

Problema 1 <pif>

Descrierea unor soluții posibile

Structurile de date utilizate: vectori

Gradul de dificultate: 2

Soluția 1 – 30 p (sursa: pif_30p_1.cpp)

Utilizăm o funcție recursivă ce este apelată pentru fiecare persoană ce intră în joc. Această funcție se autoapelează de k ori, câte un apel pentru fiecare nouă persoană adusă în joc de persoana curentă și transmite prin parametri ziua în care noua persoană a intrat în joc și tipul de persoană (tânăr sau vârstnic).

Ieșirea din recursivitate se realizează atunci când ziua depășește valoarea lui n .

Pentru a obține numărul maxim de fapte bune trebuie ca în joc să intre cât mai multe persoane. Acest lucru se obține dacă se alege întotdeauna mai întâi tipul de persoane care au cel mai mic număr de zile pentru a realiza o faptă bună. Dacă $z_v < z_t$ se vor alege mai întâi vârstnici. Dacă $z_t < z_v$ se vor alege mai întâi tineri.

Tot pentru a obține numărul maxim de fapte bune, dacă valoarea lui k este impară se vor alege $\frac{k+1}{2}$ persoane din tipul cu număr mai mic de zile pentru a realiza o faptă, iar apoi $\frac{k-1}{2}$ persoane din celălalt tip. Dacă valoarea lui k este pară se vor alege $\frac{k}{2}$ persoane din ambele tipuri.

Funcția returnează numărul de fapte bune rămase nerealizate.

Complexitate: $O\left(\frac{n}{k^{zf+zb}}\right)$

Soluția 2 – 70 p (sursa: pif_70p_1.cpp)

Pentru a evita apelurile redundante ale funcției utilizăm doi vectori în care memorăm valorile returnate de funcție pe măsură ce apelurile recursive se desfășoară. Un vector pentru vârstnici și unul pentru tineri. Înainte de fiecare apel verificăm dacă funcția a fost deja apelată pentru valorile respective (ziua și tipul de persoană), dacă a fost deja apelată luăm valoarea din vectori, dacă nu a fost apelată o apelăm.

Complexitate: $O(k \cdot n)$

Soluția 3 – 30 p (sursa: pif_30p_2.cpp)

Pentru cazul $z_t = z_v = 1$ numărul de fapte bune este același indiferent de ordinea în care ar fi alese persoanele vârstnice și tinere, iar numărul de persoane noi care intră în joc în fiecare zi este dat de un șir Fibonacci generalizat în care fiecare valoare se obține prin însumarea ultimelor k valori anterioare: $x_n = \sum_{i=n-k}^{n-1} x_i$.

Folosim un vector în care memorăm aceste valori.

Parcurgem vectorul și calculăm numărul de persoane din ziua curentă pe baza valorilor din zilele anterioare.

La sfârșit calculăm numărul de fapte bune rămase nerealizate de către cei care au intrat în joc în ultimele k zile.

Complexitate: $O(n)$

Soluția 4 – 50 p (sursa: pif_50p.cpp)

În general pentru cazul în care $z_t = z_v$ procedăm la fel ca în cazul anterior doar că în loc să parcurgem vectorul cu pasul 1, vom parcurge vectorul cu pasul z_v . Nici pentru această situație nu contează ordinea în care alegem persoanele vârstnice și pe cele tinere.

Complexitate: $O\left(\frac{n}{z_v}\right)$

Soluția 5 – 70 p (sursa: pif_70p_2.cpp)

Pentru situațiile în care $z_t \neq z_v$ procedăm asemănător, dar păstrăm numărul de tineri și vârstnici care intră în joc în fiecare zi, în vectori separați, pentru că durează un număr de zile diferit până când finalizează realizarea celor k fapte bune. Fie acestea n_v pentru vârstnici și n_t pentru tineri. Parcurgem vectorii în paralel și pentru fiecare poziție i , adunăm numărul de persoane de pe poziția i la numărul de persoane de pe pozițiile $i+z_v, i+2z_v, \dots$ pentru vârstnici și $i+z_t, i+2z_t, \dots$ pentru tineri.

La adunare trebuie ținut seama de faptul că fiecare persoană trebuie să aleagă întotdeauna mai întâi $\frac{k+1}{2}$ persoane din tipul de persoane care au cel mai mic număr de zile pentru a realiza o faptă bună și apoi $\frac{k}{2}$ persoane din tipul celălalt.

Dacă $z_v < z_t$ vom aduna în vectorul nv pe pozițiile $i+z_v, i+2z_v, \dots, i + \frac{k+1}{2} z_v$ valorile din $nv[i]$ și pe pozițiile $i+z_t, i+2z_t, \dots, i + \frac{k+1}{2} z_t$ valorile din $nt[i]$. În vectorul nt vom aduna pe pozițiile $i + \left(\frac{k+1}{2} + 1\right) z_v, i + \left(\frac{k+1}{2} + 2\right) z_v, \dots, k z_t$ valorile din $nv[i]$ și pe pozițiile $i + \left(\frac{k+1}{2} + 1\right) z_t, i + \left(\frac{k+1}{2} + 2\right) z_t, \dots, k z_t$ valorile din $nt[i]$

Dacă $z_t < z_v$ vom proceda analog inversând vectorii nv și nt .

Toate valorile care ar trebui adunate pe poziții mai mari de cât n în cei doi vectori reprezintă fapte bune ce mai sunt de realizat după trecerea celor n zile.

Complexitate $O(k \cdot n)$

Soluția 7 – 90 p (sursa pif_90p.cpp)

La soluția anterioară, pentru a elimina redundanța adunărilor la fiecare pas i , folosim câte un vector suplimentar pentru fiecare tip de persoană (unul pentru tineri și unul pentru vârstnici), în care memorăm numărul de persoane active în fiecare zi (adică numărul de persoane care mai au de realizat fapte bune). Astfel numărul de persoane noi care intră în joc în ziua i va depinde doar de numărul de persoane active din ziua $i-z_v$, respectiv $i-z_t$. La fiecare pas actualizăm numărul de persoane active, ținând seama de modul cum sunt alese tipurile de persoane.

Complexitate $O(n)$

Soluția 8 – 70 p (sursa: pif_70p_3.cpp)

Să ne propunem să calculăm în fiecare zi câte persoane intră în joc.

Fie $v[i]$ și $t[i]$, numărul vârstnicilor respectiv al tinerilor care intră în joc în ziua i . Evident $v[0] = 0$ și $t[0] = 1$.

Cum putem afla, la finalul fiecărei zile, câte fapte bune mai rămân de efectuat? Pornim cu o variabilă target egală cu k , fiind sarcina lui Trevor, pe care o vom adapta din aproape în aproape pentru a reflecta numărul de fapte bune ce trebuie efectuate la finalul zilei i .

Astfel, la finalul zilei i , cele $v[i] + t[i]$ persoane noi introduse trebuie scăzute din target, întrucât ele au fost introduse de câte o persoană ce acum are o sarcină cu 1 mai mică. De asemenea, cele $v[i] + t[i]$ persoane noi introduse, au la rândul lor de adus câte k persoane în joc fiecare. Reiese deci că variabila target va crește cu

$(v[i] + t[i]) \cdot (k - 1)$.

Să ne concentrăm acum atenția asupra calculului valorilor $v[i]$ și $t[i]$.

Putem presupune $z_t < z_v$, celălalt caz fiind similar. Este destul de intuitiv faptul că dacă o persoană aduce întâi în joc tineri, cât de mult se poate, și apoi vârstnici, vor fi mai multe persoane și în consecință mai multe fapte bune. Acest lucru se întâmplă deoarece tinerii au o frecvență de lucru mai bună, $z_t < z_v$, și cu cât sunt introduși mai devreme în joc, la rândul lor vor introduce întâi în joc tineri și faptele bune vor apărea într-un ritm accelerat. Astfel dacă $k = 2 \cdot t$, conform enunțului, atunci o persoană va trebui să introducă t tineri și t vârstnici. În schimb dacă $k = 2 \cdot t + 1$, există șansa de a adăuga mai mulți tineri, mai exact $t + 1$, rămânând de adăugat t vârstnici. În general fiecare persoană va introduce întâi kt tineri și apoi kv vârstnici, $kt + kv = k$, kt și kv având valorile stabilite mai devreme, în funcție de paritatea lui k .

În ziua i , câte un tânăr este introdus în joc de câte un tânăr introdus în ziua $i - z_t, i - 2 \cdot z_t, \dots, i - kt \cdot z_t$. De asemenea câte un vârstnic va fi introdus în joc de câte un vârstnic din ziua $i - z_v, i - 2 \cdot z_v, \dots, i - kt \cdot z_v$.

În consecință avem:

$$t[i] = t[i - z_t] + t[i - 2 \cdot z_t] + \dots + t[i - kt \cdot z_t] \\ + v[i - z_v] + v[i - 2 \cdot z_v] + \dots + v[i - kt \cdot z_v]$$

(vom considera nule valorile cu index negativ)

În cazul lui $v[i]$ raționamentul este similar, cu diferența că orice persoană ce introduce un vârstnic trebuie să fi adăugat înainte kt tineri. Deci indicii sunt translați.

$$v[i] = t[i - (1 + kt) \cdot z_t] + t[i - (2 + kt) \cdot z_t] + \dots + t[i - (kv + kt) \cdot z_t] \\ + v[i - (1 + kt) \cdot z_v] + v[i - (2 + kt) \cdot z_v] + \dots + v[i - (kv + kt) \cdot z_v]$$

Întrucât pentru calculul lui $v[i]$ și $t[i]$ se fac $O(k)$ pași, avem o complexitate finală $O(n \cdot k)$.

Complexitate $O(k \cdot n)$

Soluția 9 – 90 p (sursa: pif_90p_2.cpp)

Putem optimiza recurențele de mai sus, observând că valorile $t[]$ sunt însumate din z_t în z_t , iar valorile $v[]$ sunt însumate din z_v în z_v . Astfel dacă calculăm sume parțiale cu un pas egal cu z_t , respectiv z_v , putem reduce calculul lui $v[i]$ și $t[i]$ la $O(1)$.

De exemplu, dacă am calcula

$$st[j] = t[j] + t[j - zt] + t[j - 2*zt] + \dots,$$

care de fapt este

$$st[j] = t[j] + st[j - zt],$$

putem calcula $t[i]$ din formulă, atunci când $zt < zv$, ca fiind

$$t[i] = st[i - zt] - st[i - (kt + 1)*zt]$$

Complexitate $O(n)$