

Feladat Calculator

Bemeneti fájl calculator.in
Kimeneti fájl calculator.out

Adott N folyamat, amelyeket egy számítógépen kell lefuttatni. Rendelkezésre áll M változó, amelyekhez ezek a folyamatok hozzáférhetnek, 1-től M -ig számozva. Minden i -edik folyamat (ahol $1 \leq i \leq N$) egy folytonos változóintervallumhoz $[l_i, r_i]$ fér hozzá, és az i -edik folyamat végrehajtásához szükséges idő t_i .

Amíg egy folyamat fut, az összes szükséges változót a végrehajtási verembe (stack) menti, és a befejezésekor eltávolítja azokat onnan. Két folyamat akkor és csak akkor futhat párhuzamosan, ha nincs közös változójuk. Az operációs rendszer úgy ütemezi a folyamatokat, hogy az összes végrehajtásához szükséges teljes idő minimális legyen. Egy ilyen, a minimális időt elérő ütemezést *optimális végrehajtási forgatókönyvnek* nevezzük.

Egy változót *kritikusnak* tekintünk, ha létezik legalább egy olyan optimális végrehajtási forgatókönyv, amelyben ez a változó tölti a legtöbb időt a veremben az összes változó közül (egy forgatókönyvben egyszerre több kritikus változó is lehet).

Követelmény

Adott egy $C \in \{1, 2, 3\}$ szám, amely a feladat típusát jelöli:

- 1. követelmény** Határozza meg a megadott N folyamat lefuttatásához szükséges minimális időt.
- 2. követelmény** Tegyük fel, hogy eltávolítjuk a rendszerből az i -ik folyamatot, minden i esetén 1-től N -ig. Legyen T_i a maradék $N - 1$ folyamat lefuttatásához szükséges minimális idő. Számítsa ki és írja ki az összes T_i érték összegét, ahol i 1-től N -ig fut.
- 3. követelmény** Tegyük fel, hogy eltávolítjuk a rendszerből az i -ik folyamatot, minden i esetén 1-től N -ig. Legyen V_i a kritikus változók száma a maradék $N - 1$ folyamat kontextusában, a korábbi definíció alapján. Számítsa ki és írja ki az összes V_i érték összegét, ahol i 1-től N -ig fut.

Adatok generálása

A nagyon nagy mennyiségű adat beolvasásának elkerülése érdekében csak az első K folyamatot olvassuk be a bemeneti állományból (l_i, len_i, t_i) számhármassok formájában, ahol i 1-től K -ig fut. A $C = 1$ feladat esetén garantált, hogy $N = K$, így nincs szükség további generálásra.

A $C \in \{2, 3\}$ feladatok esetén csak az első K folyamatot olvassuk be. A többi folyamatot, $K + 1$ -től N -ig, a programjukban a következő algoritmussal kell generálni:

- 1: ciklus** $i \leftarrow K + 1, N$ **végezd el**
- $l_i \leftarrow ((l_{i-1} \text{ xor } (a \cdot l_{i-K})) \bmod M) + 1$
- $\text{len}_i \leftarrow ((\text{len}_{i-1} \text{ xor } (b \cdot \text{len}_{i-K})) \bmod M) + 1$
- $t_i \leftarrow ((t_{i-1} \text{ xor } (c \cdot t_{i-K})) \bmod T) + 1$
- 5: vége ciklus**

Megjegyzés: A bitenkénti **XOR** „kizáró vagy” logikai műveletet a C/C++ nyelvben a $\wedge$ operátor jelöli.

Minden folyamat esetében (függetlenül attól, hogy beolvastuk vagy generáltuk), az i -edik folyamat által elért intervallum jobb oldali végpontját a következő képlettel kell kiszámítani:

$$r_i = l_i + (\text{len}_i \bmod (M - l_i + 1))$$

Bemeneti adatok

A calculator.in bemeneti állomány első sora tartalmazza a C, N, M, K, T számokat, amelyeket az a, b és c konstansok követnek, mindegyiket egy-egy szóköz választja el. A következő K sor mindegyikén három szám található: l_i, len_i és t_i , amelyek az első K folyamat információit jelentik (ahol $1 \leq i \leq K$).

Kimeneti adatok

A calculator.out kimeneti állomány egyetlen sorában egyetlen természetes szám szerepeljen, amely a C feladat alapján kiszámított választ jelenti.

Megkötések és pontosítások

- $1 \leq N, M \leq 5\,000\,000$
- $1 \leq K \leq \min(N, 100\,000)$
- $1 \leq T \leq 10\,000$
- $1 \leq l_i, \text{len}_i \leq M$ és $1 \leq t_i \leq T$, minden $1 \leq i \leq K$ esetén
- $1 \leq a, b, c \leq 1\,000\,000\,000$
- **Figyelem:** Az adatgeneráló algoritmus köztes számításai meghaladhatják a 32 bites egész típusok (pl. `int` vagy `unsigned int`) ábrázolási tartományát. Ahol szükséges, ajánlott a 64 bites adattípusok (pl. `long long`) használata.

#	Pontszám	Megkötések
1	15	$C = 1, 1 \leq N, M \leq 2\,000, K = N$
2	15	$C = 1, 1 \leq N, M \leq 100\,000, K = N$
3	5	$C = 2, 1 \leq N, M \leq 200, K = N$
4	5	$C = 2, 1 \leq N, M \leq 2\,000, K = N$
5	10	$C = 2, 1 \leq N, M \leq 100\,000, K = N$
6	5	$C = 2, 1 \leq N, M \leq 2\,000\,000$
7	10	$C = 2$
8	5	$C = 3, 1 \leq N, M \leq 200, K = N$
9	5	$C = 3, 1 \leq N, M \leq 2\,000, K = N$
10	10	$C = 3, 1 \leq N, M \leq 100\,000, K = N$
11	5	$C = 3, 1 \leq N, M \leq 2\,000\,000$
12	10	$C = 3$

Példák

calculator.in	calculator.out
1 3 5 3 10 1 1 1 1 3 5 3 3 4 4 2 2	9
2 3 5 3 10 1 1 1 1 3 5 3 3 4 4 2 2	20
3 3 5 3 10 1 1 1 1 3 5 3 3 4 4 2 2	3

Magyarázatok

A fenti példákhoz $N = 3$ folyamatunk és $M = 5$ változónk van. A folyamatok a következők:

1. folyamat: $l_1 = 1, r_1 = 1 + (3 \bmod 5) = 4$, idő $t_1 = 5$. Az intervallum: $[1, 4]$.
2. folyamat: $l_2 = 3, r_2 = 3 + (3 \bmod 3) = 3$, idő $t_2 = 4$. Az intervallum: $[3, 3]$.
3. folyamat: $l_3 = 4, r_3 = 4 + (2 \bmod 2) = 4$, idő $t_3 = 2$. Az intervallum: $[4, 4]$.

A folyamatok közötti kölcsönhatást elemezve megfigyelhető, hogy:

- Az 1. folyamatnak közös változói vannak a 2. folyamattal (3. változó) és a 3. folyamattal (4. változó) is. Ezért a P_1 nem futthat párhuzamosan sem a P_2 -vel, sem a P_3 -mal.
- A 2. folyamatnak és a 3. folyamatnak nincs közös változója, így futhatnak egyszerre.

Az első példához ($C = 1$): Egy optimális forgatókönyvben P_1 egyedül fut (5 időegység), a P_2 és P_3 pedig párhuzamosan. A $\{P_2, P_3\}$ csoport számára szükséges idő $\max(t_2, t_3) = 4$. A minimális összidő $5 + 4 = 9$.

A második példához ($C = 2$):

- Eltávolítjuk a P_1 -et: Marad a P_2 és P_3 , amelyek párhuzamosan futnak. $T_1 = \max(4, 2) = 4$.
- Eltávolítjuk a P_2 -t: Marad a P_1 és P_3 . Mivel a 4. változó közös, egymás után (szekvenciálisan) kell futniuk. $T_2 = 5 + 2 = 7$.
- Eltávolítjuk a P_3 -at: Marad a P_1 és P_2 . A 3. változó közös, tehát egymás után futnak. $T_3 = 5 + 4 = 9$.

Az eredmények összege $4 + 7 + 9 = 20$.

A harmadik példához ($C = 3$):

- P_1 nélkül: Az optimális idő 4. A 3. változó 4 időegységet tölt a veremben (amíg a P_2 fut), a 4. változó pedig 2 időegységet (amíg a P_3 fut). Az egyetlen kritikus változó a 3. változó. $V_1 = 1$.
- P_2 nélkül: Az optimális idő 7. A P_1 és P_3 folyamatok egymás után futnak. A 4. változó összesen $5 + 2 = 7$ időegységet tölt a veremben, ez az egyetlen, amely eléri ezt a maximális időt. $V_2 = 1$.
- P_3 nélkül: Az optimális idő 9. A P_1 és P_2 folyamatok egymás után futnak. A 3. változó összesen $5 + 4 = 9$ időegységet tölt a veremben, ő az egyetlen kritikus változó. $V_3 = 1$.

A V_i értékek összege $1 + 1 + 1 = 3$.