Algoritmi Avansați
Seminar 1

Gabriel Majeri

1 Introducere

Ce este o problemă (computațională)?

O **problemă computațională** [1] este o problemă la care putem determina răspunsul folosind un algoritm[1].

- **Exemplu de problemă computațională:** Care sunt factorii primi ai numărului natural n?
- **Exemplu de problemă care nu se poate rezolva folosind algoritmi:** Ce pereche de pantaloni să port astăzi?[2]

Ce tipuri de probleme computaționale există?

- **Probleme de decizie** [4]**:** Răspunsul este de tipul adevărat/fals (posibil/imposibil, există/nu există etc.)
- **Probleme de optimizare** [5]**:** Răspunsul este o *soluție* (un număr, un șir de numere, un text etc.) care este "cea mai bună" dintr-un anumit punct de vedere (consumă cele mai puține resurse, produce cel mai mare profit, acoperă cele mai multe noduri etc.) față de toate celelalte soluții posibile.

[1] Un *algoritm* [2] este o serie de pași bine-definiți care îți permit să obții un rezultat plecând de la niște date de intrare. În comparație, "să ghicești răspunsul corect" nu e tocmai un algoritm. Dar poate fi o euristică [3].

[2] Dacă ai un număr finit de opțiuni (pantaloni din care să alegi) și asociezi fiecărei perechi câte un *scor* (număr) care să indice dezirabilitatea acesteia, atunci această problemă se poate reduce la o problemă computațională, aceea de a găsi perechea de pantaloni cu scorul maxim.

2 Exerciții

Cunoștințe generale

1. Dați exemplu de **2 probleme** pe care le întâlniți în viața de zi cu zi care se pot interpreta ca probleme computaționale (e.g. cum alegi traseul pe care să vii la facultate).

2. Dați exemplu de **2 probleme de decizie** și **2 probleme de optimizare** pe care le-ați întâlnit în practică sau de care ați auzit.

P versus NP

3. În cele ce urmează, să presupunem că avem **o listă** (un vector) **de** n **numere întregi**.

1. Propuneți un algoritm care să determine **cel mai mare element** (în **valoare absolută**) din listă. Ce complexitate de timp/memorie are algoritmul propus?

2. Pentru un număr întreg dat S, propuneți un algoritm care să determine **toate perechile de numere** din lista inițială **care adunate dau** S. Puteți găsi un algoritm cu complexitate de timp, în cel mai rău caz, mai bună decât $\mathcal{O}(n^2)$?

3. (3SUM [6]) Propuneți un algoritm care să determine dacă în lista inițială există **trei numere** care **adunate să aibă suma 0**. Credeți că puteți un algoritm determinist care să rezolve problema în mai puțin de n^2 pași? Dar dacă ați aborda problema în mod nedeterminist?

Metoda greedy

Mai multe informații despre principiile care stau la baza metodei greedy se pot găsi la [7].

4. Suntem administratorii unui spital și vrem să ne asigurăm că avem tot timpul cel puțin un medic prezent la camera de gardă într-un anumit interval de timp. Presupunem că avem la dispoziție n medici, care ar fi dispuși să stea de gardă fiecare între anumite ore. Vrem să asignăm câți mai puțini medici, ca ceilalți să se poată ocupa de alte urgențe.

Problema formalizată: Find dat un interval $[a, b]$ și o mulțime de intervale $[a_1, b_1], [a_2, b_2], \ldots, [a_n, b_n]$, vrem să alegem un număr *minim* dintre acestea astfel încât, reunite, să includă intervalul inițial $[a, b]$.

1. Aceasta este o problemă de **decizie** sau de **optimizare**? Dar dacă ne-ar interesa doar să vedem dacă putem acoperi intervalul dat cu intervalele disponibile?

2. Propuneți un algoritm care **să determine dacă** măcar putem acoperi intervalul dat cu intervalele disponibile.

3. Propuneți un algoritm care să rezolve problema inițială. **Demonstrați** că algoritmul propus obține soluția optimă (adică că nu poate exista o altă soluție, diferită de cea obținută prin metoda voastră, care să folosească mai puține intervale).

Referințe

[1] Wikipedia contributors, *Computational problem*, URL: https://en.wikipedia.org/wiki/Computational_problem.

[2] Wikipedia contributors, *Algorithm*, URL: https://en.wikipedia.org/wiki/Algorithm.

[3] Wikipedia contributors, *Heuristic*, URL: https://en.wikipedia.org/wiki/Heuristic_(computer_science).

[4] Wikipedia contributors, *Decision problem*, URL: https://en.wikipedia.org/wiki/Decision_problem.

[5] Wikipedia contributors, *Optimization problem*, URL: https://en.wikipedia.org/wiki/Optimization_problem.

[6] Wikipedia contributors, *3SUM*, URL: https://en.wikipedia.org/wiki/3SUM.

[7] Karleigh Moore, Jimin Khim și Eli Ross, *Greedy Algorithm*, URL: https://brilliant.org/wiki/greedy-algorithm/.