

Course «Algorithms» · Lecture 15

The Future of Algorithms

Textbook: A. Levitin, Chapters 11–12

Today’s goal

How the ideas of this course extend to current research.

- classic problems that are still improving
- **parallel algorithms:** work and span
- massive data, approximation, machine learning
- quantum algorithms and open questions

The course in retrospect

Weeks	Topic	Main idea	How we prove it
3–5	Analysis	Count the basic operation	Sums, recurrences
6–7	Dynamic programming	Overlapping subproblems	Optimal substructure
8–9	DP on graphs	Add intermediate vertices	Induction on k
10–12	Greedy	A locally best choice	Cut property, exchange
13–14	Complexity	Compare problems	Polynomial reductions

Classic problems are not closed

Shortest paths

Duan et al. (2025): deterministic bound below, for directed graphs with nonnegative real weights.

Matrix multiplication

From n^3 to Strassen's $n^{2.81}$; the best known exponent is below 2.372.

$$O(m + n \log n) \quad \longrightarrow \quad O(m \log^{2/3} n)$$

Dijkstra's algorithm isn't optimal on sparse graphs. A lower bound must be proved.

Parallel algorithms: work and span

- **Work** T_1: the total number of operations
- **Span** T_∞: the longest chain that must run in sequence
- **Parallelism** T_1 / T_∞: the maximum average speedup

$$T_p \geq \frac{T_1}{p}, \qquad T_p \geq T_\infty, \qquad T_p \leq \frac{T_1}{p} + T_\infty$$

The last bound is Brent's theorem: a greedy scheduler nearly reaches the lower bounds.

Example: merge sort and Floyd

$$T_\infty(n) = T_\infty(n/2) + \Theta(n) = \Theta(n), \qquad \frac{T_1}{T_\infty} = \Theta(\log n)$$

- The sequential merge dominates the span
- Parallel merge: span **$\Theta(\log^3 n)$**, parallelism **$\Theta(n / \log^2 n)$**
- Floyd, ignoring loop management: $T_1 = \Theta(n^3)$, $T\infty = \Theta(n)$, parallelism about n^2

DP often parallelizes well because a table's dependencies are explicit.

Amdahl’s law

$$S(c) = \frac{1}{(1 - p_0) + p_0/c} \leq \frac{1}{1 - p_0}$$

For $p_0 = 0.9$: $S(8) \approx 4.7$, and no number of cores gives more than 10.

The sequential part sets the limit, so algorithms must reduce it.

Massive data: the Bloom filter

Add a key

Set its k bits to 1 in an array of m bits.

Query a key

A bit is 0: absent for certain. All bits are 1: probably present.

- No false negatives, but false positives are possible
- Space doesn’t depend on the size of the keys

The false positive rate

$$\left(1 - \frac{1}{m}\right)^{kn} \approx e^{-kn/m}, \qquad \varepsilon \approx \left(1 - e^{-kn/m}\right)^{k}$$

$$k = \frac{m}{n} \ln 2, \qquad \varepsilon \approx 0.6185^{m/n}$$

10 bits per key: k ≈ 7 and ε ≈ 0.8%. This is the idea of sketching.

Approximation with a guarantee

```
ALGORITHM ApproxVertexCover(G)
  C ← ∅
  while G has an edge (u, v) do
     add u and v to C
     remove all edges incident to u or v from G
  return C
```

$$|C| = 2|M| \leq 2 \cdot \mathrm{OPT}$$

The chosen edges form a matching M, and any cover needs one endpoint of each: OPT ≥ |M|.

Problems differ in how well they approximate

Problem	What is known
Vertex cover	Ratio 2, by a simple algorithm
Metric TSP	Ratio 1.5 (Christofides)
General TSP	No constant ratio unless P = NP
Knapsack	Any $1 + \varepsilon$ in polynomial time

The tight example: on the path a–b–c–d–e, the algorithm may return 4 vertices, and {b, d} is optimal.

Learning: indexes and predictions

- **Learned index:** a model predicts the position; with error ε, search takes **O(log ε)**
- Interpolation search: **O(log log n)** on average, **Θ(n)** in the worst case

Consistency

Near-optimal when the prediction is correct.

Robustness

A classical guarantee when the prediction is arbitrarily wrong.

Learning supplies the hint; worst-case analysis supplies the guarantee.

Learning: discovering algorithms

AlphaTensor (2022)

4 × 4 matrices over Z_2 with 47 multiplications; two levels of Strassen use 49.

AlphaDev (2023)

Sorting up to 70% faster for 3–5 elements and about 1.7% faster for over 250,000; added to LLVM libc++.

Candidates must still be verified. Machine search makes proof more important, not less.

Quantum algorithms

Shor (1994)

Factoring and discrete logarithms in polynomial time. No polynomial classical algorithm is known.

Grover (1996)

Unstructured search in O(√N) queries, which is optimal (Bennett et al., 1997).

$$2^n \text{ candidates} \longrightarrow \text{about } 2^{n/2} \text{ queries}$$

Not believed to solve NP-complete problems in polynomial time (unproved).

August 2024: NIST published post-quantum standards FIPS 203, 204 and 205.

What we still don’t know

P versus NP is still open. Fine-grained complexity looks inside P.

SETH: for every ε > 0, there is a k such that k-SAT has no algorithm in

$$O\left(2^{(1-\varepsilon)n}\right)$$

Backurs, Indyk (2015): edit distance in the following time would refute SETH:

$$O\left(n^{2-\delta}\right)$$

So the classic DP for edit distance is probably close to optimal.

Responsibility

Fairness

Learning from past decisions can reproduce their bias.

Explainability

Can a decision be justified to the person it affects?

Privacy

Differential privacy gives a mathematical guarantee.

Energy

A better algorithm saves real resources at scale.

State exactly what is guaranteed, and under which assumptions.

Where to go from here

Direction	Key topics	Builds on
Advanced design	Flows, linear programming, randomization	Lectures 6–12
Complexity theory	Hardness of approximation, fine-grained	Lectures 13–14
Parallel, distributed	Work-span, MapReduce, GPUs	Lectures 4–5, 8–9
Massive data	Sketching, streaming, external memory	Lectures 3–4
Machine learning	Optimization, predictions	Lectures 4, 6
Quantum computing	Quantum and post-quantum algorithms	Lectures 13–14

For a thesis: state the model, state the guarantee, prove it, test it.

Pitfalls

- **Treating a textbook bound as optimal:** a lower bound must be proved
- **Assuming speedup equals the number of cores**
- **Reading a Bloom filter’s “present” as certain**
- **Reporting average accuracy instead of a guarantee**
- **Thinking quantum computers try all answers at once**

Summary

- Even classic problems keep improving
- Parallel algorithms: work and span
- A Bloom filter: 10 bits per key give about 1% error
- Approximation proves ratios without knowing the optimum
- Learned hints need consistency and robustness
- P versus NP remains open

Review questions

1. Define work, span and parallelism. Why is the parallelism of merge sort with a sequential merge only $\Theta(\log n)$?
2. Derive the false positive rate of a Bloom filter and state your assumptions. Why are there no false negatives?
3. Prove that ApproxVertexCover has a ratio of 2. Which step gives a lower bound on the optimum?
4. Why does Grover’s algorithm not imply a polynomial quantum algorithm for SAT? What does it give instead?

Reading: Levitin, Section 4.5, Chapters 11–12; Cormen et al., Chapters 26, 34, 35.