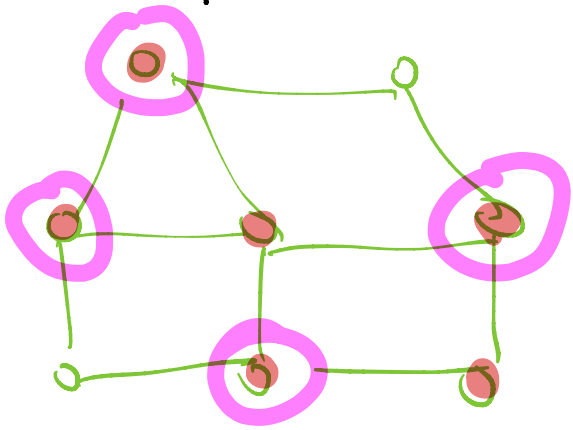
How to attack HARD problems
→ NP-complete problem, NP-hard

Running example: VC = vertex cover problem $\in$ NP-complete

Graph $G = (V, E)$ undirected

vertex cover $S \subseteq V$ if $\forall uv \in E: u \in S$ or $v \in S$



minimization: Find k_{min} : $|S| = k_{min}$, $S = VC$
G $\forall S': |S'| < k_{min}$ $S' \neq VC$

decision: VC_k : $\exists S \subseteq V$ s.t. $|S| \leq k$, $S = VC$? → Yes / No
G, k

poly min $\Rightarrow_p$ poly decision

poly decision $\Rightarrow_p$ poly min : use the former as an oracle for the latter + binary search on k

FOCUS ON DECISION VC_k

BASELINE

$\binom{n}{k}$ subsets of nodes $S \subseteq V$, where $|S| = k$ and $n = |V|$ $(m = |E|)$

↳ check each subset in poly(n) time

⇒ total time is $O(n^{k+O(1)})$ $\binom{n}{k} \sim n^k$

no need to generate $|S| < k$

PARAMETERIZED ALGORITHMS (EXACT EXPONENTIAL ALGORITHMS)

Solve the problem exactly in time

$O(f(k)\ \text{poly}(n))$ where $f(k)$ can be large

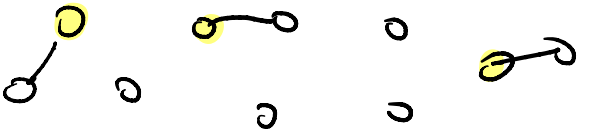
Useful when $k \ll n$, even though k can be very close to n in some instance

Note that this doesn't say anything on the P vs NP question

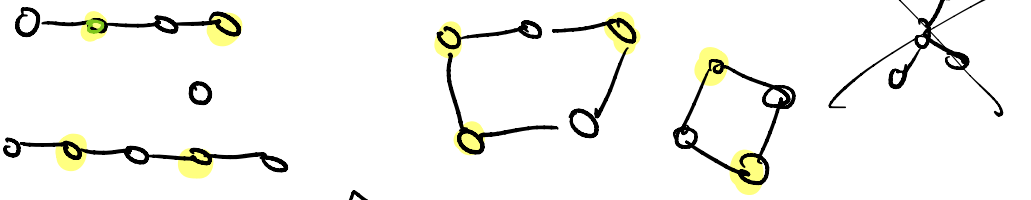
KERNELIZATION

Let's consider easy cases:

CASE 1 max degree $(G) = 1$



CASE 2 max degree $(G) = 2$



isolated nodes : NOTHING

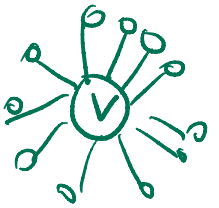
Simple paths
Simple cycles
} check if their number of edges is odd or even

VC_h as instance $\langle G, k \rangle$ to indicate that we want to check whether a vertex cover of size $\leq k$ is in G

We apply R1 and R2 below as long as possible:

R1 | If v is an isolated node ($dep(v)=0$) then solve $\langle G-\{v\}, k \rangle$

R2 | If v has degree $> k$ in G then solve $\langle G-\{v\}, k-1 \rangle$
v must belong to the VC



either choose v or all of $N(v)$

each rule take poly(n) time

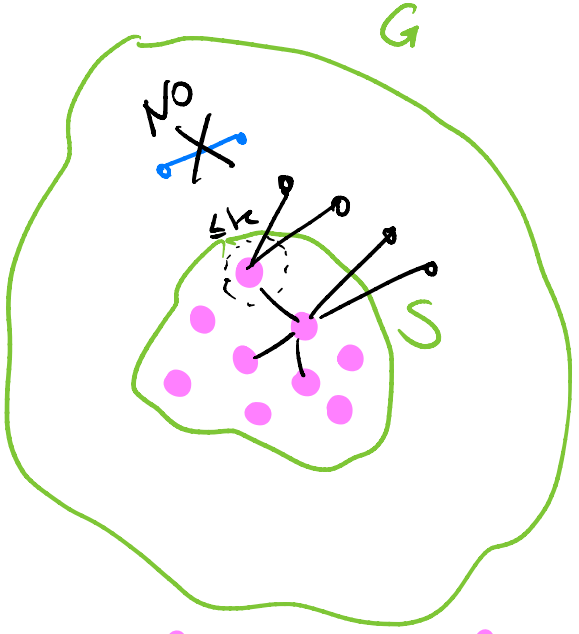
When it is not possible to apply R1 or R2 anymore, we apply

R3 | if $k<0$ or G has $> k^2+k$ nodes or $G > k^2$ edges
then report no VC for the input
else let G_k be the current graph (reduced with R1 and R2)
Solve VC on $\langle G_k, k \rangle$ and return the answer

why!

▶ If $\langle G_n, k \rangle$ has a VC of size $\leq k$ then it must have $\leq k^2 + k$ nodes and $\leq k^2$ edges

Let S be such VC, $|S| \leq k$



1) $|S| \leq k$, $\forall u \in S : d(u) \leq k$, S is VC
$\Rightarrow |E| \leq |S| \cdot k = k^2$

2) $|S| + |G_k \setminus S| \leq k + k^2$
(1)

1)+2) $\Rightarrow |G_k| = poly(k)$

≤k pink

To answer $\langle G_n, k \rangle$ we run BASELINE
Total cost $O(f(n) + poly(n))$ $\Rightarrow f(n) = (k^2)^k$

BRANCHING TECHNIQUE. $\langle G, k \rangle$ as before

v = max degree node in G : recursion on this degree

BASE CASE: $\deg(v) = 1 \Rightarrow$ CASE 1

RECURSIVE CASE: $\deg(v) \geq 2$

(i) either v belongs to VC (and we cannot claim anything for $N(v)$)

(ii) or all of $N(v)$ should belong to the VC

Total cost # recursive calls × poly(n)

each call generates two calls ⇒ # calls ≤ 2 # leaves

BASE CASE

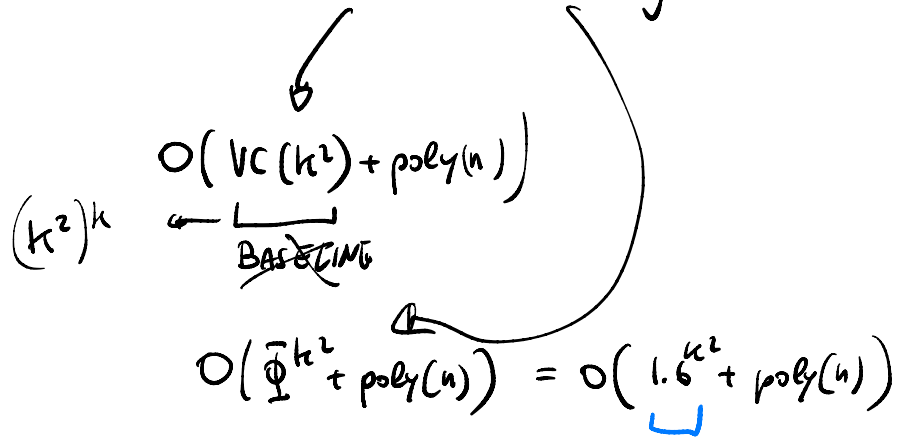
$L(k) = L(k-1) + L(k-\underbrace{N(u)}_{\geq 2}) \leq L(k-1) + L(k-2)$ — $L(k)$

(i) (ii)

$L(k)$ is Fibonacci! $L(k) \sim \Phi^k$ $\Phi = 1.6\ldots$

Total cost is $O(\Phi^k \cdot \text{poly}(n))$

Putting together the two kernelization + branching



P.S.
You can improve branching:
max degree = 2 is easy case 3!

$\Rightarrow F(k) \le F(k-1) + F(k-3)$

$\Leftrightarrow 1.6^{k^2}$ vs $(1.46)^{k^2}$