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

(cf. friends, sleep, work)

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. \( \text{OPT} \))
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: $\forall e \in E \exists v \in S : e
  \text{Parameter: } k$

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 $\leq k$ vertices.
    can skip - bigger is better
  - test coverage in $O(m)$ time ($m = \#\text{edges}$)
  $\Rightarrow O(V^k E)$ time
    - polynomial for fixed $k$
    - but not same polynomial - e.g. not $O(V^{100})$
    - inefficient in most cases
  $\Rightarrow$ define $n^{f(k)}$ to be BAD
  $\Rightarrow$ 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:

- 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 $f(k) \cdot n^{O(k)}$ to be Good
**FPT:** parameterized problem is fixed-parameter tractable (FPT) if there is an algorithm with running time \( \leq f(k) n^{o(1)} \)

\[ f: \mathbb{N} \to \mathbb{N} \quad \text{parameter} \quad \text{indep. of } k \& n \quad \text{(nonneg.)} \]

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

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

**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 \( f(k) \cdot n^c \leq \max \{ f(k)^{c+1}, n^{c+1} \} \)

\[ \leq f(k)^{c+1} + n^{c+1} \]

\[ \frac{f'(k)}{c} \]

or:

\( xy \leq x^2 + y^2 \Rightarrow f'(k) = f(k)^2 \& c' = 2c \)

**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
\(\Rightarrow n^\omega(1) + g(f(k))\) time

\((\Rightarrow)\) let \(A\) be an \(f(k)\cdot n^c\) algorithm
if \(n \leq f(k)\) then already kernelized
if \(f(k) \leq n:\)
\{ assuming \(k\) is known \}
- run \(A\) \(\Rightarrow f(k)\cdot n^c \leq n^{c+1}\) time \(\checkmark\)
- output \(O(1)\)-size YES/NO instance as appropriate (to kernelize)

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

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
  \( \Rightarrow \) remaining graph has max. degree \( \leq k \)
  \( \Rightarrow \) each remaining cover vertex covers \( \leq k \) edges
  \( \Rightarrow \) if \# remaining edges \( > k^2 \), answer is No:
    output canonical No instance: \(-, \emptyset\)
  - else \( |E'| \leq k^2 \)
- remove isolated vertices
  \( \Rightarrow |V'| \leq 2k^2 \)
  \( \Rightarrow \) 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 \( \Rightarrow O(V+E+(2k^2)^k k^2) \)
    \( = O(V+E+2^k k^{2k+2}) \) time
  - bounded search-tree solution
    \( \Rightarrow O(V+E+2^k k^2) \) 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 ≤ k?
- parameterize by k

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

⇒ decision problem is FPT

Proof: (like FPTAS ↔ pseudopoly. alg.)
- say maximization problem (& ≤ k decision)
- run EPTAS with \( \varepsilon = \frac{1}{2k} \) in \( f(2k) \cdot n^{O(1)} \)
- relative error \( \leq \frac{1}{2k} < \frac{1}{k} \)

⇒ absolute error < 1 if OPT ≤ k

⇒ so if we find solution with value ≤ k
then \( \text{OPT} \leq (1 + \frac{1}{2k}) \cdot k \leq k + \frac{1}{2} \)
integral ⇒ \( \text{OPT} \leq k \) ⇒ YES.
- else \( \text{OPT} > k \)

Also: \( \leq \geq \) decision problems are equivalent w.r.t. FPT

~ Can use this relation to prove EPTASs don't exist in some cases