(Algorithms and) Data Structures

Gabriel Istrate

March 10, 2025

Organizational

- Course objectives: present main data structures, improve alg. design.
- Course webpage: *Google classroom*, mvi7e25.
- Class handouts: *will be posted on webpage.*
- Seminar: me + TBD.
- Lab: TBD.
- Each seminarist: **responsible for their own rules concerning seminar/lab grade.**

Organizational (II)

- Grading: exam, lab.
- Exact percentages: 66%-33 %. Must pass both.
- Course attendance: strongly recommended, not strictly enforced.

However

being a student is a full time job. Would your boss accept “couldn’t do X because I had classes ?”.

- Expect you: Work hard.
- Exam: **no point in memorizing courses, there will be no ”theory” part as such.** Rather, I want to see that you understood material.

Important

- Academic honesty
- OK/encouraged: speak up in class. Two-way, rather than one-way communication. Request: *be concise, to the point, respect time spent together in class.*
- Disclaimer: I can make mistakes/be wrong. Let me know (in person, email) how I can improve things.
- **Also:** Please be in good standing with the university. I am **not** competent to answer questions of this nature (I will give you a grade if you appear in the catalog; it's your responsibility to be there)

Organizational (III)

- less programming, more math.
- Reason: need to know pointers, you're learning them this semester.
- However: *mostly C, some basic features of C++, STL.*
- Please: brush up on/read **basic features** of C/C++, I will review them in the Lab sections.
- video lectures from previous years (linked on classroom).

Textbook: combination of texts.

- **Cormen, Leiserson, Rivest, Stein**. First edition translated in Romanian as well. Theoretical, but very good book.
- (secondary) Adam Drozdek *"Data structures and algorithms in C++", third edition or newer.* Good for programming.

Where are we ?

Several methods for designing algorithms

- Divide and conquer.
- Greedy.
- Dynamic programming.
- (when everything fails) Backtracking.

Data structures

Paradigm for developing efficient algorithms based on abstraction: Abstract operations needed by the program, change implementation so that frequent operations run fast.

Example: Selection sort vs. HEAPSORT

Selection SORT

- Elements to sort in a vector.
- Find maximum element.
- Swap it with the last element.
- Proceed recursively.

Complexity

- Finding maximum in a vector: $\Theta(n)$.
- Complexity analysis: $T(n) = T(n-1) + \theta(n)$.
- Conclusion: $\Theta(n^2)$

Example: Selection sort vs. HEAPSORT

Selection SORT

- Bottleneck in Selection Sort: Find maximum element
- If we could improve finding max

HEAPSORT

- Algorithm: Same idea.
- Bottleneck: FindMax $\theta(n)$. Replace it with $O(1)$ operation (via heaps).
- Complexity $W(n) = \theta(\log n)$ (need also to update heap via HEAPIFY)
- New algorithm: complexity $\theta(n \log(n))$.