

Course «Algorithms» · Lecture 13

P, NP, and Polynomial-Time Reductions

Textbook: A. Levitin, Section 11.3

Today’s question

Are there problems with **no fast algorithm at all?**

- what “efficient” means
- decision problems, the classes P and NP
- reductions: comparing how hard problems are
- next week: NP-completeness

Polynomial vs. exponential

n	n^2	n^3	2^n
10	100	1,000	1,024
50	2,500	125,000	$\approx 1.1 \cdot 10^{15}$
100	10,000	1,000,000	$\approx 1.3 \cdot 10^{30}$

At 10^9 operations per second: $\frac{1.3 \cdot 10^{30}}{10^9} \approx 10^{21}\ \text{s} \approx 4 \cdot 10^{13}\ \text{years}$

A computer 1000 times faster?

Exponential 2^n

$$2^{n'} = 1000 \cdot 2^n \quad \Rightarrow \quad n' \approx n + 10$$

only 10 elements more

Cubic n^3

$$(n')^3 = 1000 \cdot n^3 \quad \Rightarrow \quad n' = 10\,n$$

ten times larger inputs

“A bit better” vs. “a different league”.

The convention: efficient = polynomial

Worst-case time $O(n^c)$ for some constant c. Then the problem is **tractable**.

- Not perfect: n^{40} is useless; some exponential algorithms work in practice
- But polynomials are closed under +, ×, composition
- Independent of the machine model and reasonable encodings
- In practice, polynomial algorithms usually have low degree

Decision problems: yes or no

Optimization	Decision
Find a shortest tour through all n cities	Is there a tour of length ≤ k?
Find the largest clique	Is there a clique of size ≥ k?

We add a bound k. The decision version is **no harder**: just compare the optimum with k.

… and no easier: Clique from a yes/no oracle

1. Optimal size k*: binary search on k in 1…n

$$\lceil \log_2 n \rceil \quad \text{queries}$$

2. The clique itself: delete vertices one by one; if a k*-clique still exists, keep it deleted, else put it back

O(n) calls recover the whole solution. So we study decision problems only.

The class P

P = decision problems solvable in time polynomial in the input size.

- Is the graph connected? (DFS/BFS)
- A spanning tree of weight ≤ k? (Prim, Kruskal)
- Shortest distance s → t at most k? (Dijkstra)
- Is N prime? In P only since 2002 (AKS)

Careful with the input size

Knapsack DP (Lecture 7) runs in $\Theta(nW)$. W takes only about $\log_2 W$ bits:

$$W = 2^{\log_2 W}$$

Polynomial in the **value**, exponential in the **length**: pseudo-polynomial, so knapsack is **not** shown to be in P.

The class NP: easy to check

Hamiltonian cycle: finding one seems to need up to (n – 1)! orders. Checking a proposed one is quick.

NP = decision problems whose proposed solution (a **certificate**) can be verified in polynomial time.

$$x \text{ is a yes instance} \iff \exists\, c,\ |c| \le \text{poly}(|x|) : \quad V(x, c) = \text{accept}$$

Certificates are needed only for “yes”, and they must be short.

Certificates and verifiers

Problem	Certificate	Check
Hamiltonian cycle	order of the vertices	a cycle through all vertices
Clique of size k	k vertices	all pairs adjacent
Subset sum	a subset	add up, compare with target
SAT	truth assignment	evaluate the formula
TSP (decision)	a tour	add weights, compare with k
3-coloring	a coloring	every edge has two colors

NP is not “not polynomial”

NP = **N**ondeterministic **P**olynomial time: a machine that “guesses” the certificate and then checks it.

$$\mathrm{P} \subseteq \mathrm{NP} \qquad V(x, c) = \mathrm{Solve}(x)$$

If we can solve a problem fast, the verifier just ignores the certificate.

$$\mathrm{P} = \mathrm{NP} \; ?$$

Is every problem that is easy to **check** also easy to **solve**?

A Clay Millennium Prize problem. Almost everyone believes P ≠ NP. Nobody has proved it.

Reductions: comparing problems

$A \leq_p B$: a polynomial-time f translates every question about A into one about B with the same answer.

$$x \in A \iff f(x) \in B$$

- B ∈ P ⇒ A ∈ P: compute f(x), run B's algorithm
- A hard ⇒ B at least as hard
- "$A \leq_p B$" reads: A is no harder than B

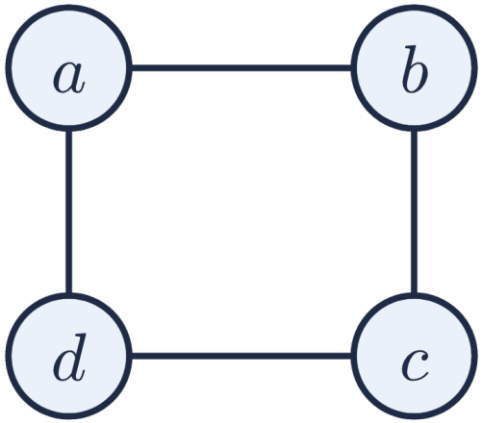
Three graph problems

Independent set: no two vertices of S are adjacent

Vertex cover: every edge has an end in S

Clique: every two vertices of S are adjacent

In the 4-cycle: {a, c} is independent, {b, d} is a vertex cover. Complements!



Independent set ⇔ vertex cover

$$S \text{ independent} \iff V - S \text{ is a vertex cover}$$

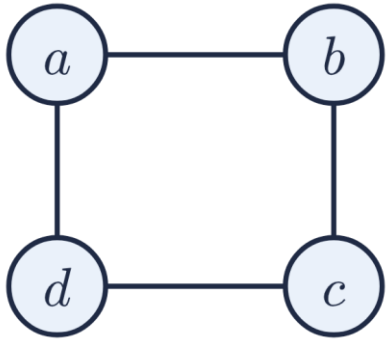
$$\text{IS} \geq k \iff \text{VC} \leq |V| - k$$

No edge has both ends in S ⇔ every edge has an end in V – S.

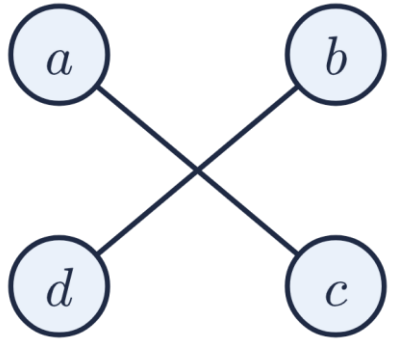
Same graph, k → |V| – k: the two problems are equally hard.

Clique ⇔ independent set in the complement

$$S \text{ a clique in } G \iff S \text{ independent in } \overline{G}$$



G



complement: {a, c} is a clique

Building the complement takes $O(|V|^2)$. All three problems are equally hard.

Common mistakes

- **NP = “not polynomial”:** no, and P ⊆ NP
- **Value vs. length** of a number in the input
- **Certificates for “no”:** NP needs them only for “yes”
- **Direction of $\leq_p$:** A $\leq_p$ B means A is no harder than B

Summary

- Efficient = polynomial; exponential stays hopeless on any hardware
- Decision and optimization versions are equally hard
- P: solvable fast. NP: yes-answers checkable fast. $P \subseteq NP$
- $P = NP$ is open; almost everyone believes $P \neq NP$
- $A \leq_p B$ moves solvability down and hardness up
- Independent set, vertex cover and clique are equally hard

Review questions

1. Define the classes P and NP. Why is $P \subseteq NP$?
2. Give a certificate and a polynomial verifier for (a) Hamiltonian cycle, (b) subset sum, (c) 3-coloring.
3. Show that S is independent iff $V - S$ is a vertex cover. What follows about the two problems?