# Algoritmi

Algoritmii pot fi clasificați drept algoritmi determiniști și nedeterminiști. Această clasificare a dat naștere și claselor de complexitate P și NP.

Un algoritm este alcătuit dintr-o intrare (input) care este procesată și transformată într-o ieșire (output).

## **Algoritmi determiniști**

Un algoritm determinist este un algoritm pentru care rezultatul este același pentru orice intrare, având același comportament în orice execuție.

Un algoritm determinist trebuie să rezolve întotdeauna corect problema și cât de repede este posibil.

### **Arbori de decizie**

Arborii de decizie sunt folosiți pentru a selecta cea mai bună direcție de acțiune în situațiile în care apare incertitudinea.

Studiul acestora oferă o perspectivă asupra noțiunilor de calcul. Astfel de modele limitate apar adesea într-o varietate de aplicații, chiar și în afara informaticii.

Deseori putem folosi arbori pentru a reprezenta procesele decizionale care au loc in cadrul unui algoritm. Mulți agloritmi de sortare și căutare pot fi analizați cu arbori de decizie deoarece efectuează comparații.

Chiar dacă cel care ia decizia nu cunoaște ce efect va avea factorul necunoscut, el are de obicei niște cunoștințe despre efectele ce pot să apară și cum e mai probabil să apară fiecare efect. Această informație poate fi utilizată pentru a selecta opțiunea care este cel mai probabil să producă rezultate favorabile. Arborii de decizie fac ușor de aplicat acest tip de analiză.

Un astfel de arbore într-un algoritm este unul ale cărui noduri reprezintă puncte de decizie și ale căror frunze reprezintă posibile rezultate.

Un arbore decizional este o structură sub forma unui arbore care conține două tipuri de noduri: noduri terminale sau frunze, și noduri decizionale:

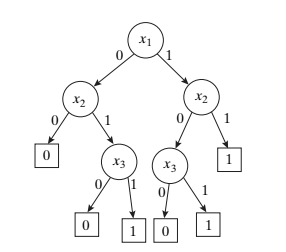
* Nod rădăcină – nod care nu are arce ce intră în el și zero sau mai multe arce care ies
* Nod intern – are exact un singur nod care intră în el și două sau mai multe arce care ies
* Frunză sau nod terminal – exact un singur nod intră, niciun arc care iese.

Fiecare nod decizional reprezintă un test pentru o anumită proprietate.

Fiecare arc, care pleacă dintr-un astfel de nod, fiind o valoare a proprietății respective.

Fiecare frunză reprezintă o clasă.

Fie o funcție. Un arbore de decizie pentru f este un arbore pentru care fiecare nod intern este notat cu și are două arce de ieșire notate cu 0 sau 1. Fiecare frunză este notată cu o valoare de ieșire 0 sau 1. Calcularea intrării se face la fiecare nod prin inpectarea intrării indicată de notația nodului. Dacă calculul continuă în subarborele respectiv. În acest fel, intrarea x urmează o cale prin arbore. În figura 1 se descrie un exemplu de arbore decizional.



Figură 1 - Un arbore decizional pentru o funcție Output-ul este 1 daca cel puțin două date de intrare este 1, altfel acesta este 0.

Complexitatea arborelui decizional al unei funcții este numărul de date examinate de cel mai eficient arbore în cazul cel mai rău.

Costul unui arbore cu input-ul și costul un este definit de

unde reprezntă setul de arbori de decizie care alcătuiesc funcția .

Din moment ce orice funcție booleană cu valori în poate fi calculată de un arbore binar de adâncime (și noduri), atunci pentru orice .

#### **Căutare binară**

Presupunem că se caută printr-o listă sortată într-un mod binar. Verificăm nodul din mijlocul listei pentru a vedea dacă este valoarea pe care o căutăm. Dacă nu este, atunci efectuăm aceeași operație pe jumătatea stângă sau pe jumătatea dreaptă a arborelui, în funcție de valorea pe care o căutăm. Acest algoritm se poate reprezenta printr-un arbore de decizie.

Proprietăţile arborilor binari de căutare:

* fiecare subarbore este un arbore binar de căutare
* valorile din subarborele stâng al unui nod sunt mai mici decât valoarea din nod
* valorile din subarborele drept al unui nod sunt mai mari decât valoarea din nod
* valoarea minimă se află în cel mai din stânga nod
* valoarea maximă se află în cel mai din dreapta nod
* înălţimea arborelui binar de căutare este cuprinsă între și .

##### Exemplu

Fie o listă sortată care conține primele 15 numere prime:

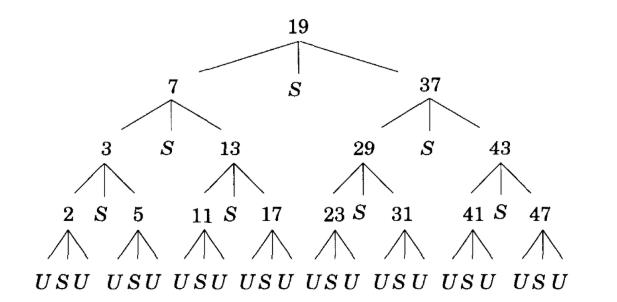
Primim o cheie și trebuie să aflăm dacă există în listă.

Arborele decizional pentru o căutare binară a listei de numere prime are numărul 19 ca rădăcină.

Se face comparația lui cu .

* Dacă atunci am ajuns la succes printr-o singură comparație.
* Dacă , atunci ne ducem la frunza/copilul stâng al lui ;
* Dacă ne ducem la frunza din dreapta.

Rezultatul devine un arbore ternar de decizie în care frunzele sunt notate ori cu , pentru o căutare cu succes, ori cu , pentru eșec.



Figură 2

În figura 2 se poate vedea că sunt maxim patru comparații pentru a afla dacă este sau nu în listă. Deci în cel mai rău caz este plus adâncimea arborelui binar. Știm că adâncimea minimă a arborelui binar cu noduri este . Deci limita inferioară a celui mai rău caz într-un arbore binar de căutare pe o listă sortată de n elemente este:

#### **Cântărire**

Presupunem că avem opt monede și vrem să aflăm care este moneda cea mai grea. Toate arată la fel și șapte dintre ele sunt la fel de grele. Trebuie să folosim o balanță.

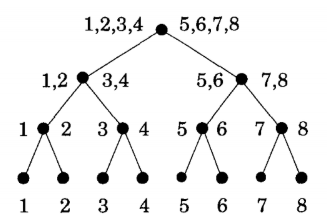
Sunt două modalități în care putem începe: dacă considerăm posibilitatea ca balanța să se afle în echilibru sau nu. Dacă nu se află niciodată în echilibru avem un arbore binar de decizie. Alfel, avem un arbore ternar.

Soluția 1 – arbore binar

Fiecare nod intern din arbore reprezintă balanța, cu un număr egal de monede pe fiecare parte. Avem următoarele posibilități:

* Dacă partea din stânga cântărește mai mult, atunci moneda mai grea se află acolo.
* Dacă partea din dreapta cântărește mai mult moneda căutată se află în partea dreaptă a balanței.

Fiecare frunză reprezintă câte o monedă. Presupunem că le numerotăm de la 1 la 8. În figura 3 este reprezentat arborele. Acesta găsește moneda cea mai grea din trei cântăriri.

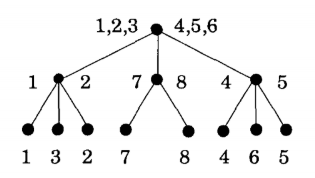


Figură 3

##### Soluția 2 – arbore ternar

În acest caz luăm în calcul posibilitatea ca balanța să se afle în echilibru. Deci nu e nevoie să folosim toate cele opt monede de la prima cântărire.

Figura 4 exemplifică o soluție de cântărire. Se observă că nu există un arc în mijlocul subarborelui din mijloc. Asta se întâmplă deoarece în punctul respectiv, una din monedele 7 sau 8 trebuie să fie mai grea. Acest algoritm găsește moneda dorită din două cântăriri.



Figură 4

##### Observație:

Oricare dintre cele opt monede poate să fie cea care cântărește mai mult. Deci trebuie să fie cel puțin opt frunze în orice algoritm decizional. Dar un arbore binar de adâncime poate să aibă frunze posibile, deci pentru a avea opt frunze, trebuie să avem . Asta înseamă că .

Dar pentru un arbore ternar de adâncime k poate aveam posibile frunze. Deci pentru a avea 8 frunze, trebuie să avem sau . Deci este o limită inferioară pentru numărul de cântăriri.

Din moment ce a doua soluție rezolvă problema din 2 cântăriri, aceasta este cea mai optimă.

## **Algoritmi nedeterminiști**

Un algoritm nedeterminist este un algoritm care poate da rezultate diferite pentru același input, cu, eventual, comportamente diferite la execuții diferite.

Un proces nedeterminist este influențat de alegerile care pot fi făcute.

Exemple de algoritmi nedeterminiști:

* algoritmii concurenți
* algoritmii aleatori

### **Algoritmi aleatori**

Algoritmii aleatori sunt cei mai cunoscuți algoritmi de tip nedeterminist, datorită vitezei și simplității.

Un algoritm aleator primește la intrare și o sursă de numere aleatoare care îi permit să facă alegeri aleatoare în timpul execuției.

Un algoritm aleator este un algoritm care în cursul execuției face anumite alegeri probabilistice. Aceste alegeri constau în generarea de valori ale unei variabile aleatoare, valorile fiind implicate în calcule pe care algoritmul le face.

Exemple:

* Protocolul Ethernet folosește alegeri aleatoare când acceseaza mediul de comunicație;
* testarea primalității în criptografie folosește alegeri probabilistice;
* unele probleme NP-dificile pot fi rezolvate pentru majoritatea intrărilor de către algoritmi aleatori.

Studiul variabilelor aleatoare asociate unui algoritm aleator este utilizat pentru a analiza eficiența și probabilitatea de a greși a algoritmului.

Aceștia se clasifică în:

* Agloritmi care rezolvă corect și întotdeauna problema asociată
* Algoritmi care au probabilitatea să greșescă în rezolvarea problemei

De exemplu:

* algoritmii Monte Carlo pot să greșească
* algoritmii Las Vegas nu greșesc niciodată și garantează că fiecare output este corect.

**Evaluarea unui game-tree**

Acest tip de evaluare are un rol important în inteligența artificală în problemele game-playing.

Un game-tree este un arbore cu rădăcină în care nodurile interne aflate la distanță pară față de rădăcină sunt etichetate cu (rădăcină are o etichetă ), iar restul nodurilor interne sunt etichetate cu .

Fiecărei frunze îi este asociată o valoare.

Fiecare frunză returnează valoarea asociată ei, fiecare nod returnează cea mai mare valoare a unuia dintre fii, fiecare nod returnează cea mai mică valoare a unuia dintre fii.

Exemplu

Presupunem că în frunze valorile sunt biți. Un nod este o operație logică , iar un nod are o operație logică .

Arborele binar complet are adâncimea și frunze.

* Într-un nod primul copil evaluat poate returna valoarea 1
* Într-un nod primul copil evaluat poate returna valoarea 0.

În amandouă cazurile agloritmul este forțat să evalueze ambii descendenți.

Algoritmul aleator evaluează recursiv un descendent ales la întâmplare al nodului curent. Dacă valoarea rezultată nu determină valorea nodului, se evaluează și celălalt descendent.

Pentru un game-tree cu adâncimea , costul mediu al evaluării este cel mult .

Considerăm mai întâi un nod care returnează 1 cu doua frunze drept copii .

Cu probabilitate algoritmul alege mai întâi frunza cu valoarea 0, numărul mediu de pași este .

Pentru un nod care returnează 0 cu două frunze drept copii, numărul mediu de pași este cel mult .

Câștigul este că, spre exemplu, într-un nod intern care returnează 1 amândoi copii trebuie sa returneze 1 ceea ce este cazul "bun" pentru .

Se evaluează mai întâi numarul de pași în cazul .

Fie un arbore cu rădăcina care are doi copii , fiecare având câte două frunze drept copii.

* Dacă rădăcina returneaza 0, atunci numărul de evaluări este (1 pentru unul dintre copii și 2 pentru frunzele acestuia).
* Dacă rădăcina returnează 1, numărul de evaluari este (amandoi copiii sunt în cazul "bun" pentru noduri ).

Considerăm acum un nod ai cărui copii sunt rădăcinile câte unui arbore cu adâncimea .

* Dacă rădăcina returnează 1, atunci cel puțin unul dintre copii returnează 1. Cu probabilitate acest copil este ales mai întâi și cu probabilitate , amândoi subarborii sunt evaluați.

Costul mediu este cel mult .

* Dacă rădăcina returnează 0, amândoi copiii vor trebui evaluați, ceea ce impune un cost de cel puțin .

Considerăm acum un nod rădăcina a unui arbore cu adâncimea .

* Dacă rădăcina returneaza 0, atunci cel puțin unul dintre copiii ai rădăcinii returneaza 0; cu probabilitate 1/2 acest copil este ales mai întâi și costul mediu este cel mult
* Dacă rădăcina returneaza 1, amandoi copiii trebuie evaluați impunând un cost de cel mult

Dacă N este numărul de frunze al unui astfel de arbore, atunci numărul mediu de pași este cel mult