Rundele #41 - #50

Cosmin-Silvestru Negruțeri

În cadrul acestui articol vom publica detalii despre desfășurarea celor 10 probleme propuse spre rezolvare la semifinala ediției 2003/2004 a concursului de programare Bursele Agora, organizat de revista noastră.

P040429: Minele de aur

Problema ne cere să determinăm o dreaptă care împarte o mulțime de puncte în două mulțimi cu același cardinal. Pentru a rezolva problema, sortăm punctele în funcție de coordonata x și returnăm ca soluție dreapta verticală care trece prin punctul cu indexul n din șirul sortat al punctelor. Este evident că această dreaptă conține cel putin n puncte pe ea și în stânga ei, și mai conține cel putin n puncte pe ea și în dreapta ei.

Analiza complexității

Ordinul de complexitate a operațiilor de citire a datelor de intrare și găsire a elementului median dintr-un șir este $O(n)$.

Operația de scriere a rezultatelor are ordinul de complexitate $O(1)$.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(n)$.

Au rezolvat corect...

Dintre cei 25 de concurenți care au participat la această etapă a concursului, șase au reușit să obțină punctajul maxim.

Mugurel-Ionuț Andreica, România
Adrian Cârcu, România
Liviu Ciortea, România
Marius Dumitran, România
Marius Nicolae, România
Claudiu Rad, România

P040430: Rivali

Aceasta problemă cere determinarea diferenței costurilor dintre arborele de cost minim și al doilea arbore de cost minim al unui graf.

O primă metodă ar fi determinarea arborelui de cost minim și eliminarea câte unei muchii, care aparține acestui arbore, din graf, și apoi determinarea noului arbore parțial pentru noul graf. Acum putem determina diferența dintre costul primului arbore obținut și costul celui mai mic obținut pentru cei n-1 arbori determinați ulterior. Datorită restricției (n = 500) această rezolvare nu s-ar fi încadrat în timp pe ultimele teste.

Altă metodă ar fi cea prezentată în cartea *Introducere în algoritmi, T. H. Cormen, C. E. Leiserson, R. R. Rivest* în problema propusă 24-1. Ideea prezentată acolo este următoarea: dacă vrem să adăugam muchia (i, j) arborelui de cost minim, mai întâi trebuie să eliminăm o muchie pentru a obține un arbore. Această muchie nou introdusă ar induce un ciclu, pe de altă parte noul nostru arbore trebuie să aibă cost minim, vom elimina muchia maximă din ciclul indus de muchia (i, j).

Parcurgând toate muchiile care nu aparțin arborelui parțial de cost minim și încercând să le adăugăm la arborele nostru, obținem al doilea arbore parțial de cost minim. Ceea ce rămâne de făcut este eficientizarea operației de determinare a muchiei maxime din ciclul indus de muchia (i, j). Putem să facem ca această operație să se execute în timp constant după o preprocesare a grafului.

Realizăm aceasta prin determinarea valorilor unei matrice M cu semnificația $M_{i,j}$ reprezintă costul maxim al unei muchii de pe drumul de la nodul i la nodul j în arborele parțial de cost minim. Acum determinarea costului arborelui în care s-a introdus muchia (i, j) este:

$$cost\_arbore\_inițial + cost(i, j) - M_{i,j}.$$

Determinarea câte unei linii din matricea M se poate face prin efectuarea unei parcurgeri în adâncime pe arborele parțial de cost minim, pornind de la nodul ale cărui linii din matrice vrem să o determinăm.

Analiza complexității

Ordinul de complexitate al operației de citire a datelor de intrare este $O(m)$.

Determinarea arborelui parțial de cost minim cu ajutorul algoritmului lui *Prim* are ordinul de complexitate $O(n^2)$. Preprocesarea unei linii din matricea M are complexitatea unei parcurgeri în lățime a unui graf cu n-1 muchii care este egală cu $O(n)$, deci ordinul de complexitate al preprocesării este $O(n^2)$.

Ordinul de complexitate al determinării celui de-al doilea arbore de cost minim este $O(m)$, după faza de preprocesare.

Ordinul de complexitate al operației de scriere a rezultatelor este $O(1)$.

În concluzie ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(n^2)$.

Au rezolvat corect...

Dintre cei 24 de concurenți care au participat la această etapă a concursului, șapte au reușit să obțină punctajul maxim.

Mugurel-Ionuț Andreica, România
Adrian Diaconu, România
Haryanto Lego, Statele Unite ale Americii
Marius Nicolae, România
Mircea Pașoi, România
Mihail-Cosmin Pițˇ-Rada, România
Alexandru Popa, România

P040431: Cerc

În continuare vom prezenta trei metode de rezolvare pentru această problemă. Luăm două puncte consecutive de pe înfășurătoarea convexă a punctelor date (pentru simplitate putem alege cel mai de jos punct și punctul cel mai din dreapta față de cel mai de jos punct), fie ele A și B.

Considerăm unghiurile formate de celelalte n - 2 puncte cu punctele A și B. Dacă avem alte două puncte P și Q și relația $\angle APB \leq \angle AQB$, atunci punctul Q este situat în interiorul cercului circumscris triunghiului APB. Tragem concluzia că pentru a avea k puncte incluse într-un cerc, putem sorta descrescător cele n - 2 puncte în funcție de unghiul format cu punctele A și B, și putem returna cercul circumscris punctelor A, B și punctului cu indexul k - 2 din șirul sortat al celorlalte n - 2 puncte. Determinarea elementului cu indexul k - 2 se poate face cu un ordin de complexitate medie $O(n)$ folosind un algoritm de selecție aleatoare.

Ne este accesibilă și altă rezolvare datorită mărginirii domeniului punctelor la coordonata maximă 10000. Ca și mai sus, luăm două puncte consecutive A și B de pe înfășurătoare și apoi facem o căutare binară pentru a determina dimensiunea razei cercului care conține k puncte în interior. are cele două puncte A și B pe circumferință, iar centrul este în semiplanul în care sunt celelalte n - 2 puncte.

O a treia rezolvare, ar fi să considerăm centrul cercului punctul de coordonate (0, 0) și apoi să calculăm distanțele de la fiecare punct la centrul cercului. Acum soluția care trebuie returnată este a k-a distanță din șirul sortat al distanțelor. Acest număr îl găsim folosind metoda de selecție aleatoare.

Primele două rezolvări determină un cerc care trece prin cel puțin trei puncte din cele n puncte date.

Analiza complexității

Ordinul de complexitate al operației de citire a datelor de intrare este $O(n)$, iar operația de scriere a rezultatelor are ordinul de complexitate $O(1)$.

În cazul primelor două metode de rezolvare prezentate, operația de determinare a punctelor A și B are ordinul de complexitate $O(n)$.

În cadrul primei metode de rezolvare, determinarea unghiurilor formate de cele n - 2 puncte rămase cu punctele A și B are ordinul de complexitate $O(n)$, iar determinarea elementului k - 2 din șirul sortat al unghiurilor are ordinul de complexitate $O(n)$.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme, folosind prima metodă prezentată, este $O(n)$.

Pentru cea de-a doua metodă de rezolvare, numărul de pași pentru determinarea razei r a cercului este $O(\log r)$, și la fiecare pas trebuie să numărăm câte puncte sunt interioare cercului de rază r care trece prin punctele A și B, pas care are ordinul de complexitate $O(n)$.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme, folosind cea de-a doua metodă prezentată, este $O(n \cdot \log r)$.

Determinarea distanțelor de la cele n puncte până la punctul de coordonate (0, 0) pentru cea de-a treia metodă de rezolvare, are ordinul de complexitatea $O(n)$, iar determinarea elementului al k-lea din șirul sortat al distanțelor are ordinul de complexitate $O(n)$. Deci, ordinul de complexitate al algoritmului de rezolvare folosind această metodă este $O(n)$.

A rezolvat corect...

Dintre cei 13 de concurenți care au participat la această etapă a concursului, doi au reușit să obțină punctajul maxim.

Mugurel-Ionuț Andreica, România
Marius Dumitran, România

P040432: Parola

Această problemă este similară cu problema *Parola Ascunsă* propusă spre rezolvare la etapa regională a concursului *ICPC* organizat de *ACM* anul trecut. O rezolvare cu ordinul de complexitate $O(n \cdot \log^2 n)$ a fost prezentată în revista *GInfo* 13/8 de către *Mihai Stroe*.

Vom prezenta în continuare o soluție care are ordinul de complexitate $O(n \cdot \log n)$. Fie S_i șirul rotit care începe la indexul i.

Trebuie să determinăm rotația care este minimă în ordine lexicografică. Pentru a face aceasta, vom compara șirurile în forma în care am organiza confruntările la un turneu de tenis, câștigătorul turneului fiind rotația care este minimă în ordine lexicografică.

Să vedem ce se întâmplă într-un meci al turneului:

Fie șirul inițial împărțit în trei secvențe A, B și C. La pasul actual trebuie să comparăm șirul ABC cu șirul BCA, unde A și B au aceeași lungime.

Dacă $A < B$, atunci ABC este câștigătorul meciului dintre ABC și BCA.

Dacă $A > B$, atunci BCA este câștigătorul meciului dintre ABC și BCA.

Dacă $A = B$, atunci îl luăm pe ABC ca și câștigător. Dacă decizia noastră este greșită și $ABC > BCA$, atunci înseamnă că $B > C$ și $A > C$, deci la un moment ulterior al turneului, BCA ar fi înfrânt de CAB sau de învingătorul lui CAB, deci, prin ignorarea lui BCA nu am alterat rezultatul final.

Când comparăm șirul S_i cu șirul S_j trebuie să comparăm $j - i$ elemente (practic subsecvențele notate cu A și B).

La runda k a turneului șirurile care trebuie comparate vor fi la distanța cel mult 2^{k-1} și vor fi $n/2^k$ șiruri de comparat.

Deci complexitatea unei runde este $O(n)$ și numărul total de runde este log n.

Algoritmul în pseudocod al rezolvării este următorul:

```
Pentru i =1, n execută:
   //în w păstrăm învingătorii rundei anterioare
   adaugă pe i la w
sfârșit pentru
cât timp |w| > 1 execută:
   inițializează(t)
   cât timp |w| > 1 execută:
      compară primele w2 - w1 litere ale rotației w1 și
      rotației w2
      dacă (w1 > w2) atunci
         adaugă pe w2 la t
      altfel
         adaugă pe w1 la t
      sfârșit dacă
         șterge din w pe w1 și w2
   sfârșit cât timp
   dacă |w| = 1 atunci
      adaugă pe w1 la t
   sfârșit dacă
   w ← t
sfârșit cât timp
```

Analiza complexității

Ordinul de complexitate al operației de citire a datelor de intrare este $O(n)$.

Ordinul de complexitate al algoritmului de determinare a șirului cerut este $O(n \cdot \log n)$.

Operația de scriere a rezultatelor are ordinul de complexitate $O(1)$.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(n \cdot \log n)$.

Au rezolvat corect...

Dintre cei 25 de concurenți care au participat la această etapă a concursului, patru au reușit să obțină punctajul maxim.

Mugurel-Ionuț Andreica, România
Adrian Cârcu, România
Marius Dumitran, România
Mircea Pașoi, România

P040433: Coridorul

Un cerc de rază R dată nu poate trece dintr-un capăt al coridorului în altul , dacă există o succesiune de puncte în care fiecare două puncte sunt la distanță mai mică decât $2 \cdot R$, primul punct este la distanță mai mică decât $2 \cdot R$ de marginea de *Nord* a coridorului, iar ultimul punct este la distanță mai mică de $2 \cdot R$ de marginea de *Sud* a coridorului.

Determinarea existenței unui astfel de lanț pentru o rază dată se face ușor printr-o căutare în lățime pe graful în care se consideră existența unei muchii între două puncte atunci când distanța dintre ele este mai mică decât $2 \cdot R$. De aici se conturează ideea de a face o căutare binară pentru a afla dimensiunea razei maxime. Dacă nu avem grijă la implementare, o astfel de rezolvare poate depăși ușor timpul de execuție impus în enunț.

O altă rezolvare care se bazează pe aceeași idee este determinarea pentru fiecare punct a numărului d_i cu semnificația că d_i reprezintă valoarea minimă a muchiei maxime de pe orice lanț cu care se poate ajunge de la peretele din *Nord* la punctul *i*. Acest tablou poate fi determinat ușor, folosind un algoritm similar algoritmului *Dijkstra*.

Rezultatul nostru este egal cu:
min(max(d_i, distanța de la punctul *i* la peretele de sud))/2.

Analiza complexității

Ordinul de complexitate al operației de citire a datelor de intrare este $O(k)$.

Numărul de iterații a căutării binare pentru prima metodă de rezolvare este $O(\log R)$, iar ordinul de complexitate al căutării în adâncime este $O(k^2)$. Deci acest pas din rezolvare are ordinul de complexitate $O(k^2 \cdot \log R)$.

În cazul celei de-a doua metode de rezolvare ordinul de complexitate al algoritmului este același cu cel al algoritmului lui *Dijkstra* de găsire a drumurilor minime, adică $O(k^2)$.

Operația de scriere a rezultatelor are ordinul de complexitate $O(1)$.

În concluzie ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(k^2 \cdot \log R)$, dacă folosim prima metodă și $O(k^2)$, dacă folosim cea de-a doua metodă.

A rezolvat corect...

Dintre cei 14 de concurenți care au participat la această etapă a concursului, unul singur a reușit să obțină punctajul maxim.

Silviu-Ionuț Gănceanu, România

P040434: Ordinea

Problema este o variantă a problemei clasice numită problema lui *Josephus*. Dimensiunea lui N a fost aleasă destul de mică și o rezolvare care folosea o listă circulară înlănțuită și scotea pe rând din listă câte un copil ar fi luat 86 de puncte, ceea ce este destul de mult având în vedere că pe celelalte probleme propuse se scoteau mai dificil puncte fără rezolvări optime.

Să prezentăm acum două soluții bazate pe același principiu care ar fi luat punctaj maxim.

Ideea este că la fiecare pas se scoate un element din listă și trebuie găsit un element cu un index determinat (dacă elementele eliminate la pasul curent k era la poziția i, atunci elementul de eliminat la pasul următor este pe poziția $(i - 1 + k)$ modulo $(n - i) + 1$). Dacă am folosi un șir de indecși, atunci găsirea unui element de index p cere timp constant, iar ștergerea unui element de pe poziția p ar avea complexitate medie $O(n)$, complexitatea operațiilor este prea mare. Prezentăm în continuare două structuri de date care pot fi folosite cu succes în rezolvarea problemei.

Prima structură de date: Un șir num_i cu semnificația numărul elevilor cu numărul de ordine între i și $i + [\sqrt{n}]$ care mai sunt în listă. Pentru a elimina un element de index p din listă trebuie să actualizăm valorile lui num pentru indecșii $p - [\sqrt{n}]$, ..., p. Pentru a găsi elementul al p-lea care nu a fost încă eliminat, parcurgem în salturi de $[\sqrt{n}]$ și ajungem la un index i, astfel încât al p-lea element se află între i și $i + \sqrt{n}$, deci pentru găsirea celui de-al p-lea element care nu a fost încă eliminat avem maxim $2 \cdot \sqrt{n}$ operații.

A doua structură de date: Folosim un arbore binar complet care conține 2^x noduri astfel ca 2^x să fie mai mare sau egal cu $2 \cdot n$. Nodul rădăcină păstrează informație despre elementele 1, ..., n ale șirului, fiul stâng păstrează informații despre elementele 1, ..., $[(n + 1)/2]$ și fiul drept despre elementele $[(n + 1)/2] + 1$, ..., n, deci un nod fiu stâng păstrează jumătatea din stângă a numerelor păstrate de nodul tată, iar fiul drept păstrează jumătatea dreaptă a numerelor păstrate de nodul tată Informația păstrată pentru nodul asociat intervalului (i, j) este numărul de elemente, cu indecși între i și j inclusiv, existente în lista noastră. Ștergerea din structura noastră de date are complexitatea $O(\log n)$ pentru că un copil este cuprins în $\log n$ intervale $[i, j]$ monitorizate în arbore. Găsirea elementului de index p din listă se face similar cu procedeul de determinare al elementului p dintr-un șir de la selecția aleatoare, complexitatea acestei operații fiind și ea $O(\log n)$.

Analiza complexității

Ordinul de complexitate al operației de citire a datelor de intrare este $O(1)$, iar operația de scriere a rezultatelor are ordinul de complexitate $O(n)$.

Căutarea și scoaterea celor n elemente din structura de date folosind prima structură de date prezentată are ordinul de complexitate $O(n \cdot \sqrt{n})$.

În cazul celei de-a doua structuri de date căutarea și scoaterea celor n elemente din structura de date $O(n \cdot \log n)$.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $\boldsymbol{O(n \cdot \sqrt{n})}$ dacă folosim prima structură de date și $O(n \cdot \log n)$, dacă folosim cea de-a doua structură.

Au rezolvat corect...

Dintre cei 32 de concurenți care au participat la această etapă a concursului, șapte au reușit să obțină punctajul maxim.

Mugurel-Ionuț Andreica, România
Adrian Cârcu, România
Liviu Ciortea, România
Marius Dumitran, România
Silviu-Ionuț Gănceanu, România
Mircea Pașoi, România
Sorin Stancu-Mara, România

P040435: Pioni

Această problemă este o aplicație de bază pentru numerele *Sprague-Grundy*, pentru o analiză a ei citiți articolul *Teoria jocurilor: numerele SPRAGUE GRUNDY*, de la pagina 19.

Analiza complexității

Ordinul de complexitate al operației de citire a datelor de intrare este $O(m + t \cdot n)$.

Ordinul de complexitate al sortării topologice a nodurilor grafului este $O(n + m)$. Determinarea valorilor *Sprague Grundy* pentru noduri are complexitatea $O(n \cdot$ valoarea maximă $SG) \leq O(n^2)$.

Ordinul de complexitate al rezolvării unui caz este $O(n)$.

Operația de scriere a rezultatelor are ordinul de complexitate $O(t)$.

În concluzie ordinul de complexitate al algoritmului de rezolvare a acestei probleme este mai mic decât $\boldsymbol{O(m + n \cdot t + n^2 + t)}$.

Au rezolvat corect...

Dintre cei 10 de concurenți care au participat la această etapă a concursului, șase au reușit să obțină punctajul maxim.

Chattopadhyay Amit, India
Mugurel-Ionuț Andreica, România
Marius Dumitran, România
Silviu-Ionuț Gănceanu, România
Mircea Pașoi, România
Adrian Vladu, România

P040436: Matrice binară

Orice două dreptunghiuri cu laturile paralele cu axele de coordonate, care nu se intersectează pot fi despărțite printr-o dreaptă verticală sau una orizontală. Să ne ocupăm acum de determinarea ariei maxime care poate fi acoperită de două dreptunghiuri care conțin numai valoarea 0, dreptunghiuri care pot fi despărțite printr-o dreaptă orizontală (cazul cu dreapta verticală se rezolvă analog).

Dacă am avea pentru fiecare linie i calculate numerele u_i (aria maximă a unui dreptunghi care conține numai 0, situat mai sus de linia $i + 1$) și l_i (aria maximă a unui dreptunghi care conține numai 0, situat mai jos de linia i), atunci rezultatul nostru ar fi $\max(u_i + l_i, \forall 1 \leq i \leq n)$. Să vedem acum, cum putem calcula u_i, calcularea lui l_i făcându-se analog.

Am redus astfel problema la determinarea ariei maxime a unui dreptunghi care conține numai 0. Această pro-

blemă se poate rezolva în mai multe moduri, rezolvarea optimă având complexitatea $O(n \cdot m)$, unde m și n sunt dimensiunile matricei în care căutăm dreptunghiul. Ea a fost prezentată pe lista de discuții a lui Cătălin Frâncu, fost olimpic, și poate fi găsită la adresa:
http://probleme.francu.com/arhiva/R0038/rezolv.html.

O rezolvare care s-ar fi încadrat în timp ar fi fost următoarea: pentru elementul j al liniei i avem calculat numărul h_j care reprezintă înălțimea bandei continue de zerouri care pornește de la celula (i, j) și merge în sus. Aria dreptunghiului care conține numai 0 și care are colțul din dreapta-jos în celula (i, j) și colțul din stânga-jos în (i, k) este $(j - k + 1) \cdot \min(h_p)$, $k \le p \le j$. Dacă avem șirul h calculat pentru linia curentă, putem determina aria maximă a unui dreptunghi care conține numai 0, cu colțul de dreapta-jos în (i, j) cu un algoritm care are ordinul de complexitate $O(n)$.

Analiza complexității
Ordinul de complexitate al operației de citire a datelor de intrare este $O(m \cdot n)$.

Ordinul de complexitate al determinării ariei maxime a unui dreptunghi situat mai sus de linia $i + 1$ este $O(n) \cdot O(n \cdot i)$. Complexitatea totală a determinării elementelor șirului u este $O(n^3)$.

Operația de scriere a rezultatelor are ordinul de complexitate $O(1)$.

În concluzie ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(n^3)$. Această complexitate poate fi redusă la $O(n^2)$, dacă rezolvăm subproblema determinării ariei maxime a unui dreptunghi care conține numai 0 urmând algoritmul prezentat pe "*Lista lu' Frâncu*".

Au rezolvat corect...
Dintre cei 9 concurenți care au participat la această etapă a concursului, patru au reușit să obțină punctajul maxim.

Mugurel-Ionuț Andreica, România
Silviu-Ionuț Gănceanu, România
Marius Nicolae, România
Sorin Stancu-Mara, România

P040437: Numere

În această problemă accentul se pune mai mult pe atenție decât pe partea de algoritmică. Trebuie să tratăm multe cazuri. O funcție care ne va fi utilă în rezolvarea problemei este cea a determinării la ce poziție în șir am scris numărul k, această funcție va avea complexitatea $O(\log N)$, unde N = 1.000.000.000.

Vom da acum câteva cazuri:
- `798`**`99`**`10010110` – `99` generează secvența căutată;
- `2345612345` – numărul generator este `123456`;
- `1771` – numărul generator este `717`;
- `567234` – numarul generator este `234567`.

Ultimele trei cazuri sunt mai dificile decât primul.

La fiecare caz determinăm poziția în șir a numărului care generează secvența folosind funcția ajutătoare (numărul total de cazuri este mic, deci rezolvarea pentru dimensiunea testelor va avea timp de rulare foarte mic).

Analiza complexității
Ordinul de complexitate al operației de citire a datelor de intrare și scriere a rezultatelor este $O(t)$.

Rezolvarea are complexitatea $O(nr \cdot \log N \cdot t)$, unde nr reprezintă numărul de cazuri care trebuie tratate, iar t reprezintă numărul maxim de teste.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(nr \cdot \log N \cdot t)$.

A rezolvat corect...
Dintre cei 15 de concurenți care au participat la această etapă a concursului, unul singur a reușit să obțină punctajul maxim.

Mugurel-Ionuț Andreica, România

P040438: Cercuri

O rezolvare polinomială pare să fie greu de găsit pentru această problemă, și în cazul a numai două cercuri apar o grămadă de cazuri care trebuie tratate. Rezolvarea unei variante mai simple a problemei ne va ajuta în soluția problemei generale. Varianta mai simplă care este ușor de rezolvat este determinarea pătratului de latură minimă cu laturile paralele cu axele de coordonate care conține în el cercurile date. Acum, pentru a determina un pătrat de latură minimă care acoperă cercurile, rotim puțin câte puțin sistemul de coordonate față de origine, și pentru fiecare rotație determinăm latura minimă a pătratului care conține cercurile și are laturile paralele cu axele de coordonate. Dacă unghiul de rotație este destul de mic vom obține latura minimă a pătratului cu o precizie de 8 zecimale.

Analiza complexității
Ordinul de complexitate a operațiilor de citire a datelor de intrare și scriere a rezultatelor este $O(1)$.

Determinarea laturii pătratului cerut are ordinul de complexitate $O(\pi/2/r)$, unde r reprezintă numărul de rotiri ale sistemului de coordonate.

Trebuie să fim atenți ca numărul de rotații să fie suficient de mic pentru ca rezolvarea noastră să se încadreze în timp, și suficient de mare ca să obținem lungimea laturii cu o precizie de 8 zecimale.

În concluzie ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(\pi/2/r)$.

A rezolvat corect...
Dintre cei 15 de concurenți care au participat la această etapă a concursului, unul singur a reușit să obțină punctajul maxim.

Mugurel-Ionuț Andreica, România