Algoritmi nedeterminiști

Madalina Hurmuz

December 3, 2018

1 Clasificare

Algoritmii se pot clasifica în:

- **Deterministi**:
  - fiecare acțiune/operație are un rezultat unic determinat
  - **serialitate**: pentru orice moment de timp t din cursul execuției există o singură acțiune efectuată la momentul t
- **Nedeterminiști**:
  - există acțiuni/operații al căror rezultat nu este unic definit ci are valori într-o mulțime finită de posibilități
  - **paralelism**: structură 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 căi
  - nu are implementare practică

2 Operații

- choice(A)
  - ramifică copia curentă a algoritmului în cardinal(A) copii
  - A = mulțime **finită**!
  - pentru fiecare valoare din A se creează o 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
- fail - copia curentă se termină cu insucces; restul copiilor continuă execuția

- success - copia curentă se termină cu succes; celelalte copii sunt terminate (practic execuția întregului algoritm se termină cu succes)

Observații

- **Valorea de return a algoritmului**:
  - success (când una din căi se termină cu success, nu contează restul)
  - fail (când toate căile se termină cu fail)
- **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: există arbore de acoperire din 1 muchie? dacă nu, există din 2 muchii? etc)

3 Complexitate

Complexitatea temporală a unui algoritm nedeterminist (se mai numește și **complexitate angelică**):

- suma complexităților operațiilor din secvența/calea cea mai scurtă care termină 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).

4 Etape

Etape în execuția unui algoritm nedeterminist:

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

5 Exemple de algoritmi nedeterminiști

5.1 Cautarea unui element într-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
}
```

5.2 Test dacă un număr 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
}
```

5.3 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) // generarea pozitiei 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
}
```

Obs:

P = PTIME = clasa problemelor rezolvabile prin algoritmi deterministi polinomiali
NP = NPTIME = clasa problemelor rezolvabile prin algoritmi nedeterministi 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 într-un algoritm nedeterminist polinomial), iar dacă P = NP rămâne în continuare o problemă deschisă (cel mai plauzibil este că nu sunt egale, însă nu s-a putut demonstra încă).

Problemă:

- **tractabilă:** rezolvabilă printr-un algoritm determinist polinomial
- **intractabilă:** toți algoritmii deterministi care o rezolvă sunt supra-polinomiali

6 Exerciții

Găsiți un algoritm nedeterminist pentru următoarele probleme și calculați complexitatea în fiecare caz:

1. Fiind dat un vector de numere, există o subsecvență de elemente egale consecutive de lungime > k?
2. Având un graf, să se determine dacă există un drum de la nodul u la nodul v care are lungimea < decât o valoare dată 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 să nu aibă un vecin de aceeași culoare?
4. k-clică:
   Dându-se un graf G(V, E) și un număr k, există un subgraf complet (o clică) de dimensiune k?
5. k-acoperire (vertex cover): Dându-se un graf G(V, E) și un număr k, există o submulțime de k noduri astfel încât fiecare muchie (v1,v2) să aibă cel puțin unul dintre nodurile care o compun (v1 sau v2) în submulțimea aleasă?

6. Submulțime de sumă dată (Q-sume):
   Se dă o mulțime de N numere și un număr Q. Există o submulțime de numere a căror sumă să fie fix Q?

7. Problema comis-voiajorului (TSP):
   Se dă o mulțime de orașe conectate între ele prin drumuri. Există vreo modalitate ca un comis voiajor să viziteze toate orașele o singură dată și să se întoarcă de unde a plecat?

8. Plasati 8 regine pe o tablă de șah fară ca acestea să se atace.

9. Problema subgrafurilor izomorfe:
   Două grafuri G1(V1, E1) și G2(V2, E2) sunt izomorfe dacă există o funcție bijectivă $f : V1- > V2$ astfel încât: muchia (u, v) este în E1 <=> muchia (f(u), f(v)) este în E2 (obs: două grafuri pot fi izomorfe dacă au același număr 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 capăt în S?

11. Problema partiționării:
    Dându-se o mulțime de t numere întregi, există o împarțire a elementelor sale în două submulțimi S1 și S2 care să aibă sume egale?

12. SAT(Boolean Satisfiability Problem):
    Se dă o expresie booleană în forma normală conjunctivă - o conjuncție de clauze, unde clauzele sunt disjuncții. Exemplu $(x1 \vee \neg x2) \wedge (\neg x1 \vee x2 \vee x3) \wedge \neg x1$. Să se determine dacă există o posibilitate de atribuire a variabilelor astfel încât expresia să fie adevărată.