Proiectarea Algoritmilor - Pregătire Test Practic

May 11, 2024

Contents

1 Organizare

- Ideea testului este de a simula condițiile de la un interviu de algoritmică. Prima probă de angajare la multe companii poate să aibă exact acest format: Aveți X minute pentru a rezolva corect o problemă pe un site care oferă posibilitatea de evaluare automată. Intervievatorii vor să vadă nu doar că ați găsit o soluție rapidă, dar și că ați scris cod curat și, mai ales, ați luat in considerare toate cazurile particulare care pot să apară.
- Testul va avea loc într-o sâmbătă, pe site-ul hackerrank.com și durează 2 ore.
- Veți avea de rezolvat o problemă ușoară, respectiv o problemă de dificultate medie.
- Problemele vor fi alese din materia studiată:
  - Divide et impera.
  - Greedy.
  - Programare dinamică.
  - Backtracking.
  - Parcurgeri pe grafuri: DFS, BFS. Sortare topologică.
  - Aplicații DFS: identificarea punctelor de articulație, punților, componentelor tare conexe.
  - Drumuri minime.
- Testul este **open-book**, cu câteva constrângeri:
  - Puteți folosi cod scris de voi anterior, respectiv soluțiile de la laboratoare/teme.
  - Pe internet puteți accesa **DOAR**: pagina de ocw[1]/Moodle, documentația oficială a limbajului preferat [2] [3], respectiv pagina de pe hackerrank.com a concursului.
  - Este interzis să deschideți **orice** alt tab în browser/aplicație/plugin pentru comunicare/rezolvarea problemelor. Implicit, **nu puteți folosi modele de AI** pentru rezolvarea problemelor.
- Vă recomandăm să vă aduceți hârtie / pix / apă.

[1] https://ocw.cs.pub.ro/courses/pa
[2] C++: Standard Template Library
[3] Java: java.util, java.util.collections

2 Sugestii Pregătire

- **Foarte important**: Citiți cu atenție **toate enunțurile**. Pare ceva evident dar este prima greșeală pe care o fac foarte multe persoane la testul practic: Fie nu citesc integral enunțul de la o problemă, fie se opresc la prima problema care pare accesibilă și le ignoră pe celelalte.
- Problemele ușoare presupun in general o prelucrare a datelor de intrare și aplicarea unuia dintre algoritmii intâlniți la laborator/curs cu modificări minimale. Pentru a rezolva problemele mai grele poate fi necesară combinarea mai multor tehnici.
- **Nu vă speriați** dacă nu recunoașteți imediat soluția. Citiți cu atenție, integral, enunțul și încercați să reduceți problema la una mai simplă.
- **Nu vă speriați** dacă nu vă merge implementarea din prima. Dacă ați petrecut deja mult timp pe o problemă și primiți același mesaj de eroare **uitați-vă și pe altă problemă și reveniți ulterior**. E important să nu rămâneți blocați pe o anumită idee - uneori ajută să faceți o mică pauză pentru a putea căuta o soluție și din alt unghi.
- În enunț sunt descrise atât dimensiunile maxime ale testelor folosite cât și limitele de timp. Această informație este foarte importantă pentru că vă sugerează **care este complexitatea acceptabilă**.
- Unul din aspectele cele mai importante evaluare la test este abilitatea de a face debug. Mini-Tutorial debugging.
- Pentru debug/punctaj parțial poate ajuta și implementarea unei **soluții mai simple**, cu complexitate mai mare dar care garantează rezultate corecte. Împreună cu un generator automat de teste vă poate ajuta să descoperiți soluția eficientă (ex: relația de recurență la o problemă de PD) sau să găsiți cazurile particulare pe care soluția eficientă pică.
- Este importantă și **experiența de implementare**. Implementarea corectă în condiții de examen a unui algoritm este o experiență diferită de înțelegerea conceptului din punct de vedere teoretic. Prin urmare, vă sugerez să implementați cel puțin un exemplu din fiecare categorie.
- **Familiarizați-vă cu algoritmii și structurile de date din bibliotecile standard** ale limbajului preferat. [4] [5]) Nu are rost să scrieți la test propriul algoritm dacă aveți disponibilă o implementare eficientă în biblioteca standard la un apel de funcție distanță.
- **Familiarizați-vă** și cu platforma Hackerrank pe care o să aibă loc testul rezolvând câteva probleme pe site. Sugestia mea este să lucrați local și doar la final să uploadați pe site (să nu se piardă soluțiile dacă se întamplă ceva cu site-ul/browserul).
- Dacă în problemă este specificat că trebuie să rezolvați mai multe teste la aceeași rulare, nu uitați să resetați structurile de date între teste. În general, **aveți grijă la inițializarea datelor**.
- Când vă pregătiți - **nu căutați soluția problemei imediat** pe Google. Este esențial să puteți să găsiți și singuri soluția, altfel nu o să fie foarte productivă pregătirea.
- **Simulați** condițiile de la test (problemele ușoare/medii). Încercați să rezolvați în aceleași condiții problemele date în 2018 sau 2017:
  - Test 2018 Varianta 1
  - Test 2018 Varianta 2
  - Test 2017 Varianta 1
  - Test 2017 Varianta 2
- Uitați-vă și pe laborator și încercați să rezolvați problemele de la bonus.
- Nu în ultimul rând: Citiți cu **atenție**, **integral**, **toate enunțurile**!

[4] C++: Standard Template Library
[5] Java: java.util, java.util.collections

3 Probleme

Această secțiune conține o listă cu probleme de la testele practice din anii anteriori. La final aveți și câteva hint-uri - uitați-vă doar dacă ați petrecut cel puțin 15-20 min. pe o problemă și nu ați găsit nici o soluție acceptabilă.

1. Graph Cycle - Problemă ușoară, Test Practic 2015
2. Play With Strings - Problemă ușoară, Test Practic 2015
3. Binary Strings - Problemă medie, Test Practic 2015
4. Autobuz - Problemă medie, Test Practic 2015
5. Secv1 - Problemă ușoară, Test Practic 2015
6. Game1 - Problemă medie, Test Practic 2015
7. Cycle - Problemă ușoară, Test Practic 2015
8. Cypher - Problemă medie, Test Practic 2015
9. Labirint - Problemă ușoară, Test Practic 2016
10. Excursie - Problemă ușoară, Test Practic 2016
11. Magazine - Problemă medie, Test Practic 2016
12. Game of PA - Problemă medie, Test Practic 2016
13. TopSort - Problemă ușoară, Test Practic 2016
14. Unique edge - Problemă ușoară, Test Practic 2016
15. ACS Impact - Problemă ușoară, Test Practic 2016
16. Florărie - Problemă ușoară, Test Practic 2016
17. Mate - Problemă medie, Test Practic 2016
18. Super-glue chess - Problemă medie, Test Practic 2016
19. Testul surpriză - Problema ușoară, Test Practic 2019
20. Generare șiruri - Problemă ușoară, Test Practic 2019
21. Număr minim de operații - Problemă medie, Test Practic 2019
22. Conversie valutară - Problemă grea, Test Practic 2019
23. Cursa deterministă - Problemă grea, Test Practic 2019
24. Matei si diamantele - Problemă ușoară, Test Practic 2019
25. Gigel și răspândirea veștilor - Problemă ușoară, Test Practic 2019
26. Gigel si sosetele - Problemă medie, Test Practic 2019
27. Gigel si telefonul - Problemă medie, Test Practic 2019
28. Organizare mafiotă - Problemă grea, Test Practic 2019
29. Cutiile magice - Problemă grea, Test Practic 2019

Soluții/Hint-uri

1.

- Dacă în timpul unei parcurgeri DFS, descoperiți o muchie de la nodul curent la un nod gri (aflat în curs de vizitare) X, înseamnă că nodul curent este accesibil din X și, de asemenea, X este accesibil din nodul curent.
- Atenție! Graful de intrare poate să nu fie conex.

2.

- Greedy

3.

- Programare dinamică.
- Simulați răspunsul corect pentru primele valori ale lui N ca să descoperiți relația de recurență. De câte soluții precedente avem nevoie pentru a calcula soluția pentru dimensiunea i?
- Din păcate, valoarea maxima a lui N ne sugerează că nu putem rezolva problema în $O(N)$.
- Trick: Exponențiere logaritmică pe matrice

4.

- Sunt mai multe abordări posibile.
- $d[i]$ - numărul maxim de pasageri care pot fi transportați din nodul 1 în nodul i. Dacă știm $d[i]$ cum putem actualiza această valoare pentru vecinii nodului i?
- Alternativ, putem incerca să "ghicim" răspunsul. Presupunem că soluția este X și verificăm dacă întradevăr există un drum de la 1 la N care permite transportarea a minim X pasageri. În funcție de răspuns, vom încerca în continuare cu o valoare mai mică sau mai mare.

5.

- Greedy.

6.

- Simulați jocul pentru câteva valori mici ale lui N și M. Cum s-ar putea rezolva problema pentru $M = 1$?
- Jucătorii mută alternativ, nu pot sta pe loc și nu pot muta "înapoi". De pe anumite poziții doar un anume jucător se poate afla la mutare.
- Cum putem descompune problema în subprobleme? (sa vizualizăm soluția ca o secvență de decizii) - Programare dinamică.

7.

- Problema se reduce la a găsi un ciclu într-un graf neorientat de dimensiune 3.
- Nu este necesară reprezentarea explicită a grafului în memorie.
- DFS.

8.

- În problemă este descrisă starea inițială a cifrului, starea finală și tranzițiile posibile și ni se cere să descoperim o secvență cu un număr minim de tranziții intre cele două stări. Acesta lucru ar trebui sa ne ducă cu gândul la o problemă de drum minim.
- Vrem sa minimizăm doar **numărul** de tranziții, deci este suficientă o parcurgere in lățime prin graful de stări pentru a găsi soluția.

9.

- BFS.

10.

- Ni se cere drumul de cost minim între oricare două noduri iar în caz de egalitate să alegem drumul cu cele mai multe noduri intermediare.
- Dimensiunile datelor problemei ne sugerează că o complexitate $O(N^3)$ este acceptabilă - putem adapta direct algoritmul Roy-Floyd.

11.

- Tot o problemă de drum minim, doar că avem **Na** surse și **Nm** destinații.
- Putem rezolva problema fără să apelam algoritmul de drum minim separat pentru fiecare sursă?

12.

- Graful este orientat aciclic. Câte drumuri sunt posibile între două noduri?
- Dimensiunile mari de intrare ne sugerează că nu putem explora exhaustiv toate drumurile din graf.
- Putem simula pentru câteva grafuri mici. De la ce noduri am porni calcularea răspunsului?
- Dacă pentru fiecare descendent X al unui nod Y cunoaștem cele două valori:
  - $maxSum[X]$ - suma maximă pe care o putem acumula pornind din X.
  - $maxPower[X]$ - puterea minimă necesară pentru a acumula $maxSum[X]$.

  Putem calcula aceste valori și pentru Y?

13.

- Trebuie adaptat algoritmul lui Kahn prezentat in laborator astfel încât să selecteze întotdeauna nodul cu cel mai mic indice dintre cele care pot fi prelucrate.

14.

- Soluția trivială: Dacă eliminăm o anumită muchie, mai putem ajunge din A în B?
- DFS.
- Un test conține mai multe instanțe de grafuri: Aveți grijă la implementare!

15.

- Intuiție Greedy - Dacă avem de ales intre X nivele pe care le putem selecta, vom alege acel nivel care ne garanteaza punctaj maxim la final.
- Dimensiunile problemei ne sugerează că nu putem construi explicit graful de tranziții - dar acest lucru nu este necesar.

16.

- Sunt mai multe abordări posibile; în principiu trebuie exploatată structura specială a datelor de intrare pentru a evita verificări redundante in matrice.
- **Greedy**: $O(K * (N + M))$: pornim de la cea mai mare valoare și apoi ne deplasam pe linii/coloane pana ajungem la o valoare mai mica sau egala decât cea cautată.
- Este posibilă și o soluție mai simplă, cu căutare binară, în $O(K * log(N * M))$.

17.

- Programare dinamică - este o problemă foarte asemănătoare *cu cel mai lung subșir comun.*

18.

- Backtracking - este o reformulare a unei probleme clasice.
- Atenție la implementare.

19.

- Care este soluția pentru $K = 2$?
- Greedy.

20.

- Cerința este să **generăm** toate șirurile, $N \leq 15$, care au o anumită proprietate.
- Backtracking.

21.

- Mai generic: În problemă este descrisă o stare inițială, o stare finală și tranzițiile posibile dintre stări. Se cere numărul minim de tranziții pentru a ajunge din starea inițială în starea finală.

22.

- Similar ca la problema anterioară, trebuie să recunoaștem cum putem construi un graf pornind de la datele de intrare.

23.

- BFS din fiecare țintă pentru calcularea distanței spre fiecare alergător + alegere greedy.

24.

- Trebuie să identificăm pozițiile pe care se poate ajunge pornind din căsuța inițială, în funcție de distanța Manhattan.

25.

- Cum putem modela un graf pornind de la datele de intrare?

26.

- Greedy.

27.

- Programare dinamică - Care este costul minim necesar pentru a ajunge in intersecția i cu j unități de baterie?

28.

- Trebuie să verificăm dacă există o cale între **oricare** două noduri X și Y ($X \to Y$ sau $Y \to X$).
- Putem adapta un algoritm de sortare topologică studiat la laborator.

29.

- Programare dinamică - Probabilitatea ca bila să ajungă în cutia i dupa j pași
  - Cum putem actualiza această valoare dacă știm răspunsul pentru $j - 1$ pași?
- Bonus - Cum am putea aborda problema folosind trucul de exponențiere pe matrice?