u$
- return OR of two outcomes
- like guessing in dynamic programming, but memoization doesn't help here
- recursion tree:
  ![Recursion Tree](image)
  - at leaf ($k = 0$):
    return $|E| = 0$
  - $O(V)$ time to delete $u$ or $v$
  - $O(2^k \cdot V)$ time
    - $O(V)$ for fixed $k$
    - degree of polynomial independent of $k$
    - also polynomial for $k = O(\log V)$
    - practical for e.g. $k \leq 32$
  - define $f(k) \cdot n^{O(1)}$ to be *Good*
**FPT:** A parameterized problem is **fixed-parameter tractable (FPT)** if there is an algorithm with running time $\leq f(k) n^{o(1)}$.

Parameter: $f: \mathbb{N} \rightarrow \mathbb{N}$ (nonneg.)

**Question:** Why $f(k) \cdot n^{o(1)}$ not $f(k) + n^{o(1)}$?

**Theorem:** $\exists f(k) \cdot n^c$ algorithm $\iff \exists f'(k) \cdot n^{c'}$ algorithm

**Proof:**

$(\Leftarrow)$ trivial (assuming $f'(k) \& n^{c'} \geq 1$)

$(\Rightarrow)$

- if $n \leq f(k)$ then $f(k) \cdot n^c \leq f(k)^{c+1}$
- if $f(k) \leq n$ then $f(k) \cdot n^c \leq n^{c+1}$

So $f(k) \cdot n^c \leq \max \left\{ f(k)^{c+1}, n^{c+1} \right\}$

or $\frac{f'(k)}{f(k)} \leq \frac{n^{c+1}}{f(k)^{c+1}}$.

**Example:** $O(2^k \cdot n) \leq O(4^k + n^3)$
Kernelization: a simplifying self-reduction polynomial-time algorithm converting input \((x,k)\) into small equivalent input \((x',k')\) 
\[|x'| \leq f(k) \iff \text{answer}(x) = \text{answer}(x')\]

Theorem: \(\text{FPT} \iff \exists\text{ kernelization}\)

Proof: \((\Leftarrow)\) kernelize \(\Rightarrow n' \leq f(k)\) 
run any finite \(g(n')\) algorithm 
\(\Rightarrow n^{o(1)} + g(f(k))\) time

\((\Rightarrow)\) let \(A\) be an \(f(k) \cdot n^c\) algorithm 
\(\begin{cases} \text{if } n \leq f(k)\text{ then already kernelized} \\ \text{if } f(k) \leq n:\end{cases}\)

assuming \(k\) is known
- run \(A \Rightarrow f(k) \cdot n^c \leq n^{c+1}\) time
- output \(O(1)\)-size \(\text{YES/NO}\) instance as appropriate (to kernelize)

if \(k\) is unknown: run \(A\) for \(n^{c+1}\) time 
& if not done, know already kernelized

So (exponential) kernel exists. Recent work aims to find polynomial (even linear) kernels when possible.
Polynomial kernel for $k$-vertex cover:
- make graph simple:
  - remove loops & multi-edges
  - any vertex of degree $\geq k$ must be in cover (else need $\geq k$ vertices to cover inc. edges)
  - remove such vertices (\& incident edges) one at a time, decreasing $k$ accordingly
  - remaining graph has max. degree $\leq k$
  - each remaining cover vertex covers $\leq k$ edges
  - if $\#$ remaining edges $> k^2$, answer is No:
    output canonical No instance: $$
  - else $|E'| \leq k^2$
  - remove isolated vertices
  - $|V'| \leq 2k^2$
  - reduced to instance $(V', E')$ of size $O(k^2)$
  - running time: $O(VE)$ obvious, $O(V+E)$ with more work

- if we now apply:
  - brute-force solution $\Rightarrow O(V+E+(2k^2)^k k^2)$
    \[= O(V+E + 2^k k^{2k+a}) \text{ time} \]
  - bounded search-tree solution $\Rightarrow O(V+E + 2^k k^2) \text{ time}$

Best algorithm to date: $O(kV + 1.274^k)$
[Chen, Kanj, Xia - TCS 2010]
Connection to approximation algorithms:
- take optimization problem, integral OPT
- consider associated decision problem: $OPT \leq k$?
- parameterize by $k$

Theorem: optimization problem has \underline{EPTAS (efficient PTAS)} $f(\frac{1}{\varepsilon}) \cdot n^{O(1)}$
e.g. Approx-Partition \cite{L17}

$\Rightarrow$ decision problem is FPT

Proof: (like FPTAS $\Leftrightarrow$ pseudopoly. alg.)
- say maximization problem ($\leq k$ decision)
- run EPTAS with $\varepsilon = \frac{1}{2}k$ in $f(2k) \cdot n^{O(1)}$
- relative error $\leq \frac{1}{2}k < \frac{1}{k}$
  $\Rightarrow$ absolute error $< 1$ if $OPT \leq k$
- so if we find solution with value $\leq k$
  then $OPT \leq (1 + \frac{1}{2}k) \cdot k \leq k + \frac{1}{2}$
  integral $\Rightarrow$ OPT $\leq k \Rightarrow$ YES.
- else OPT $> k$

Also: $\leq$, $\geq$ decision problems are equivalent w.r.t. FPT

\text{~Can use this relation to prove \underline{EPTAs don't exist in some cases}}