SEMINAR 14

METODA GREEDY

PRINCIPII

- Găsește soluția incremental: construiește soluții prin extinderea lor treptată
- La fiecare pas soluția este extinsă cu **cel mai bun candidat dintre candidații rămași la un moment dat** (**„lacom”**) i.e. se adaugă succesiv la rezultat elementul care realizează optimul local
- O decizie luată pe parcurs nu se mai modifică ulterior

ELEMENTE

- **Mulțimea candidat** - mulțimea de unde se aleg elementele soluției
- **Funcția de selecție -** funcția care alege cel mai bun element pentru a fi adăugat la soluție
- **Funcția *acceptabil*** - funcția folosită pentru a determina dacă un candidat curent poate contribui la soluția pentru problemă
- **Funcția obiectiv -** funcție care calculează o valoare pentru soluție/soluții parțiale
- **Funcția soluție -** funcție care verifică dacă s-a ajuns la o soluție pentru problemă

1. Se dă o sumă M și tipuri de bancnote. Să se găsească o modalitate de a plăti suma M de bani folosind cât mai puține bancnote.

2. Se dorește programarea unor task-uri pentru a fi efectuate. Toate task-urile au aceeași importanță, și sunt caracterizate de momentul de start (s_i) și momentul în care se termină (s_f). Să se programeze cât mai multe task-uri pentru o persoana, fără ca acestea să se suprapună.

Ex. 1:

$T_i = (s_i, s_f)$

T1=(10, 20)
T2 = (12, 25)
T3 = (20, 30)

Numar maxim de activitati: 2 (T1, T2)

Ex. 2:

T1=(1, 2)
T2 = (3, 4)
T3 = (0, 6)
T4 = (5, 7)
T5 = (8,9)
T6 = (5, 9)

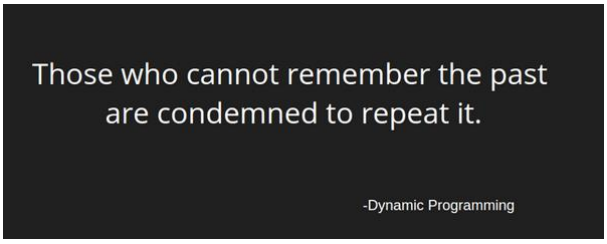
Număr maxim de task-uri: 4 (T1, T2, T4, T5)

PROGRAMARE DINAMICĂ

- Optimality principle (forward, backward, mixed form)

  A problem is said to satisfy the Principle of Optimality if the subsolutions of an optimal solution of the problem are themesleves optimal solutions for their subproblems.

- Memoization/Tabulation



- Overlapping subproblems

In other words, pentru a aplica programare dinamică:

- Ne asigurăm că principiul optimalității este demonstrat pentru problemă
- Definim structura soluției optime; definim o recurență care indică modul în care putem obține optimul general din optimele parțiale
- Calculăm soluția optimă începând cu subproblema cea mai simplă, pentru care cunoaștem soluția, și apoi construim treptat

1. [LONGEST INCREASING SUBSEQUENCE] Se dă o listă cu numere întregi. Să se determine și să se afișeze subsecvența de lungime maximă cu elementele în ordine crescătoare.

   ✓ Princpiul de optimalitate verificat
   ✓ Structura soluției optime: construim secvențe L = <L1, L2,…, Ln> astfel încât pentru i, $1 \leq i \leq n$, L_i = lungimea celei mai lungi subsecvente care se termină la poziția i
   ✓ Definiția recursivă pentru valoarea soluției optime:

$$L_i = max\{1 + L_j \mid A_j \; unde \; A_j \leq A_i, j = i-1, i-2, \ldots, 1\}$$

2. Se dă o listă cu numere întregi. Să se determine și să se afișeze suma maximă care se poate calcula prin însumarea elementelor unei subliste.

3. [LONGEST COMMON SUBSEQUENCE] Se dau 2 string-uri. Să se determine lungimea subsecvenței comune maxime între cele 2 string-uri.

4. Coin change problem
5. Problema rucsacului

(acestea sunt genul de probleme care pot aparea la examen, i.e. de acest nivel de dificultate)

Other resources & further reading:

GREEDY

Standard Greedy Problems

PROGRAMARE DINAMICĂ

To recap and delve further: Abdul Bari - YouTube (capitol 3 - Greedy, 4 - Programare dinamică - explicatii mai pe larg pentru longest_common_subsequence)

To clarify if needed:

- Tabulation vs. Memoization (1) or Tabulation vs. Memoization (2)

Examples on how to apply dynamic programming:

Solved Dynamic Programming Problems (also, math) (probleme de dificultate putin mai ridicata, implicand si alte concepte - **nu** intra la examen, dar e exemplificat modul de abordare (demonstrare principiul de optimalitate - definitie structura - definitie recurenta)

si

Solved Dynamic Programming Problems (a little less math) (Basic Problems Section - probleme clasice de Dynamic Programming; pentru fiecare se parcurg mai multe solutii - recursiv simplu, recursive+memoization, dynamic programming si se vede foarte bine imbunatatirea ca performanta/eficienta pe care o presupune programarea dinamica).