OLIMPIADA DE INFORMATICĂ – FAZA PE SECTOR

7-8 februarie 2004

CLASELE a XI-a şi a XII-a

PROBLEMA 1

Munca

(100 de puncte)

La o fabrică din Bucureşti cei N angajaţi nu prea au chef de muncă. Ghinionul lor este că le-a venit un manager nou care chiar vrea să facă treabă. Acesta a împărţit fiecăruia exact câte o sarcină şi a calculat cât timp trebuie să dureze fiecare sarcină. Sarcina angajatului **i** durează $\mathbf{T_i}$ zile ($\mathbf{T_i}$ este un număr natural).

Ca să stea degeaba cât mai mult timp, angajaţii au început să comenteze: „Eu nu pot să-mi încep treaba până când nu termină unii dintre angajaţi!!!". Astfel fiecare angajat **i** a alcătuit o listă cu ceilalţi muncitori de care depinde el, adică acei muncitori ce trebuie să îşi fi terminat sarcinile lor, înainte ca muncitorul **i** să se poată apuca de treabă. Evident că unii nu depind de nimeni (lista lor are lungimea 0), deci ei se pot apuca imediat de treabă.

Managerul a fost nevoit să accepte acest lucru, dar a calculat şi timpul minim la care trebuie să se termine toate sarcinile. Calculul său respectă cererilor muncitorilor, dar în final, el a ameninţat că dacă se depăşeşte acel timp, atunci vor fi toţi concediaţi.

De parcă nu era de ajuns că o mare parte din timp vor sta degeaba, fiecare angajat s-a gândit să înceapă **cât mai târziu posibil, dar în aşa fel încât toate sarcinile să se termine în termenul stabilit**.

Cerinţa:

Să se determine timpul minim la care trebuie terminate sarcinile şi de asemenea, pentru fiecare angajat, durata maximă de întârziere (în zile, faţă de momentul când poate începe), **fără a afecta timpul total de terminare**. În calculul duratei maxime se presupune că toţi ceilalţi angajaţi vor începe lucrul imediat ce acest lucru este posibil (fiecare se gândeşte că **numai el va întârzia cât mai mult, iar ceilalţi vor începe cât mai devreme posibil**).

Date de intrare:

Fişierul de intrare **munca.in** are următoarea structură:

pe prima linie numărul **N**

urmează **N** linii; a **i**-a linie descrie sarcina angajatului **i** prin: $\mathbf{T_i}$ $\mathbf{D_i}$ $\mathbf{S_{i1}}$ $\mathbf{S_{i2}}$ ... $\mathbf{S_{iDi}}$ (întregi despărţiţi de un spaţiu), unde $\mathbf{T_i}$ este durata sarcinii angajatului **i**, $\mathbf{D_i}$ este numărul de angajaţi de care depinde angajatul **i**, (care trebuie să-şi termine treaba lor, astfel ca el să poată începe) iar $\mathbf{S_{i1}}$ $\mathbf{S_{i2}}$ ... $\mathbf{S_{iDi}}$ sunt angajaţii de care el depinde.

Date de ieşire:

Fişierul de ieşire **munca.out** va conţine $N°+1$ linii. Pe prima linie se află timpul minim de terminare a tuturor activităţilor (se consideră că activităţile pot începe în momentul de timp **0**). Pe următoarele **N** linii se află durata maximă de timp pe care fiecare angajat o poate întârzia.

Restricţii

$0 < N$, $T_i : < 101$

Observații:

Toate sarcinile sunt realizabile; nu există dependențe circulare (de forma **i** urmează lui **j**, iar **j** urmează lui **i**).

În listele de dependențe, numerotarea începe de la 1, iar angajații dintr-o listă nu se repetă.

Nu se acordă punctaje parțiale. Un test ia toate punctele doar dacă toate valorile sunt corecte.

Timpul trece, leafa merge, noi cu drag muncim.

Exemplu:

munca.in	munca.out
4 10 0 4 0 4 1 2 5 2 1 3	15 0 2 2 0

Explicații pentru exemplul din stânga
Muncitorul 1 are de lucru 10 zile; el nu depinde de nimeni, dar trebuie să se apuce imediat de lucru, pentru că altfel se măreşte timpul total de lucru.
Muncitorul 2 poate începe oricând (adică timpul 0 este cel mai devreme) şi poate întârzia maximum 2 zile, fără să afecteze timpul total de lucru al întregii echipe.
Muncitorul 3 depinde de muncitorul 2; el poate întârzia maximum 2 zile (se apucă cel mai devreme în ziua a 4-a şi cel mai târziu în ziua a 6-a).
Muncitorul 4 are de lucru 5 zile şi poate începe doar după ce au terminat treaba muncitorii 1 şi 3 treaba lor. El trebuie să se apuce de treabă fără întârziere; altfel se va prelungi timpul total de lucru.

Timp maxim de execuție: 1 secundă

PROBLEMA 2

Robot

(100 puncte)

Un robot se plimbă pe un şir infinit de căsuțe, numerotate din 1 în 1, de la $-\text{infinit}$ la $+\text{infinit}$. Inițial, la momentul $T = 0$, robotul este situat în căsuța **0. În fiecare secundă, robotul se poate deplasa în una dintre cele două căsuțe vecine sau poate sta pe loc. Deplasarea durează trei sferturi de secundă,** dar robotul **nu poate începe deplasarea decât la începutul unei secunde.**
Căsuțele pot fi colorate cu două culori: alb sau negru. Robotul se poate afla doar în căsuțe albe; dacă, la un moment dat, **robotul se află într-o căsuță neagră, atunci el va fi distrus.**

La momentul $T = 0$, toate căsuțele sunt colorate cu alb.

La anumite momente de timp pot avea loc evenimente; în total, sunt **N** evenimente. Un eveniment **i** este dat prin **4** numere întregi, $\mathbf{t_i}$ $\mathbf{x_i}$ $\mathbf{y_i}$ $\mathbf{c_i}$ cu semnificația: „la începutul ultimei zecimi a secundei $\mathbf{t_i}$, toate căsuțele de la $\mathbf{x_i}$ la $\mathbf{y_i}$ inclusiv sunt colorate instantaneu cu culoarea $\mathbf{c_i}$, indiferent de culoarea pe care o aveau înainte de acest moment". Culoarea **0** este alb , iar coloarea **1** este negru.

Colorarea în negru a căsuței în care se află robotul duce la distrugerea acestuia. Robotul ştie în ce constau evenimentele şi în ce ordine se vor succeda, deci va avea grijă să nu fie distrus.

Cerința:

Să se afişeze toate căsuțele în care se poate afla robotul, imediat după ultimul eveniment.

Date de intrare:

Fişierul **robot.in** conține pe prima linie numărul **N** de evenimente. Urmează **N** linii; evenimentul **i** este reprezentat pe linia **i** a fişierului, prin 4 numere întregi $\mathbf{t_i}$ $\mathbf{x_i}$ $\mathbf{y_i}$ $\mathbf{c_i}$ separate prin câte un spațiu. Se garantează că $t_i < t_{i+1}$ pentru orice **i** intre **1** şi $n-1$ inclusiv.

Date de ieşire:

Fişierul **robot.out** va conține pe prima linie numărul **NR** de căsuțe în care se poate afla robotul imediat după producerea ultimului eveniment. Pe următoarea linie se vor afişa numerele de ordine ale căsuțelor, în ordine crescătoare. Numerele vor fi separate prin câte un spațiu.

Restricții şi precizări:

$1 <= N <= 1\,000$; $1 <= t_i <= 1\,000\,000$;

pentru un anumit **i,** $\mathbf{c_i}$ este fie **0** (alb), fie **1** (negru);

se garantează că, pentru testele date, **robotul nu se va putea afla în mai mult de 10.000** căsuțe imediat după ultimul eveniment;

Exemplu:

robot.in	**robot.out**
5 1 1 4 1 3 –10 –5 1 5 1 3 0 6 –3 –2 1 7 10 20 0	5 –4 –1 0 1 2

Explicații pentru exemplul din stânga

Observați în ce căsuțe se putea afla robotul după un număr întreg de secunde:

nr. de secunde	**căsuțe posibile**
1	–1 0
2	–2 –1 0
3	–3 –2 –1 0
4	–4 –3 –2 –1 0
5	–4 –3 –2 –1 0
6	–4 –1 0 1
7	–4 –1 0 1 2

Timp maxim de execuție: 1 secundă

Autori:

Mihai Stroe Facultatea de Automatică şi Calculatoare, anul VI masterat
şi
Andrei Marius Facultatea de Automatică şi Calculatoare, student anul IV