TABĂRA DE PREGĂTIRE ȘI SELECȚIE A LOTULUI NAȚIONAL DE INFORMATICĂ MAI 2023, SLATINA

DESCRIEREA SOLUȚIILOR, BARAJ 3, JUNIORI

Ziua 3 — Problema 1: Teatru

Propusă de: Inst. Frâncu Cătălin, Clubul Nerdvana, București

Fie F_N al N-lea număr al lui Fibonacci. Observăm că actul N are lungime F_N. Precalculăm primele N numere Fibonacci.

Soluție în memorie F_{N+2}: Pentru primele două subtaskuri, putem pur și simplu crea prin concatenare toate actele de la 1 la N. Apoi tipărim subsecvența de la P la $P + L - 1$ din actul N. Această soluție necesită timp și memorie:

$$F_N + F_{N-1} + F_{N-2} + \cdots + F_2 + F_1 = F_{N+2} - 1$$

și se va încadra în 128 MB de memorie doar pentru $N \leq 38$. La limită, putem folosi și un singur bit pentru fiecare literă (având doar două litere distincte); astfel această versiune poate rezolva și subtaskul 3.

Soluție în memorie F_N: Observăm că actele $2, 3, \ldots, N-1$ sunt sufixe ale actului N. De aceea, putem folosi o singură zonă de memorie de F_N caractere în care construim, de la dreapta spre stânga, actele $2, 3, \ldots, N$. Astfel memoria necesară este doar de F_N octeți, suficientă până la $N \leq 40$.

Soluție în $\mathcal{O}(N \cdot L)$: $P \leq K < P + L$ și calculăm replica de pe fiecare poziție. Fie $C(K, N)$ replica de pe poziția K din actul N. Putem defini aceste valori prin cazurile de bază $C(1, 1) =$ 'A', $C(1, 2) =$ 'D' și recurența:

$$C(K, N) = \begin{cases} C(K, N-2) & \text{dacă } K \leq F_{n-2} \\ C(K - F_{N-2}, N-1) & \text{dacă } K > F_{n-2} \end{cases}$$

Diferența între subtaskurile 4 și 5 este că până la $F_{46} = 1.836.311.903$ putem opera pe `int`, iar mai departe este necesar tipul `long long`.

Soluție în $\mathcal{O}(N + L)$: Cât timp intervalul $[P, P + L)$ este complet inclus în actul $N - 2$ sau $N - 1$, reducem problema la acel act. Cînd P și $P + L$ se află respectiv în actele $N - 2$ și $N - 1$, reducem problema la a tipări un sufix al actului $N - 2$ și un prefix al actului $N - 1$.

Prefixul unui act N, dacă este mai scurt decât actul $N - 2$, se reduce la un prefix al acelui act. Altfel prefixul constă din actul $N - 2$ în totalitate și un prefix al actului $N - 1$. Similar, sufixul unui act N fie se reduce la un sufix al actului $N - 1$, fie la un sufix al actului $N - 2$ și actul $N - 1$ în totalitate. Cazul de bază, dacă coborâm suficient (eventual până la actul 1), este un act în totalitate (dacă sufixele și prefixele devin vide).

Întrucât la fiecare nivel al recurenței există cel mult un prefix și un sufix, vor exista cel mult N din fiecare. Pentru a tipări actele acoperite în întregime, le putem descompune naiv (recursiv) până la lungime 1, cu un efort total $\mathcal{O}(L)$. Pentru un plus de viteză, putem precalcula primele 10-20 de acte.

Altă soluție în $\mathcal{O}(N+L)$: Să inserăm (ipotetic) în fiecare act o bară verticală la locul concatenării. Acum, să analizăm replica 5 din actul 8 și să o regăsim în actele anterioare. Am notat-o cu X în loc de D, pentru ușurință.

```
Actul 8: DADAXDAD|ADDADDADADDAD
Actul 6: DAD|AXDAD
Actul 5: AX|DAD
Actul 3: A|X
Actul 2: X
```

Pornind din actul 8, am coborât de partea stângă a barei și am ajuns la actul 6. De acolo am coborât de partea dreaptă a barei până la actul 5, etc. Notăm asta ca: `8 s 6 d 5 s 3 d 2`. Tipărim A sau D după acum actul final este 1 sau 2. Iată traseele replicilor 5-9:

```
Replica 5: 8 s 6 d 5 s 3 d 2      --> D
Replica 6: 8 s 6 d 5 d 4 s 2      --> D
Replica 7: 8 s 6 d 5 d 4 d 3 s 1  --> A
Replica 8: 8 s 6 d 5 d 4 d 3 d 2  --> D
Replica 9: 8 d 7 s 5 s 3 s 1      --> A
```

Ca mod de construcție, observăm că pe stânga coborâm două niveluri, iar pe dreapta doar unul. Mai observăm că traseele a două replici consecutive diferă la ultimul moment unde prima replică coboară pe stânga, fix lângă bară, iar a doua coboară pe dreapta. Așadar, algoritmul construiește traseul primei replici ca pe o stivă cu actul cel mai mic la vârf. Apoi, de L ori,

- tipărește caracterul corect;
- coboară în stivă până la primul `s`;
- schimbă `s`-ul în `d`;
- urcă în stivă, pe stânga, din două în două niveluri, până la 1 sau 2.

La prima vedere, pare că algoritmul este $\mathcal{O}(NL)$. Totuși, să analizăm comportamentul stivei. Dacă la un moment dat ultimele direcții sunt `s d ... d`, după trecerea la replica următoare direcțiile vor fi `d s ... s`. Majoritatea trecerilor vor modifica doar ultimul element din stivă din `s` în `d`. Recunoaștem aici incrementarea unui contor binar, unde `s` = 0 și `d` = 1, care are cost amortizat (mediu) de două operații per incrementare. Așadar, complexitatea totală este $\mathcal{O}(N)$ pentru construcția stivei inițiale, apoi $\mathcal{O}(L)$ amortizat.

Soluție alternativă (Prof. Ciurea Stelian): O soluție frumoasă din punct de vedere matematic pornește de la observația că, pentru a k-a poziție numărând de la dreapta, replica este 'D' dacă și numai dacă

$$\lfloor (k+1)\phi \rfloor - \lfloor k\phi \rfloor = 2,$$

unde $\phi = (1+\sqrt{5})/2$ este numărul de aur. Din păcate, implementarea directă, chiar și cu `long double`, greșește ocazional din cauza erorilor de precizie. Ea poate fi dusă până la 100 de puncte cu gestionarea proprie a zecimalelor.

Ziua 3 — Problema 2: Culegeri

Propusă de: Stud. Pop Ioan Cristian, Universitatea Politehnică din București

În acest editorial vom analiza numai cerința $T = 2$, încât prima cerință este inclusă în aceasta.

Soluție în $\mathcal{O}(2^N \cdot N)$: Această soluție presupune generarea tuturor scenariilor posibile. În fiecare zi, avem două opțiuni: să creștem producția sau să printăm culegeri. Fiind N zile, vor fi $\mathcal{O}(2^N)$ scenarii, iar apoi în $\mathcal{O}(\mathcal{N})$ putem simula fiecare scenariu. Putem reduce complexitatea

la $\mathcal{O}(2^N)$ dacă menținem incremental stocul pe parcursul recursivității. Această soluție va obține în jur de 15 puncte.

Soluție în $\mathcal{O}(N^2)$: Putem construi următoarea dinamică:

$dp[i][j]$ = stocul maxim posibil de la finalul celei de-a i-a zi, dacă în j dintre zile am crescut capacitatea de producție.

Relația de recurență va fi:

$$dp[i][j] = max(dp[i-1][j] + (k+j), dp[i-1][j-1]) - c[i]$$

$dp[i-1][j] + (k+j)$ reprezintă soluția în care printăm culegeri. Plecăm de la o soluție cu capacitatea de producție $k+j$ (deci în j dintre zile am crescut producția) și printăm $k+j$ culegeri.

$dp[i-1][j-1]$ reprezintă soluția în care decidem să creștem capacitatea de producție, fără să modificăm stocul. Numărul de culegeri din stoc va fi egal cu cel de la ziua precedentă.

La final, va trebui să scădem $c[i]$ din această sumă, fiind obligați să livrăm $c[i]$ culegeri. Se poate obține o valoare negativă. În acest caz, este imposibil să obținem o soluție validă pentru perechea (i, j), și nu va fi folosită în crearea altor soluții.

Stocul maxim de la finalul celei de-a i-a zi va fi

$$S_i = \max(dp[i][0],\ dp[i][1],\ \ldots,\ dp[i][i])$$

O optimizare care se poate aduce acestei soluții este aceea că nu este necesară crearea unei matrici, ci doar păstrarea a două linii. Observăm ca linia $dp[i]$ este calculată strict pe baza valorilor de pe linia $dp[i-1]$, deci putem reduce memoria utilizată. O astfel de soluție va obține în jur de 50 de puncte.

Soluție în $\mathcal{O}(N)$: Una dintre observațiile de la care pleacă această soluție este aceea că investițiile (adică creșterea capacității de producție) trebuiesc făcute cât mai devreme posibil.

O altă observație este că nu toate investițiile pot fi profitabile. De exemplu, este inutil să investim în ultima zi. Pe caz general, dacă putem produce K culegeri, nu are sens să investim de la ziua $N-K$ încolo.

O ultimă observație este aceea că noi putem simula un scenariu în care decidem să investim la ziua i. Dacă stocul nu a ajuns niciodată negativ, atunci investiția este bună. Altfel, suntem obligați să renunțăm la ea (și să privim progresul fabricii ca și când această investiție nu s-a făcut niciodată).

Având în vedere cele de mai sus, pornim cu următoarea soluție: construim o stivă cu zilele în care investim. Presupunem că vom face orice investiție care are potențialul să fie profitabilă – deci la ziua i vom investi dacă este posibil și vom adăuga ziua i pe stivă.

Dacă stocul ajunge negativ, suntem obligați să renunțăm la investiții. Atunci, vom elimina investițiile una câte una de pe stivă și vom modifica stocul. Dacă ne aflăm la ziua i, și eliminăm o investiție făcută în ziua j, vom pierde $i-j$ zile în care am avut producția $k+1$, iar în ziua j vom produce k. Deci, vom câștiga $(i-j+1) \cdot k - (i-j) \cdot (k+1)$ culegeri, adică $k-(i-j)$.

Dacă o investiție s-a dovedit a fi profitabilă, nu are cum să fie scoasă de pe stivă - deoarece se garantează că există o soluție. Astfel, putem face acest proces zilnic (presupunem că putem investi și renunțăm la investiții dacă este nevoie), iar la final vom avea stocul maxim pentru ziua N.

Pentru a obține stocul maxim pentru celelalte zile, putem merge înapoi - de la ziua N la ziua 1. Știm pe baza soluției pentru ziua N că avem pe stivă singurele investiții care merită făcute (o soluție pentru ziua i nu va avea investiții diferite de soluția pentru ziua N). Așa că, pe măsură ce ne „întoarcem" în timp (iterăm cu i de la ziua $N-1$ spre ziua 1), renunțăm la investițiile care devin neprofitabile (în loc să avem ultima zi N, acum ultima zi este i - potențialul profit va scădea).

Această soluție va obține 100 de puncte. Alte implementări pot construi un vector cu investițiile făcute, iar apoi căuta ternar începând cu ce investiție să renunțe, obținând punctaje asemănătoare.

Ziua 3 — Problema 3: Countall

Propusă de: Stud. Bogdan Vlad Mihai, Facultatea de Matematică și Informatică, Universitatea București

Pentru a rezolva problema, vom împărți perechile de numere prime între ele în trei categorii:

- perechi formate din două numere care deja sunt fixate în șir;
- perechi formate din două numere care înlocuiesc două valori de 0;
- perechi formate cu un număr fixat și un număr care înlocuiește un 0.

O primă observație importantă este că fiecare pereche de poziții apare de același număr de ori în toate șirurile care se pot forma. Pentru simplitate, vom nota cu Z numărul de valori de 0 din șir și cu R numărul de valori fixate din șir.

Soluție în $\mathcal{O}(N^2 \cdot logM + M^2 \cdot logM)$: Vom calcula numărul de perechi de valori prime între ele, unde ambele numere sunt fixate în șir. Acest număr se poate calcula ușor dacă fixăm fiecare două numere din șir și facem cel mai mare divizor comun al lor. Vom nota acest număr cu $fixedPairs$. Aceste perechi vor apărea în fiecare șir care se poate obține, deci, trebuie să înmulțim $fixedPairs$ cu M^Z.

Notăm cu $freePairs$ numărul de perechi de numere prime între ele, unde ambele numere înlocuiesc două valori de zero din șir. Acest număr se poate calcula ușor în $O(M^2 \cdot logM)$. Aceste perechi contribuie la toate șirurile care au $Z-2$ valori de zero și restul de valori fixate. Trebuie să ținem cont și de pozițiile pe care așezăm numerele din pereche. Prin urmare, $freePairs$ trebuie înmulțit cu $Z \cdot (Z-1)/2 \cdot M^{Z-2}$.

Notăm cu *combinedPairs* numărul de perechi formate dintr-un număr fixat și o valoare de zero din șir. Acestea apar în toate șirurile cu $Z-1$ valori de zero, deci, acest număr trebuie înmulțit cu $M^{Z-1} \cdot Z$.

Soluție în $\mathcal{O}(M \cdot 2^F \cdot F + M \cdot logM)$: Pentru soluția completă, este nevoie să calculăm valorile $fixedPairs$, $freePairs$ și *combinedPairs* mai rapid.

Pentru început, vom calcula phi_i ca fiind numărul de valori mai mici sau egale decât i care sunt prime cu i. Acest șir se poate calcula folosind un ciur în $O(M \cdot logM)$. Acum, $freePairs$ este egal cu $1 + \sum_{i=2}^{M} 2 \cdot phi_i$.

Pentru a calcula $fixedPairs$ și *combinedPairs* vom folosi principiul includerii și al excluderii. Ne vom concentra pe calcularea $fixedPairs$, de vreme ce *combinedPairs* se calculează într-un mod asemănător.

Pentru un număr fixat x, vom reține o listă cu numerele prime distincte care apar în descompunerea sa în factori primi. Observăm că dimensiunea aceste liste este foarte mică; mai exact, ea poate să aibă cel mult 7 elemente. Vom încerca să calculăm pentru fiecare număr din șir, numărul de numere din stânga sa cu care **nu** este prim (și vom nota acest număr cu *badPairs*).

Așadar, pentru fiecare submulțime nevidă a listei de factori primi distincți, va trebui să adăugăm, respectiv să scădem din *badPairs* (în funcție de paritatea cardinalului submulțimii) fr_{subset}, unde fr_{subset} este numărul de numere din șir din stânga numărului pe care îl procesăm care conțin numerele din *subset* în descompunerea lor în factori primi. Conform principiului includerii și al excluderii, trebuie să scădem fr_{subset} dacă subset are dimensiune pară, respectiv să adăugăm fr_{subset} dacă subset are dimensiune impară. La final, trebuie să adăugăm $i - badPairs$ la $fixedPairs$.

Detaliile în ceea ce privește calcularea răspunsului (după calcularea $fixedPairs$, *combinedPairs* și $freePairs$) sunt aceleași cu cele de la subtask-ul anterior.

Complexitatea finală este $\mathcal{O}(M \cdot 2^F \cdot F + M \cdot logM)$.

Echipa

Problemele pentru această etapă au fost pregătite de:

- Prof. Panaete Adrian, Colegiul Național „A.T. Laurian”, Botoșani
- Prof. Șerban Marinel, Colegiul Național „Emil Racoviță”, Iași

- Prof. Chește Ciprian, Liceu Tehnologic „Grigore C. Moisil”, Buzău
- Prof. Nodea Gheorghe Eugen, Colegiul Național „Tudor Vladimirescu”, Târgu Jiu
- Prof. Pițt-Rada Ionel-Vasile, Colegiul Național „Traian”, Drobeta-Turnu Severin
- Prof. Lica Daniela, Centrul Județean de Excelență Prahova
- Prof. Costineanu Raluca, Colegiul Național „Ștefan cel Mare”, Suceava
- Prof. Cerchez Emanuela, Colegiul Național „Emil Racoviță”, Iași
- Prof. Bunget Mihai, Colegiul Național „Tudor Vladimirescu”, Târgu Jiu
- Prof. Boian Flavius, Colegiul Național „Spiru Haret”, Târgu Jiu
- Prof. Szabo Zoltan, Inspectoratul Școlar Mureș
- Inst. Frâncu Cătălin, Clubul Nerdvana, București
- Stud. Tulbă-Lecu Theodor Gabriel, Universitatea Politehnică din București
- Stud. Bogdan Vlad Mihai, Facultatea de Matematică și Informatică, Universitatea București
- Stud. Pop Ioan Cristian, Universitatea Politehnică din București
- Stud. Onuț Andrei, Universitatea Yale, S.U.A.

Cursurile de pregătire au fost susținute de:

- Prof. Marius Nicoli, Colegiul Național „Frații Buzești”, Craiova
- Prof. Lica Daniela, Centrul Județean de Excelență Prahova
- Inst. Frâncu Cristian, Clubul Nerdvana, București
- Stud. Popa Bogdan Ioan, Facultatea de Matematică și Informatică, Universitatea București
- Stud. Dăscălescu Ștefan-Cosmin, Facultatea de Matematică și Informatică, Universitatea București

Comisia tehnică a fost formată din:

- Prof. Opriță Petru Simion, Liceul „Regina Maria”, Dorohoi
- Prof. Bolohan Mihai, Liceul „Regina Maria”, Dorohoi