Algoritmi nedeterministi

Algoritmii se pot clasifica in:

Deterministi:

- fiecare acțiune/operație are un rezultat unic determinat
- **serialitate**: pentru orice moment de timp t din cursul execuției exista o singură acțiune efectuată la momentul t

Nedeterministi:

- există acțiuni/operații al căror rezultat nu este unic definit ci are valori într-o mulțime **finită** de posibilități
- **paralelism**: structura arborescentă de operații; operațiile de pe o cale din arbore sunt efectuate serial; operațiile de pe căi diferite sunt efectuate în paralel; la un moment de timp t se execută mai multe acțiuni pe diverse cai
- nu are implementare practica

Operațiile caracteristice algoritmilor nedeterministi sunt:

- <u>choice(A)</u> – ramifică copia curentă a algoritmului în cardinal(A) copii
  - A = mulțime **finita!**
  - pentru fiecare valoare din A - copie a algoritmului care continuă cu acea valoare
  - variabilele locale ale algoritmului sunt clonate pentru fiecare copie
  - copiile continuă în paralel și independent una de alta
- <u>fail</u> – copia curentă se termină cu insucces; restul copiilor continua execuția
- <u>success</u> – copia curentă se termina cu succes; celelalte copii sunt terminate (practic execuția întregului algoritm se termina cu succes)

Obs1: Valoarea de return a algoritmului:

- success (când una din cai se termină cu success, nu conteaza restul)
- fail (când toate căile se termina cu fail)

Obs2: Algoritmii de optim pot fi modelați ca apeluri succesive ale unor **algoritmi de decizie** (de exemplu determinarea arborelui de acoperire minim pe un arbore fără costuri pe muchii: exista arbore de acoperire din 1 muchie? dacă nu, exista din 2 muchii? etc)

Complexitatea temporala a unui algoritm nedeterminist (se mai numeste si <u>complexitate angelica</u>):

- suma complexităților operațiilor din secventa/calea cea mai **scurta** care termina algoritmul cu **success**
- dacă toate căile întorc fail, complexitatea este suma complexităților de pe calea cea mai lungă încheiată cu fail
- **Obs:** choice(A) are complexitate O(1).

Etape in execuția unui algoritm nedeterminist:

- **generare** (choice – generează cate o copie pentru fiecare candidat la a fi soluție) – aceasta este partea nedeterminista a algoritmului
- **testare** (fiecare candidat generat este testat dacă e o soluție corecta)

Exemple de algoritmi nedeterminiști:

- **Cautarea unui element intr-un vector**

```
// V = vectorul, n = nr de elemente din vector, e = elementul cautat
caut(V, n, e) {
      i = choice(1..n)  //generare - pozitia elementului cautat
      if (V[i]=e)
            success  //testare - este elementul cautat?
      fail
}
```

- **Test daca un numar natural este neprim**

```
// n = numarul testat
neprim(n) {
      i = choice(2..floor(sqrt(n))) // generare – posibilii divizori
      if (n mod i=0) // testare – este i divizor pentru n?
            success
      fail
}
```

- **Sortarea unui vector de elemente strict pozitive**

```
// V = vectorul, n = nr de elemente din vector
sort(V, n) {
      for i=1 to n
            Aux[i]=0 // in Aux vom crea vectorul sortat
      for i=1 to n {
            j = choice(1..n) // generare - poziția ocupata de V[i]
            if (Aux[j]>0) // testare - nu e deja ocupata?
                  fail
            Aux[j] = V[i]
      }
      for i=1 to (n-1)
            if (Aux[i]>Aux[i+1]) // testare - e sortat?
                  fail
      success // daca niciun test nu a dat fail, success
}
```

P = PTIME = clasa problemelor rezolvabile prin algoritmi determiniști polinomiali
NP = NPTIME = clasa problemelor rezolvabile prin algoritmi nedeterminiști polinomiali

Obs: NP nu înseamnă ca nu este P! NP vine de la “non deterministic polynomial time”, nu de la “not P”. In fapt, $P \subseteq NP$ (orice algoritm determinist polinomial poate fi ușor transformat intr-un algoritm nedeterminist polinomial), iar dacă $P = NP$ rămâne în continuare o problemă deschisa (cel mai plauzibil este că nu sunt egale, însă nu s-a putut demonstra încă).

Problema:

- **tractabila:** rezolvabilă printr-un algoritm determinist polinomial
- **intractabilă:** toți algoritmii determiniști care o rezolva sunt supra-polinomiali

Găsiți un algoritm nedeterminist pentru următoarele probleme si calculati complexitatea in fiecare caz:

1. Fiind dat un vector de numere, exista o subsecventa de elemente egale consecutive de lungime > k?
2. Avand un graf, sa se determine daca exista un drum de la nodul u la nodul v care are lungimea < decat o valoare data dim.
3. Colorarea unui graf:
   Dându-se un graf G(V, E) și k culori, se pot colora nodurile grafului doar cu cele k culori astfel încât niciun nod sa nu aibă un vecin de aceeași culoare?
4. k-clica:
   Dându-se un graf G(V, E) și un număr k, exista un subgraf complet (o clica) de dimensiune k?
5. k-acoperire (vertex cover):
   Dându-se un graf G(V, E) și un număr k, exista o submulțime de k noduri astfel încât fiecare muchie (v1,v2) sa aibă cel puțin unul dintre nodurile care o compun (v1 sau v2) în submulțimea aleasa?
6. Submulțime de sumă dată (Q-sume):
   Se da o mulțime de N numere și un numar Q. Exista o submulțime de numere a căror suma să fie fix Q?
7. Problema comis-voiajorului (TSP):
   Se dă o mulțime de orașe conectate intre ele prin drumuri. Exista vreo modalitate ca un comis voiajor sa viziteze toate orașele o singura data și să se întoarcă de unde a plecat?
8. Plasati 8 regine pe o tabla de sah fara ca acestea sa se atace.
9. Problema subgrafurilor izomorfe:
   Doua grafuri G1(V1, E1) si G2(V2, E2) sunt izomorfe daca exista o functie bijectiva f:V1->V2 astfel incat: muchia (u, v) este in E1< => muchia (f(u), f(v)) este in E2 (obs: doua grafuri pot fi izomorfe daca au acelasi numar de noduri).
   Dându-se două grafuri, G1 și G2, există un subgraf în G1 care să fie izomorf cu G2?
10. Independent set:
    Dându-se un graf G(V, E) și un număr k din Z, există o mulțime S de k noduri astfel încât orice muchie are cel mult un capat in S?
11. Problema partiționării:
    Dându-se o mulțime de t numere intregi, există o impartire a elementelor sale in două submulțimi S1 și S2 care sa aiba sume egale?
12. SAT(Boolean Satisfiability Problem)
    Se dă o expresie booleană în forma normala conjunctivă - o conjuncție de clauze, unde clauzele sunt disjuncții. Exemplu $(x1 \lor \neg x2) \land (\neg x1 \lor x2 \lor x3) \land \neg x1$. Să se determine dacă există o posibilitate de atribuire a variabilelor astfel încât expresia să fie adevărată.