**Agoritmi si structuri de date**

---

Originea algoritmilor ?

Conceptul de algoritm are o istorie lungă care implică inventarea numerelor, a matematicii și a computerelor. Încă din antichitate au fost atestate procedee pas cu pas de rezolvare a problemelor matematice. Aceasta include matematica babiloniană (în jurul anului 2500 î.Hr.), matematica egipteană (în jurul anului 1550 î.Hr.), matematica indiană (în jurul anului 800 î.Hr. și mai târziu; de exemplu, Shulba Sutras, Școala Kerala și Brāhmasphuṭasiddhānta), matematica greacă (în jurul anului 800 î.Hr. și BC Eratosg. algoritm euclidian) și matematică arabă (secolul al IX-lea, de exemplu algoritmi criptografici pentru ruperea codurilor bazate pe analiza frecvenței).

Ce este o structura de date ?

-În informatică, o structură de date este o metodă sistematică de stocare a informațiilor și datelor într-un calculator sub o anumita forma, în așa fel încât ele să poată fi folosite în mod eficient. Deseori o alegere bine făcută a structurii de date va permite și implementarea unui algoritm eficient.

-Exemple de structuri de date: Hash table(care pentru o cheie de un tip de date leaga alta valoare de un tip de date), set(care sorteaza automat orice valoare adaugata), graf etc.

Ce este un algoritm?

-Un algoritm (cuvântul are ca origine numele matematicianului persan Al-Khwarizmi) înseamnă în matematică și informatică o metodă sau o procedură de calcul, alcătuită din pașii sau operațiile elementare necesare pentru rezolvarea unei probleme sau categorii de probleme, adesea foarte eficient sau ce urmaresc un anumit scop.

-Exemple de algoritmi: DFS(parcurgere grafuri), Dijkstra(cel mai scurt drum dintre 2 noduri), ciurul lui Enatostene(determinarea eficienta a numerelor prime) etc.

La ce sunt folositi algoritmii si stucturile de date?

-Algoritmii si structurile de date au ca principal scop rezolvarea unor probleme de diverse tipuri intr-un mod cat mai eficient atat din punct de vedere a timpului de executie cat si a spatiului de stocare, sunt folositi in aproximativ toate domeniile deoarece pentru aproape orice „task” exista un algoritm care poate face procesul mai eficient.

Experiment

-Experimentarea s-a găsit foarte utilă cu algoritmi complecși pentru care nu se cunoaște nicio analiză teoretică tratabilă. Algoritmii pot fi evaluati aplicându-le seturi de cazuri de testare și analizându-le performanța.Testarea a dat caracterizări valoroase ale anumitor metode precum divide-and-cuquer, algoritmi greedy, programare dinamica, interpreți de mașini cu stări finite, interpreți de mașini de stivă,căutări euristice și algoritmi randomizați. Testarea a oferit perspective semnificative asupra performanței algoritmilor paraleli și distribuiți.

Design si proiectare

-Mulți algoritmi utili și practici au fost proiectați și plasați în bibliotecile de programe - de exemplu, software matematic, căutare, sortare, generare de numere aleatoare, potrivire a modelelor textuale, hashing, grafice, arbori, protocoale de rețele de comunicații, actualizări de date distribuite, semafore ,detectoare de blocaj, sincronizatori, manageri de stocare, liste, tabele și algoritmi de paginare. Multe rezultate teoretice au fost traduse în sisteme utile și practice, cum ar fi criptosistemul public RSA, compilatoare de calitate a producției și aspectul circuitului VLSI si multe altele.

# Relatii cu alte subdomenii(dependenta si influenta)

-Algoritmii sunt utilizați în toate domeniile de calcul si nu doar. Pentru că este o modalitate fantastică de automatizare a deciziilor computerizate, multe domenii depind de algoritmi si nu ar putea exista fara ajutorul acestora de aceea algorimii si structurile de date au o influenta directa si puternica asupra o mare parte din domeniile informaticii. De exemplu la sistemele de operare si bazele de date sunt folositi algoritmii de cautare pentru a gasi ceva in device sau in baza de date. Un alt exemplu de folosire a algorimilor ar fi la inteligenta artificiala care se impart in mai multe categorii cum ar fi: Algoritmi de învățare supravegheați, Algoritmi de învățare nesupravegheați, Algoritmi de învățare prin întărire. Un alt exemplu este la Grafica, unde in motoarele de creare a graficii se folosesc o gama vasta de structuri de date dar si algoritmi , cum ar fi: Algoritmi de generare a liniilor DDA, Algoritmul lui Xiaolin Wu, Algoritmul de generare a liniilor lui Bresenham, Algoritm de umplere prin inundare, Algoritmul de umplere a limitelor etc., si multe alte domenii. Iar algoritmii la randul lor depind de experimente, persoane care sa ii poate dezvolta si crea dar si de alti subalgoritmi deja creati.

# Problemele algoritmilor

-Principala problema a algoritmilor este aceea ca nu pot rezolva orice, si in genereal problemele ce nu pot fi rezolvate prin algoritmi se impart in 2 mari categorii:

1. **Probleme de nedecis**.În teoria computabilității, o problemă indecidabilă este un tip de problemă de calcul care necesită un răspuns da/nu, dar în care nu poate exista niciun program de calculator care să ofere întotdeauna răspunsul corect.
Exemplu: „Halting problem" – Este dat un program arbitrar cu un set de date arbitrar decideti daca are o bucla infinita sau nu. Alan Turing a demonstrat ca pentru o asemenea problema nu exista un algoritm general care sa dea mereu un raspuns corect, demonstrand ca pentru orice functie care trebuie sa determine daca o sa fie o bucla infinita sau nu exista o functie care o sa faca functia sa dea un raspuns gresit.
2. **Probleme greu de rezolvat**. Aceste probleme sunt teoretic imposibil de rezolvat într-un timp rezonabil - adică există soluții algoritmice cunoscute, dar algoritmii sunt prea ineficienți/lenti pentru a rezolva problema atunci când numărul de intrări crește.
Exemplu: „Password Guessing"- Consta in ghicirea Brute Force a unei parole de lungime N cu cele 26 de litere, pentu (1 litera 26 combinatii, 2 litere 626 combinatii... 26 litere cu 26^26 combinatii) si daca adaugam si cifre si crestem lungimea numarul de combinatii creste semnificativ, si asa se pentru orice problema care creste exponential si trebuie verificat fiecare caz.

# Persoane Importante

-Edsger W. Dijkstra - Algoritmul există în multe variante. Algoritmul original al lui Dijkstra a găsit calea cea mai scurtă între două noduri date, dar o variantă mai comună fixează un singur nod ca nod „sursă" și găsește cele mai scurte căi de la sursă la toate celelalte noduri din grafic, producând un arbore cu calea cea mai scurtă.

-Charles Pierre Trémaux -
O versiune a primei căutări în profunzime a fost investigată inițial de un matematician francez, Charles Pierre Trémaux, în secolul al XIX-lea, ca strategie de rezolvare a labirinturilor. Căutarea în profunzime (DFS) este un algoritm pentru parcurgerea sau căutarea structurilor de date arborescente sau grafice. Algoritmul începe de la nodul rădăcină (selectând un nod arbitrar ca nod rădăcină în cazul unui grafic) și explorează cât mai departe posibil de-a lungul fiecărei ramuri înainte de a reveni.

-Gennady Korotkevich – (Stie cel mai bine sa foloseasca algoritmii si structurile de date). Gennady Korotkevich este un programator competitiv din Belarus care a câștigat competiții internaționale majore încă de la vârsta de 11 ani, precum și numeroase competiții naționale. Realizările sale de top includ șase medalii de aur consecutive la Olimpiada Internațională de Informatică, precum și campionatul mondial din finala mondială a Concursului Internațional de Programare Colegială din 2013 și 2015. Din decembrie 2022, Gennady este cel mai bine cotat programator pe Codeforces, CodeChef, Topcoder, AtCoder și HackerRank. În ianuarie 2022, a obținut un rating istoric de 3979 pe Codeforces, devenind primul care a depășit bariera de 3900.

-Vojtěch Jarník, Robert C. Prim - Algoritmul lui Prim este un algoritm din teoria grafurilor care găsește arborele parțial de cost minim al unui graf conex ponderat. Înseamnă că găsește submulțimea muchiilor care formează un arbore care include toate vârfurile și al cărui cost este minimizat.

## Carti, reviste, conferinte importante

-Competitive Programmer's Handbook – Este o carte care include toate bazele pentru programare competitiva incluzand algoritmi, structuri de date, teoreme si exemple foarte bune care fac o mare diferenta in cunoasterea si crearea unui algoritm eficient.

- CS50(Harvard) – a fost o conferinta unde s-au discutat totul despre algoritmi si structuri de date, incepand de la limbaje de programare si pana la programare orientata obiect, este o conferinta care si pana la momentul dat este considerata un „must-watch” pentru orice persoana care doreste sa inteleaga cum sa faca si foloseasca un algoritm eficient de la 0.

-Mai putem include toate concursurile de programare cum ar fi google „Kick Start”, IOI, ICPC si pana ca contest-uri de pe Codeforces unde persoanele se aduna pentru a concura in dezvoltarea anumitor algoritmi care uneori sunt foarte complecsi si acolo se gasesc solutii foarte eficiente.

## Cum invatam Alg/DS la UVT?

La UVT am invatat algoritmi, structuri de data si analiza eficientei la ASD1, P1 invatam la ASD2 si P2. Dar pentru doritori au fost si concursuri cum ar fi ONIS, ICPC, SEERC, CCC(Cloudflight Coding Contest) care te ajuta sa intelegi si implementezi un algoritm eficiend si iti ofera si sansa de a ii pune in practica.