Descrierea Soluțiilor
Olimpiada Societății pentru Excelență și Performanță în Informatică
Etapa barajelor pentru selecția echipelor naționale
Baraj 3 Juniori

1 Problema Aprox

Propunător: prof. Cheșcă Ciprian, Liceul Tehnologic "Grigore C. Moisil", Buzău

Subtask 1 ($N \leq 500$) – 14 puncte

Se generează toate perechile de numere naturale $(a, b), 1 \leq a, \leq N, 2 \leq b \leq N$ pentru simularea tuturor fracțiilor cu proprietățile cerute de enunțul problemei și se determină fracția care are valoarea cea mai apropiată de numărul x.

Complexitatea temporală a soluției este $O(N^2)$.

Subtask 2 ($N \leq 1\ 000\ 000$) – 31 de puncte

Să presupunem că reținem numărul x sub forma unei fracții ireductibile: $x = \frac{p}{q}$.

Pentru fiecare numitor posibil $1 \leq b \leq N$ calculăm cel mai mare număr natural a astfel încât $\frac{a}{b} \leq \frac{p}{q}$ astfel: $a = \left[\frac{b \cdot p}{q}\right]$. (cu $[x]$ am notat partea întreagă a lui x).

Observăm că pentru un b fixat, avem: $\frac{a}{b} \leq \frac{p}{q} \leq \frac{a+1}{b}$, deci nu este nevoie să iterăm prin toate perechile (a, b), ci doar să ne uităm la aceste două perechi.

La final alegem pe aceea cu diferența minimă față de $\frac{p}{q}$.

Complexitatea temporală a soluției este $O(N)$.

Subtask 3 ($N \leq 1\ 000\ 000\ 000$) – 55 de puncte

Varianta 1 - student Theodor-Gabriel Tulbă-Lecu

Folosim următorul algoritm de secvențe Farey pentru aproximarea numerelor reale.

Ideea este să începem cu două fracții, $\frac{0}{1}$ și $\frac{1}{1}$ ce reprezintă două aproximări inițiale ale numărului x, să le denumim L și R.

La fiecare nouă etapă calculăm o nouă fracție, pe care să o denumim M, prin adunarea numărătorilor fracției L și R și respectiv adunarea numitorilor, apoi determinăm ce poziție are pe axă numărul x față de această nouă fracție și restrângem corespunzător intervalul, similar căutării binare.

Conform proprietăților secvenței Farey, acest nou termen M este o fracție ireductibilă și între acest nou termen și limitele L și R nu mai există alte fracții cu numitor mai mic decât noua fracție calculată.

Procedăm astfel până când numitorul fracției mij depășește N.

În final este foarte important să verificăm care dintre fracțiile L, M sau R aproximează cel mai bine numărul x.

Algorithm 1: Algoritm pentru aproximarea cu fracții a unui număr real

intrare: $0 < x < 1,\ x \in \mathbb{R},\ N \in \mathbb{N}^*$

ieșire : $(p, q),\ 1 \leq p < q \leq N$, a.î. $\left|x - \frac{p}{q}\right| = minim$

```
L.x ← 0,  L.y ← 1
R.x ← 1,  R.y ← 1
while L.y ≤ N&&R.y ≤ N do
    M.x ← L.x + R.x
    M.y ← L.y + R.y
    if M.y > N then
        break
    end
    if M = x then
        return M
    else if M < x then
        L ← M
    else
        R ← M
    end
end
if x − L < R − x then
    return L
else
    return R
end
```

Pentru a obține o complexitate mai bună observăm că în loc să mergem de mai multe ori pe aceeași ramură de decizie cu ajutorul secvenței Farey, putem cauta binar numărul de pași necesari până vom schimba ramura.

Cum nu vom schimba ramura de decizie de mai mult de $\mathcal{O}(log(N))$ ori, deoarece la fiecare schimbare este garantat că am înjumătățit intervalul, complexitatea temporală a algoritmului este $O(log^2(N))$.

Algorithm 2: Algoritm optimizat pentru aproximarea cu fracții a unui număr real

intrare: $0 < x < 1,\ x \in \mathbb{R},\ N \in \mathbb{N}^*$
ieșire : $(p, q),\ 1 \leq p < q \leq N$, a.î. $\left|x - \frac{p}{q}\right| = minim$

$L.x \leftarrow 0,\ L.y \leftarrow 1$
$R.x \leftarrow 1,\ R.y \leftarrow 1$
while $L.y \leq N \&\& R.y \leq N$ **do**
  $M.x \leftarrow L.x + R.x$
  $M.y \leftarrow L.y + R.y$
  if $M.y > N$ **then**
    break
  end
  if $M = x$ **then**
    return M
  else if $M < x$ **then**
    Căutam binar P, a.î.:
      P este maxim;
      $L.y + R.y \cdot (P + 1) \leq N$;
      $\frac{L.x + R.x \cdot P}{L.y + R.y \cdot P} \leq x.$
    $L.x \leftarrow L.x + R.x \cdot (P + 1)$
    $L.y \leftarrow L.y + R.y \cdot (P + 1)$
  else
    Căutam binar P, a.î.:
      P este maxim;
      $R.y + L.y \cdot (P + 1) \leq N$;
      $\frac{R.x + L.x \cdot P}{R.y + L.y \cdot P} \leq x.$
    $R.x \leftarrow R.x + L.x \cdot P$
    $R.y \leftarrow R.y + L.y \cdot P$
  end
end
if $x - L < R - x$ **then**
  return L
else
  return R
end

Varianta 2 - ing. Mihail-Cosmin Piț-Rada

În continuare este prezentat un algoritm prin care se poate obține fracția $\frac{x}{y}$ care aproximează cel mai bine fracția $\frac{p}{q}, p \leq q$.

Vom presupune că avem respectate $a \cdot x + b \cdot y \leq N$ și $x \leq y$, iar p, q, a, b, N sunt date inițial.

Din teorema împărțirii cu rest avem că ($p = 0$ este un simplu caz particular):

$$q = c \cdot p + r, r < p, p > 0$$

$$y = d \cdot x + t, t < x, x > 0$$

Deoarece $\frac{1}{c+1} < \frac{p}{q} \leq \frac{1}{c}$ și $\frac{1}{d+1} < \frac{x}{y} \leq \frac{1}{d}$, este preferabil să avem $c = d$ pentru a avea $\frac{p}{q}$ și $\frac{x}{y}$ cât mai apropiate, în același interval.

Dar se poate să nu fie posibil acest lucru, caz în care este nevoie să analizăm $d > c$ sau $d < c$, în funcție de ce aproximare căutăm (superioară sau inferioară).

Cazul $d = c$. Vom rezolva numai inf(p, q, a, b, n, x, y), cazul sup(...) fiind similar.

De observat că avem:

$$a \cdot x + b \cdot y \leq N \iff a \cdot x + b \cdot (d \cdot x + t) \leq N \iff b \cdot t + (a + b \cdot c) \cdot x \leq N$$

.

Considerăm expresia: $f(s) = \frac{1}{c+s}$.

Rezultă: $\frac{p}{q} = \frac{p}{p \cdot c + r} = \frac{1}{-c} + \frac{r}{p} = f\left(\frac{r}{p}\right)$ și $\frac{x}{y} = \frac{x}{d \cdot x + t} = \frac{1}{d + \frac{t}{x}} = f\left(\frac{t}{x}\right)$.

Deoarece f este descrescătoare și $f\left(\frac{t}{x}\right)$ vrem să fie cea mai bună aproximare inferioară pentru $f\left(\frac{r}{p}\right)$, avem $f\left(\frac{t}{x}\right) \leq f\left(\frac{r}{p}\right)$.

Deasemenea din $a \cdot x + b \cdot y \leq N$, avem că $\frac{t}{x}$ trebuie să fie cea mai bună aproximare superioară pentru $\frac{r}{p}$ cu $\frac{t}{x} \geq \frac{r}{p}$, care respectă $b \cdot t + (a + b \cdot c) \cdot x \leq N$.

Altfel, dacă $\frac{t}{x}$ nu ar fi cea mai bună aproximare superioară ar trebui să existe $\frac{u}{v}$ mai bună astfel încât am avea $\frac{t}{x} > \frac{u}{v} \geq \frac{r}{p}$, și deoarece f este descrescătoare, am avea $f\left(\frac{t}{x}\right) < f\left(\frac{u}{v}\right) \leq f\left(\frac{r}{p}\right)$, de unde ar rezulta că $f\left(\frac{u}{v}\right)$ este o mai bună aproximare pentru $f\left(\frac{r}{p}\right)$, ceea ce ar fi o contradicție.

Deci $inf(p, q, a, b, n, x, y)$ se reduce la $sup(r, p, b, a + b \cdot c, t, x)$.

Cazul $d > c$ (aproximare inferioară) avem $\frac{1}{d+1} < \frac{x}{y} \leq \frac{1}{d} \leq \frac{1}{c+1} < \frac{p}{q} \leq \frac{1}{c}$

Maximizăm expresia $\frac{x}{d \cdot x + t}$ în condițiile $b \cdot t + (a + b \cdot d) \cdot x \leq N$ și $d \geq c + 1$

Din condiții rezultă că putem să descreștem $t \implies t = 0$.

Maximizăm $\frac{1}{d}$ (avem $x > 0$) în condițiile $(a + b \cdot d) \cdot x \leq N$.

Rezultă $d \geq c+1$ și $x = 1$ dacă $a + b \cdot (c+1) \leq N$ sau $x = 0$ în caz contrar.

Fracția $\frac{x}{y}$ este sau $\frac{1}{c+1}$ sau $\frac{0}{1}$ sau fără soluție.

Cazul $d < c$ (aproximare inferioară) $\implies \frac{1}{c+1} < \frac{p}{q} \leq \frac{1}{c} \leq \frac{1}{d+1} < \frac{x}{y} \leq \frac{1}{d}$.

Minimizăm $\frac{x}{d \cdot x + t}$ în condițiile $b \cdot t + (a + b \cdot d) \cdot x \leq N$ și $d \leq c - 1$ și rezultă că este sigur să descreștem $x \implies x = t - 1$.

Minimizăm $\frac{x}{(d+1) \cdot x - 1}$ în condițiile $(a + b + b \cdot d) \cdot x \leq N + b$ și $d \leq c - 1$ și rezultă că este important să avem d cât se poate de apropiat de $c - 1$ și rezultă că trebuie să încercăm cu $x = 1$ pentru a avea libertate în alegerea lui d, de aici rezultă $d \leq \frac{n-a}{b}$, apoi rezultă $d = min\left(c - 1, \frac{n-a}{b}\right)$, dacă $d \leq 0$ atunci nu avem soluție, rezultă $\frac{x}{(d+1) \cdot x - 1}$ este descrescătoare deci vrem x să fie cât mai mare posibil, rezultă $x = \frac{n+b}{a+b+b \cdot d}$, acum avem x și d restul este simplu.

Complexitatea temporală a soluției este $O(log(p+q))$

2 Problema Catmin

Propunător: prof. Piț-Rada Ionel-Vasile, Colegiul Național "Traian", Drobeta-Turnu Severin

Observații

Strategia de rezolvare presupune analizarea numerelor ca șiruri de cifre.

Să notăm cu M numărul total de cifre al celor N numere din șir. Algoritmul va avea exact M pași. La un pas k trebuie selectată cifra minimă dintre toate cifrele semnificative din șirul de numere. Pentru că este posibil ca cifra minimă să poată fi selectată de la mai multe numere, atunci trebuie făcută compararea lexicografică a lor.

Să luăm următorul exemplu simplu. Presupunem că șirul este format doar din două numere: 32 și 317. Dacă am alege la primul pas cifra 3 de la primul număr, atunci în continuare numărul minim posibil care s-ar obține ar fi 32317. Dar dacă se alege cifra din numărul minim lexicografic, adică 3 din al doilea număr, atunci șirul rămas va fi 32 și 17 și acum se obține soluția minimă posibilă, adică 31327.

Mai există însă un caz care poate să apară, anume când avem de ales între prima cifră minimă dintr-un șir precum 2, 223, 21. Compararea lexicografică înseamnă că s-ar alege numărul 2 (și implicit cifra 2 a sa) și nu am mai obține soluția minimă lexicografic.

Pentru a funcționa corect alegerea conform minimului lexicografic trebuie să completăm încă de la început cu cifre de 9 la finalul tuturor numerelor din șir astfel încât numerele să aibă aceeași lungime. Mai mult, atunci când se alege o cifră dintr-un număr, atunci se elimină acea cifră și se completează cu 9 la final. Deci dacă șirul este 2, 223, 21, atunci îl scriem 299, 223, 219. După ce se ia 2 din numărul 219 șirul devine 299, 223, 199.

Subtask 1 ($N \leq 500$) – 21 de puncte

Putem ca la fiecare pas din algoritm să căutăm numărul cel mai mic lexicografic, îi extragem prima cifră și o afișăm și îl reintroducem fără prima cifră.

Complexitatea temporală este $\mathcal{O}(N \cdot M)$.

Subtask 2 ($N \leq 10\ 000$) – 33 de puncte

Pentru a optimiza căutarea celui mai mic număr la fiecare pas, vom utiliza un min-heap. Astfel, complexitatea căutării, extragerii minimului și inserării unui număr va fi $O(log(N) \cdot \Sigma)$. După extragerea din min-heap a celui mai mic dintre numerele rămase vom extrage din număr prima cifră pe care o vom adăuga la șirul final apoi vom completa numărul cu o cifră 9, păstrând lungimea conform precizărilor anterioare, și apoi vom reintroduce numărul în min-heap.

Complexitatea temporală obținută este $O(N \cdot log(N) \cdot \Sigma)$, unde cu Σ am notat lungimea unui număr care se prelucrează.

Subtask 3 ($N \leq 100\ 000$) – 46 de puncte

Compararea lexicografică are complexitatea $\mathcal{O}(\Sigma)$. Pentru a reduce complexitatea comparării vom încerca să utilizăm compararea numerică $\mathcal{O}(1)$. După extragerea din min-heap a celui mai mic dintre numerele rămase vom extrage din număr prima cifră pe care o vom adăuga la șirul final apoi vom completa numărul cu o cifră 9, păstrând lungimea conform precizărilor anterioare, și apoi vom reintroduce numărul în min-heap.

Complexitatea temporală finală va fi $\mathcal{O}(N \cdot (\Sigma + log(N)))$.

3 Problema Umede

Propunător: prof. Nodea Gheorghe-Eugen, Colegiul Național ”Tudor Vladimirescu”, Târgu-Jiu

Observații preliminare

- o etajeră atinsă de apă poate inunda cel mult două etajere aflate pe un nivel inferior
- numărul maxim de etajere aflate pe un nivel este $< max(x)/2$

Soluție $\mathcal{O}(N^2)$

Implementarea necesită utilizarea vectorilor. Se parcurg etajerele aflate pe fiecare nivel y (de sus În jos), prin marcarea etajerelor atinse de apă de la un nivel superior. Se folosește un vector în care este marcat corespunzător: $up[x] = 1$ dacă de pe nivelul imediat superior curge apă, 0 altfel.

Soluție $\mathcal{O}(N \cdot log2(N))$

Varianta 1

Se sortează descrescător etajerele după y, iar pentru același y sortate după $x1$. Se folosește un vector în care se rețin pozițiile x care sunt *surse de apă*, inițial singura sursă de apă era la poziția X. Apoi prin interacțiunea dintre etajerele de apă de pe nivelul curent și sursele de apă de la nivelul anterior se obține un vector cu noile *surse de apă* de la nivelul curent, utilizând un algoritm asemănător cu cel de interclasare.

Varianta 2

Se sortează descrescător etajerele după y, la y egal crescător după x.

Vom folosi o coadă pentru a reține etajerele atinse de apă. Inițial adăugăm în coadă prima etajeră atinsă de apă plecând din punctul de coordonate (X, Y) al robinetului.

Căutăm după y, pentru fiecare extremitate a etajerei extrase din coadă, prima etajeră ce poate fi atinsă de apă aflată pe un nivel inferior raportat la nivelul (y-ul) etajerei extrase, iar dacă aceasta nu a fost umezită anterior, o adăugăm în coadă. Progresiv, marcăm etajerele umezite $umede[i] = 1$.

Punctul *b*) se poate rezolva prin smenul lui Mars (Difference Array) normalizat.

Echipa

Setul de probleme pentru această rundă a fost pregătit de:

- prof. Boian Dumitru Flavius, Colegiul Național "Spiru Haret", Târgu-Jiu
- prof. Bunget Mihai, Colegiul Naţional "Tudor Vladimirescu", Târgu-Jiu
- prof. Cheșcă Ciprian, Liceul Tehnologic "Grigore C. Moisil", Buzău
- prof. Costineanu Veronica Raluca, Colegiul Național "Ștefan cel Mare", Suceava
- prof. Dumitrascu Dan Octavian, Colegiul National Dinicu Golescu, Câmpulung
- prof. Iordaiche Cristina, Liceul Teoretic " Grigore Moisil" Timisoara, Timișoara
- prof. Lica Daniela, Centrul Județean de Excelență, Prahova
- prof. Nicoli Marius, Colegiul Național "Frații Buzești", Craiova
- prof. Nodea Gheorghe-Eugen, Colegiul Național "Tudor Vladimirescu", Târgu-Jiu
- prof. Piț-Rada Ionel-Vasile, Colegiul Național "Traian", Drobeta-Turnu Severin
- student Pop Ioan-Cristian, Universitatea Politehnica București, Facultatea de automatică și calculatoare.
- prof. Pracsiu Dan, Liceul Teoretic Emil Racoviță, Vaslui
- student Tulbă-Lecu Theodor-Gabriel, Universitatea Politehnica București, Facultatea de automatică și calculatoare.