Course «Algorithms and Data Structures»

6B06102 – Information Systems
6B06109 – Artificial Intelligence
6B06103 – Computer Engineering
6B06104 – Computer Science

Lecture 15

The Future of Algorithms

Textbook: A. Levitin, Chapters 11–12

Do new technologies make algorithms obsolete?

Many processors

Parallel sorting is divide and conquer on many cores

Huge data

A Bloom filter is a hash table that saves memory

Learning

A learned index is a smarter binary search

The answer is the opposite: each new idea grows out of what you already know.

The course on one page

Weeks	Topic	Main idea	Key costs
4	Arrays and search	Sorted: throw half away	O(n), O(log n)
5	Recursion	A smaller copy of a problem	Call stack
6–7	Lists, stacks, queues	Restricted access	O(1) at the ends
9	Trees, BST	Go left or right	O(h)
10–11	Sorting	Divide and conquer	$O(n^2)$, O(n log n)
12	Hash tables	Compute the place of a key	O(1) on average
13–14	Graphs, BFS, DFS	Queue wide, stack deep	O(V + E)

The way you organize the data decides which algorithms are possible.

Many processors: parallel algorithms

- Processors no longer get much faster by themselves
- Computers now have many **cores**; a graphics card has thousands of small ones
- An algorithm uses them only if it’s designed for that

Merge sort: the two halves don’t depend on each other, so two cores can sort them at the same time.

The limit: Amdahl's law

$$S = \frac{1}{(1-p) + \frac{p}{c}}$$

Cores c, with p = 0.9	2	4	8	16	as many as you like
Speedup S	1.8	3.1	4.7	6.4	10

The sequential 10% remains, so the speedup can't exceed 10.

Too much data: the Bloom filter

Is a key on the slow disk? Keeping all the keys in memory is too expensive.

To add a key

Compute its hash values and set those bits to 1.

To check a key

A bit is 0: definitely not there. All bits are 1: probably there.

An array of m bits and a few hash functions. The keys themselves aren’t stored.

Bloom filter: the algorithm

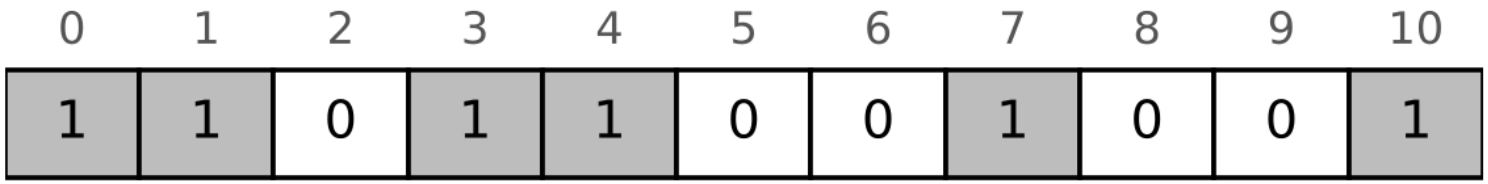
```
ALGORITHM BloomAdd(x)
    for each hash function h do
        B[h(x)] ← 1

ALGORITHM BloomCheck(x)
    for each hash function h do
        if B[h(x)] = 0
            return "definitely not present"
    return "probably present"
```

O(k) steps for k hash functions, however many keys there are.

Bloom filter: adding keys

$$m = 11, \qquad h_1(x) = x \bmod 11, \qquad h_2(x) = (3x + 1) \bmod 11$$

Key	h_1	h_2	Bits set
12	1	4	1, 4
25	3	10	3, 10
40	7	0	7, 0



Bloom filter: checking keys

Key	h_1, h_2	Bits	Answer	Truth
25	3, 10	1, 1	probably present	correct
50	6, 8	0, 0	definitely not present	correct, no disk read
33	0, 1	1, 1	probably present	false positive

It can wrongly say “yes” but never wrongly say “no”.

A key that was added has always set its bits. A false positive costs only one useless disk read.

Learning makes algorithms faster

For “Zhang”, you open the phone book near the end. **Interpolation search:**

$$m = l + \frac{K - A[l]}{A[r] - A[l]} (r - l)$$

$$K = 70 : \quad m = \frac{67}{95} \cdot 12 \approx 8, \quad A[8] = 74 > 70$$

$$m = \frac{67}{67} \cdot 7 = 7, \quad A[7] = 70$$

Two steps, where binary search took three. The worst case can be O(n).

Learned indexes

- Replace the formula with a small trained model
- The model predicts the position; a short search corrects the error
- On suitable data it can replace a search tree and use less memory (Kraska et al., 2018)
- **Algorithms with predictions** use a learned hint, but keep a guarantee when it’s bad

Learning helps to discover algorithms

AlphaDev (Google DeepMind, 2023) searched for short sorting programs at the level of machine instructions.

Up to 70% faster

for sequences of three to five elements

About 1.7% faster

for sequences of more than 250,000 elements

Added to LLVM libc++. To judge such results, people still count operations, test and prove.

Quantum algorithms

A different model of computation, not simply a faster computer. It helps only with some problems.

Grover (1996)

Searches an unsorted collection of N items in about √N steps.

Shor (1994)

Splits a large number into prime factors in polynomial time.

A good data structure is hard to beat

N	Sequential search	Grover, about √N	Binary search, sorted data
10^6	10^6	10^3	20
10^{12}	10^{12}	10^6	40

If the data can be sorted, plain binary search beats Grover easily.

Security after quantum computers

- Much of internet security relies on factoring being hard
- Today’s quantum computers are too small and error-prone to threaten it
- People are preparing in advance

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

What we still don’t know: P versus NP

Example: find the shortest route that visits every city once. Checking a route is easy. Finding the best one seems very hard.

If a solution is easy to check, must it also be easy to find?

- Most researchers believe the answer is no, but nobody has proved it
- One of the Clay Millennium Problems: one million dollars
- In practice: **approximation algorithms** and **heuristics**

Responsibility

Fairness

A model can learn old prejudices from old decisions.

Explainability

If a loan is refused, can anyone say why?

Privacy

What personal data is used, and who can see it?

Energy

An efficient algorithm also saves electricity.

Where to go from here

If you liked…	Study next	You'll meet
Big-O and clever ideas	Design and analysis of algorithms	Dynamic programming, NP-completeness
Trees and hash tables	Databases	Indexes, B-trees
Graphs	Networks, optimization	Shortest paths, flows
Learning from data	Machine learning	Decision trees, neural networks
Many cores, huge data	Parallel and distributed systems	MapReduce, GPU programming
Hashing, number theory	Cryptography	Encryption, digital signatures

Common mistakes

- Thinking that a faster computer can save a bad algorithm
- Thinking that “probably present” means the key is there
- Thinking that a quantum computer solves everything instantly
- Thinking that an algorithm is neutral because it’s mathematics

Summary

- Each new direction grows from the basics.
- More cores help only as far as Amdahl’s law allows.
- Big data accepts small errors for huge savings.
- Machine learning speeds up and helps to discover algorithms.
- Quantum algorithms help on some problems; P versus NP is open.
- Use algorithms responsibly. Think first, code last.

Review questions

1. If 80% of a program can run in parallel, what is the speedup on 4 cores? What is the largest possible speedup?
2. Using the Bloom filter from this lecture, check the key 18. What is the answer, and is it correct? Why can a Bloom filter never answer “definitely not present” for a key that was added?
3. Why doesn’t Grover’s algorithm make binary search useless? What does the P versus NP question ask?

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