Descrierea Soluțiilor
Olimpiada Societății pentru Excelență și Performanță în Informatică, Etapa Națională
Clasa a IX-a

1 Problema Le Mans

Propunător: Tulbă-Lecu Theodor-Gabriel, Universitatea Politehnica din București

Soluție pentru cazul $M = 1$ – 8 puncte

În cazul în care $M = 1$, atunci problema se reduce la a găsi diferența maximă dintre vitezele a oricăror 2 mașini, astfel răspunsul va fi $diff_{max} \cdot T$.

Soluie pentru cazul $M = 2$ – 9 puncte

Pentru M = 2, avem în vedere cea mai lentă M_l și cea mai rapidă M_r mașină, și respectiv cea mai din spate P_1 și cea mai din față P_2 poziție de start.

Distingem 2 cazuri:

1. M_l și M_r încep de pe aceeași poziție, caz în care putem să alegem toate mașinile de pe acea poziție.
2. M_l începe de pe poziția P_2 și M_r începe de pe poziția P_1. Atunci, pentru celelalte mașini alegem pozițiile astfel încat distanța parcursă plecând de la acea poziție să fie cât mai aproape de intervalul dat de distanțele parcurse de mașinile M_l și M_r.

Soluție pentru cazul $M, N \leq 7$ – 10 puncte

Pentru N și M suficient de mici, putem implementa un algoritm de tip backtracking pentru a încerca toate modurile de așeza mașinile pe grila de start. **Complexitate temporală:** $\mathcal{O}(M^N)$

Soluție pentru cazul $N, M \leq 100$ – 23 de puncte

Putem să fixăm o mașină pe o poziție și impunem ca aceasta să fie mașina care va parcurge cea mai mică distantă, pe care o vom nota cu D_{min}. Acum putem parcurge toate celelalte mașini și vom alege pentru fiecare mașină cea mai mică poziție de start astfel încât aceasta să termine înaintea ultimei mașini.

Putem implementa acest lucru sortând pozițiile de start și cautând binar pentru fiecare mașină i, poziția $p[x]$ minimă astfel încât $v[i] * T + p[x] \geq D_{min}$.

Complexitate temporală: $\mathcal{O}(N \cdot M \cdot M \cdot log(M))$

O altă implementare a acestei soluții este sa sortăm atât mașinile cât și pozițiile de start și să găsim soluțiile folosind doi pointeri, unul pentru mașini si unul pentru poziții. Putem face acest lucru deoarece o mașină mai rapidă va avea o poziție de start mai mică sau egală cu mașina precedentă în vectorul sortat.

Complexitate temporală: $\mathcal{O}(N \cdot M \cdot (N + M))$

Soluție pentru cazul $v[i] \leq 10^3, p[i] \leq 10^6$ – 19 puncte

Pentru a obține o soluție mai bună vom face următoarea construcție:

Întrucât valoarea maximă a distanței parcurse de o mașină, D_{max}, este mai mică decât $2 \cdot 10^6$, putem construi un vector de apariții, sa îl numim *pos*. Vom adăuga pentru fiecare pereche mașină i și poziție de start j – pe $pos[v[i] \cdot T + p[j]]$, perechea (i, j).

Soluția optimă este intervalul de lungime minima $[a, b]$ astfel încât, în cadrul tuturor perechilor din vectorul de apariție *pos* cuprinse între a și b, fiecare mașină să apară cel puțin o singură dată.

Acest lucru se poate realiza ușor folosind doi pointeri pentru capetele intervalului și un vector de frecvență pentru a ști de câte ori apare o mașină în interval.

Complexitate temporală: $\mathcal{O}(N \cdot M + D_{max})$

Soluție oficială – 100 de puncte

Vom proceda la fel ca la soluția anterioară, dar de data aceasta vom înlocui vectorul de frecvență cu un vector de tuple $(v[i] \cdot T + p[j], i, j)$, pe care îl vom sorta crescător după distanța parcursă. Astfel **complexitatea temporală** finală este: $\mathcal{O}(N \cdot M \cdot log(M \cdot N))$

2 Problema Oposumi

Propunător: Mihaela Cismaru

Soluție cerința 1

Se parcurge matricea triunghiulară în următorul fel: se parcurge coloana 1 de sus in jos, se parcurge coloana 2 de sus în jos... se parcurge coloana N de sus în jos. La fiecare pas hotărâm care este poziția cea mai înaltă : poziția curentă sau prima poziție de pe următoarea coloană și notăm înălțimea maximă a celor două ca fiind cel mai de sus loc unde se poate situa oposumul ce urmează să fie așezat.

Soluție cerința 2

Se așază elementele de la 1 la K-1 astfel: se așază primii N oposumi pe coloana 1 de sus în jos, următorii N-1 oposumi pe coloana 2 de sus în jos... și asa mai departe. Când am terminat de pus cei k-1 oposumi, punem oposumul rămas pe cea mai de sus poziție liberă, aceasta fiind fie pozitia ce urmează, fie prima poziție de pe următoarea coloană. După ce am așezat oposumul k, continuăm să așezăm în același mod oposumii rămași.

3 Problema ELHC

Propunători: Ștefania Ionescu, University of Zurich,
Alexandru Petrescu, University of Oxford (Keble College),
Vlad Gavrilă, University of Cambridge (Churchill College)
Autori editorial: Vlad Gavrilă, Ștefan Manolache

Soluție parțială – 7 puncte, respectiv 21 de puncte

Prima observație necesară este că o particulă G lansată la nivelul de energie k într-un tunel de circumferință P va activa senzorul G^k modulo P. Astfel, prima abordare este să trecem prin puterile lui G modulo P și să marcăm într-un vector de apariție care dintre senzorii $1, 2, ...P-1$ au fost activați. Putem face asta calculând fiecare putere individual, liniar, în $O(T * P^2)$, pentru a rezolva problema pentru 7 puncte, sau să calculăm în $O(T * P)$ toate puterile, pentru a rezolva problema pentru 21 de puncte.

Soluție parțială – 74 de puncte

A doua observație importantă este că valorile senzorilor pe care îi activăm sunt periodice. Daca $G^k \equiv 1$ (modulo P), atunci $G^{k+1} \equiv G$, $G^{k+2} \equiv G^2$, etc. Așadar, odată ce activăm senzorul 1, incrementând nivelul k de energie, suntem siguri că am activat toți senzorii pe care puteam să-i activăm.

Mai mult, avem garanția că $G^{P-1} \equiv 1$ modulo P, folosind Mica Teoremă a lui Fermat. Astfel, dacă G^{P-1} este prima putere a lui G egală cu 1 modulo P, înseamnă că până atunci nu am întâlnit nicio repetiție, deci toate valorile $1, 2, ..., P-1$ sunt atinse de catre $G, G^2, G^3, ..., G^{P-1}$ (nu neapărat în ordinea aceasta). În schimb, dacă există o putere a lui G mai mică, egală cu 1 modulo P, atunci nu este posibil să atingem toate numerele dorite, din cauza periodicității.

Așadar, problema noastră se reduce la: un experiment cu o particulă G într-un tunel de lungime P are date complete $\iff$ G^{P-1} este cea mai mică putere a lui G egală cu 1 modulo P.

A treia observație de care avem nevoie este că, dacă k este cel mai mic exponent pentru care $G^k \equiv 1$ (modulo P), atunci $k | P-1$. Această afirmație se poate demostra ușor prin reducere la absurd: dacă k este cel mai mic exponent pentru care $G^k \equiv 1$ (modulo P), dar $k \nmid P-1$, atunci restul r al împărțirii lui $P-1$ la k ar fi un exponent mai mic decât k pentru care $G^r \equiv 1$ modulo P, fapt ce reiese din periodicitatea valorilor puterilor lui G modulo P.

Așadar, tot ce trebuie să facem e să verificăm, pentru toți divizorii d lui $P-1$, dacă $G^d \not\equiv 1$ (modulo P). Putem afla divizorii lui P în complexitate $O\left(\sqrt{P}\right)$, iar dacă folosim un algoritm de exponențiere rapidă, putem obține puterile lui G corespunzătoare divizorilor lui $P-1$ în $O(\log P * D(P-1))$, (unde $D(n)$ este numărul divizorilor lui n, care în medie este $logN$ și niciodată nu este mai mare de $2 * \sqrt{n}$). Astfel, complexitatea finală a soluției pentru 74 de puncte este $O\left(T * \left(\sqrt{P} + \log P * D(P-1)\right)\right)$

Soluție completă

Pornind de la soluția precedentă, observăm că este suficient să verificăm doar puterile $\frac{P-1}{p_1} = K_1$, $\frac{P-1}{p_2} = K_2$, ... $\frac{P-1}{p_x} = K_x$, unde $p_1, p_2, ..., p_x$ sunt factorii primi ai lui $P - 1$. Este suficient să verificăm doar aceste puteri pentru că orice divizor d al lui $P-1$ mai mic decât $P-1$ este garantat să fie, de asemenea, un divizor al unuia dintre aceste numere. Astfel, dacă există k, cel mai mic divizor pentru care $G^k \equiv 1$ modulo P, atunci din periodicitate vom obține și că pentru orice multiplu de k, $m = nk$, $G^m \equiv 1$ modulo P. Cum cel puțin unul dintre numerele $K_1, K_2 ... K_x$ este un multiplu al lui k, obținem faptul că pentru a confirma existența lui k este suficient să verificăm doar dacă există un y pentru care $G^{K_y} \equiv 1$ modulo P. Deoarece numărul de factori primi al unui număr crește asimptotic cu $O\left(\frac{\log P}{\log \log P}\right)$, obținem complexitatea finală $O\left(T * \left(\sqrt{P} + \frac{\log^2 P}{\log \log P}\right)\right)$ (păstrând același algoritm de factorizare a lui P, dar și exponențierea rapidă pentru obținerea puterilor lui K din soluția precedentă).

Altă optimizare, nenecesară pentru obținerea punctajului maxim, este să folosim Ciurul lui Eratostene pentru a precalcula toate numerele prime mai mici ca valoarea maximă a lui P, pentru a optimiza obținerea divizorilor fiecărui P din fișierul de intrare.

*Echipa
Setul de probleme pentru această rundă a fost pregătit de:

- student Andrei Arhire, Universitatea ”Alexandru Ioan Cuza” din Iași, Facultatea de Informatică.
- profesor Alin Burța, Colegiul Național ”B.P. Hasdeu”, Buzău.
- studentă Mihaela Cișmaru, Universitatea Politehnica București, Facultatea de automatică și calculatoare.
- software engineer Vlad Gavrilă, Google, Zurich.
- student Ștefan Manolache, University of Oxford, Lady Margaret Hall College.
- profesor Adrian Panaete, Colegiul Național ”A.T.Laurian”, Botoșani.
- student Alexandru Petrescu, University of Oxford, Keble College.
- student Bogdan-Ioan Popa, Universitatea din București, Facultatea de Matematică și Informatică.
- profesor Zoltan Szabo, Liceul Tehnologic ”Petru Maior” Reghin / ISJ Mureș - Tg.Mureș.
- student Cezar Trișcă-Vicol, University of Oxford, Christ Church College.
- student Theodor-Gabriel Tulbă-Lecu, Universitatea Politehnica București, Facultatea de automatică și calculatoare.