Clasa a VII-a și a VIII-a

DESCRIERE SOLUȚII

PROBLEMA 1 TASK

Autori: studenti Diana MICLOIU și Vlad NITU
DELFT UNIVERITY OF TECHNOLOGY- COMPUTER SCIENCE AND ENGINEERING

Soluția optimă este să punem toate timpurile inițiale într-o coadă. Pentru fiecare timp rămas (de la m+1 la n) în vectorul task[i], vedem cărui prieten trebuie să i-l atribuim. Vom ține minte într-o variabilă cel mai mic timp curent al unui prieten și îl eliminăm din coadă. Următorul task este preluat de prietenul care termină primul, adică cel care a avut cel mai scurt task de rezolvat. Soluția curentă sol va fi actualizată, iar timpul nou obținut va fi adăugat în coadă. La sfârșit, se va afișa soluția sol.

Soluție elev Ionescu Maria - Colegiul Național "Mihai Viteazul", Ploieşti

Pentru început, dacă avem mai mulți prieteni (notăm cu m) decât taskuri (notăm cu n), pur și simplu vom afișa cea mai mare durată de timp în care un prieten face un task. Prietenul se află între primii n care sunt în șir deoarece toți prietenii care sunt pe pozițiile de la n+1 la m nu vor mai avea ce taskuri să facă.
Pentru cazul în care avem mai multe taskuri decât prieteni, vom reține duratele de timp ale prietenilor într-un vector pe care îl sortăm.
O idee ar fi ca, pe urmă, să adăugăm de fiecare dată următorul task la primul element din vector (prietenul care rămâne fără task de făcut) și să sortăm de fiecare data.
O idee mai eficientă este să folosim căutare binară pentru a găsi poziția din șir în care s-ar încadra elementul de pe prima poziție (notăm cu k poziția din șir găsită). Permutăm elementele de la 2 la k cu o poziție la stânga și inserăm elementul nou pe poziția k. Se repetă procedeul până la terminarea taskurilor și se afișează ultimul element din șir.

PROBLEMA 2 DEFENSE

Autor: elev Erik POPESCU
Colegiul Național "Mihai Viteazul", Ploieşti

O soluție este ca, pentru fiecare scut, să se ia la rând toate celulele pe care acesta **le** acoperă și să se adauge valoarea ***def*** corespunzătoare scutului. Această soluție nu va obține punctajul integral deoarece va fi depășită limita de timp pentru o parte din teste.
Pentru punctajul integral se folosește Șmenul lui Mars pentru matrice.
În acest sens se folosește o matrice auxiliară A (care va avea cu o linie și o coloană mai mult decât cea inițială). Această matrice trebuie să aibă inițial toate valorile egale cu 0. Pentru fiecare scut pentru care cunoaștem cele 5 valori *i*1, *j*1, *i*2, *j*2 și *def*, facem următoarele modificări în matricea auxiliară **A:**

A[i1][j1]=A[i1][j1]+def
A[i2][j2]=A[i2][j2]+def
A[i1][j2+1]=A[i1][j1+1]-def
A[i2+1][j1]=A[i2+1][j1]-def

După parcurgerea tuturor scuturilor se calculează elementele matricei B cu relația de forma:

A[i][j] = A[i][j] + A[i-1][j] + A[i][j-1] - A[i-1][j-1]

Ploiești, 28 mai 2022

Valorile finale ale defense-ului pentru fiecare celulă vor fi: D[i][j] + A[i][j]. Pe măsură ce se calculează aceste valori se caută și maximul și numărul de apariții al acestuia.

Pentru situația în care sunt doi jucători, sunt necesare niște modificări pentru mutările lui Programel (mutările cu număr par). Astfel, o celulă indicată prin coordonatele i, j pentru mutarea lui Programel, va fi echivalentul coordonatelor $n - i + 1, m - j + 1$ din perspectiva lui Algorel.
Având în vedere schimbarea de unghi, coordonatele colțurilor trebuie inversate. Colțul din stânga – sus din unghiul lui Programel, devine colțul din dreapta-jos pentru Algorel. La fel și pentru celălalt colț al scutului.

PROBLEMA 3 PETRECERE

Autor: elev Erik POPESCU
Colegiul Național "Mihai Viteazul", Ploiești

Vom face mai întâi niște observații privind a doua cerință. Ceea ce avem de făcut este să plasăm cei *m* copii celor *k* adulți. Pentru aceasta folosim următoarea teoremă:

Teoremă. Numărul de variante de a plasa *n* obiecte identice în *k* cutii este egal cu C_{n+k-1}^{k-1}.

Deoarece în problemă contează strict numărul de copii pe care îl are fiecare adult, putem considera cei *m* copii ca fiind "obiectele identice de plasat" și cei *k* adulți ca fiind "cutiile". Astfel, pentru a plasa cei *m* copii către cei *k* adulți, folosind formula de mai sus și schimbând notațiile cu cele din problemă, obținem că există un număr de variante egal cu C_{m+k-1}^{k-1}.

Observăm astfel că avem de calculat aproape același lucru ca și la cerința 1. Diferența la care trebuie să fim atenți este că la cerința 2 cele două valori de intrare se dau în altă ordine decât la cerința 1. Din acest motiv, dacă cerința este 2, vom citi variabilele în ordinea cealaltă (adică citim b și apoi a). Mai departe, indiferent de cerință, vom avea de calculat același lucru.

Chiar dacă teorema nu este cunoscută, formula poate fi "ghicită". Din modul în care este formulat enunțul problemei se poate recunoaște faptul că cele două cerințe au legătură între ele și că pentru a doua cerință sunt necesare doar "mici modificări" față de prima. Astfel se poate considera că și la a doua cerință vom avea de calculat combinări asemănătoare cu cele de la primul subpunct. Mai departe, "ghicirea" formulei exacte se poate face prin încercări pornind de la exemplele 3 și 4, precum și alte încercări care se mai pot face pe hartie cu numere mici.

Astfel, indiferent de cerință, rezolvarea problemei se reduce la a calcula C_n^k.

Soluții intermediare

- Dacă $n \leq 20$, cum 20!= 2.432.902.008.176.640.000, se pot calcula **n!, k! și (n-k)!** folosind **long long int**. Se fac apoi împărțirile necesare și se obține rezultatul.
- Pentru valori intermediare ale lui *n* , se pot descompune în factori primi numerele **n!, k! și (n-k)!,** se fac împărțirile ținând cont de faptul că $\frac{a^x}{a^y} = a^{x-y}$ și se calculează apoi rezultatul modulo 1.000.000.007

Ploiești, 28 mai 2022

Soluție de 100 de puncte

Se calculează $\frac{1}{k!}$ și $\frac{1}{(n-k)!}$ modulo 1.000.000.007 folosind inversul modular, adică $\left(\frac{1}{k!}\right)\%p = (k!)^{p-2}\ \%\ p$, $\left(\frac{1}{(n-k)!}\right)\%p = ((n-k)!)^{p-2}\ \%\ p$, unde p = 1.000.000.007.

Evităm astfel împărțirile și vom putea calcula C_n^k folosind doar înmulțiri modulo p, cu relația:

$$C_n^k = (n! * (k!)^{p-2} * ((n-k)!)^{p-2})\%p$$

Ridicările la putere se vor realiza folosind exponențierea rapidă.

Mai putem observa și că $\frac{n!}{k!*(n-k)!}$ se poate simplifica fie prin $k!$ fie prin (n-k)! Putem simplifica prin maximul din cele două valori, la numitor rămânând doar factorialul minimului dintre cele două). Obținem $\frac{n!}{k!*(n-k)!} = \frac{(maxim+1)*(maxim+2)*_{...}*n}{(minim)!}$ și putem folosi aici inversul modular pentru $\frac{1}{(minim)!}$

PROBLEMA 4 DRONE

Autor:STUDENT Andrei DRAGAN
UNIVERSITATEA BABES-BOLYAI

Mai întâi se citesc toate pozițiile dronelor și se validează așezarea acestora. Deși planșa de joc din final va conține numai caracterele “-“, “o“, “x“ și “X“, până la afișarea finală, pentru a reține fiecare dronă încă nedescoperită, se va modifica planșa de joc astfel: prin “C“ se va reprezenta cabina dronei, iar prin “*“, celelalte părți ale dronei.

Dacă jocul nu este terminat, se va lua în calcul fiecare lovitură și fie se va afișa mesajul corespunzător (în cazul în care atacul nu este valid), fie se va modifica conținutul celulei respective așa cum prezintă cerință. Atenție – în cazul în care se doboară o celulă ce are conținutul “C”, atunci toate celulele ce alcătuiesc drona respectivă își vor schimba conținutul în “X”.

În final, se va afișa planșa de joc, în funcție de atacurile efectuate. Înainte de afișare se va lua în considerare ca dronele încă nedescoperite, adică celulele ce au conținutul fie “C”, fie “*”, să rămână nedescoperite – anume conținutul celulei respective va fi modificat din nou în “-“.