***GREEDY***

***CARACTERISTICI GREEDY:***

1. determină soluția unei probleme de optimizare pe baza alegerii unui optim local
2. timpul de rezolvare este de obicei polinomial
3. folosirea criteriului de alegere se face pe setul de date de intrare, eventual cu o pregătire prealabilă a acestora
4. soluțiile se construiesc selectând cel mai valoros candidat, neluând în calcul ce s-a întâmplat la iterațiile precedente, plecându-se inițial de la mulțimea vidă
5. se caută optimul global, bazându-se la fiecare etapă pe alegerea unui optim local (așteptarea este ca succesiunea de optime locale să ducă la optimul global)
6. la fiecare pas se folosește un criteriu de selecție pentru a alege cel mai valoros candidat
7. se urmărește obținerea unei valori optime a unei funcții obiectiv definite la început

***CONDIȚII:***

- o mulțime de candidați;

- o funcție care verifică dacă o anumită mulțime de candidați constituie o soluție posibilă, nu neapărat optimă, a problemei;

- o funcție care verifică dacă o mulțime de candidați este fezabilă, adică dacă este posibil să completăm această mulțime astfel încât sa obținem o soluție posibilă, nu neapărat optimă, a problemei;

- o funcție de selecție care indică la orice moment care este cel mai promițător dintre candidații încă nefolosiți;

- o funcție obiectiv care dă valoarea unei soluții, funcția pe care urmărim să o optimizăm (minimizam/maximizam).

***OBSERVAȚII:***

* posibilul inconvenient al acestei tehnici constă în faptul că alegerea pe baza unui criteriu de optim local NU conduce întotdeauna spre un optim global;
* metoda aleasă trebuie însoțită de demonstrația faptului că soluția obținută pe baza alegerii optimelor locale este în final un optim global;
* algoritmii Greedy conduc în multe cazuri la soluții optime ( și simplu de obținut), dar nu întotdeauna;
* metoda Greedy este destul de puternică și se aplică în probleme din multe domenii (optimizare discretă, grafuri, probleme de transport, etc);
* Soluția este construită pas cu pas ca la Backtracking, fără mecanismul de revenire la pasul anterior (ambele tehnici oferă soluții sub formă de vector, Greedy oferă (eventual ) doar soluția optimă);
* Tehnica Greedy duce (de cele mai multe ori) la rezolvare în timp polinomial;
* Pentru problemele pentru care nu se cunosc algoritmi care necesită timp polinomial, se caută soluții, chiar dacă nu optime, dar apropiate de acestea și care au fost obținute în timp util. Multe din aceste soluții sunt obținute cu Greedy. Astfel de algoritmi se numesc algoritmi euristici.

***VERIFICAREA CORECTITUDINII:***

Proprietatea substructurii optime: orice soluție optimă a problemei inițiale conține o soluție optimă a unei subprobleme (problema de același tip, dar de dimensiune mai redusă).

Proprietatea alegerii tehnicii Greedy: componentele unei soluții optime au fost alese folosind criteriul Greedy de selecție sau pot fi înlocuite cu elemente alese folosind acest criteriu fără a altera proprietatea de optimalitate.

**Ex:**

Presupunem ca setul de scrisori este ordonat crescător după timpul de ardere necesar pentru fiecare scrisoare.

Proprietatea de alegere "greedy":

Fie O = (o1, o2, . . . , om) o soluție optimă și X = (, , . . . ,) soluția dată de algoritm. Există mai multe situații:

* Dacă k > m atunci O nu este soluția optimă. 2
* Dacă k = m atunci X este optimă.
* Dacă k < m, atunci putem înlocui în O pe cu (scrisoarea care arde cel mai repede) fără a altera restricția problemei și păstrând același număr (maxim) de spectacole selectate. Obținem soluția optimă O = (, , . . . , ).

Proprietatea de substructura optimă. Considerăm soluția optimă O' = (x1, o2… om) determinată anterior. Presupunem că (o2, o3... om) **nu** este soluție optimă a subproblemei selecției din {scrisoare\_2,scrisoare\_3,...,scrisoare\_n}. Rezultă că există O'' = (o''2... o''k'') o altă soluție cu k'' > m. Acest lucru ar duce la o soluție ( x1, o''2... o''k'') mai bună decât O' = (x1, o2… om). Contradicție.

**ALGORITM:**

* Se împarte problema în subprobleme independente.
* Pentru fiecare subproblemă se alege un optim local.
* Algoritmul face întotdeauna alegerea care arată cel mai bine în acel moment.
* În final, mulțimea de optime locale ar trebui să ducă la un optim global.

Nu se ajunge întotdeauna la optimul global!

***AVANTAJE:***

* Algoritmi simpli și intuitivi.
* Oferă cea mai bună soluție la nivel local.
* Complexitatea în timp este, în general, polinomială.
* Se poate folosi în multe domenii: determinarea celor mai scurte drumuri în grafuri (Dijkstra), determinarea arborelui minimal de acoperire (Prim, Kruskal), codificare arborilor Huffmann, planificarea activităților, problema spectacolelor și problema fracționară a rucsacului.

***DEZAVANTAJE:***

* Succesiunea de optime locale nu duce întotdeauna la un optim global.
* Nu avem întotdeauna soluție.
* Demonstrația corectitudinii soluției găsite poate fi greoaie.