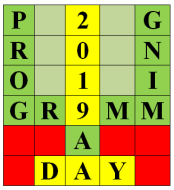
**Universitatea din Pitești, Facultatea de Educație Fizică și Informatică**
**Departamentul de Matematică-Informatică**
**Concursul de Informatică "Programming Day", Ediția a III-a** `Clasele XI-XII`
**18 mai 2019**




## Problema 1 – grafuri — 100 puncte

Se dau T grafuri neorientate notate cu $G_1$, $G_2$, …, $G_T$, fiecare prin numărul de noduri, numărul de muchii și muchii. Spunem că un graf neorientat este *2-neconex echilibrat*, dacă are exact două componente conexe, ambele componente conexe având același număr de noduri. Un graf neorientat este *aproape 2 – neconex echilibrat*, dacă prin adăugări de muchii (păstrând toate muchiile inițiale) se obține un graf *2-neconex echilibrat*.

**Cerinţă**

Cunoscând numărul de grafuri **T** și datele pentru fiecare graf $G_1$, $G_2$, …, $G_T$, se cere:

*1. numerele de ordine (în ordine crescătoare) ale grafurilor neorientate care sunt 2-neconexe echilibrate;*

*2. numerele de ordine (în ordine crescătoare) ale grafurilor neorientate care sunt aproape 2 – neconex echilibrate.*

**Date de intrare**

Fişierul de intrare *grafuri.in* conţine pe prima linie un număr natural **p**. Pentru toate testele de intrare, numărul **p** poate avea doar valoarea **1** sau **2**.

Pe linia a doua se află T, numărul de grafuri, iar pe următoarele linii datele pentru fiecare graf neorientat în formatul: $n_i$ $m_i$ (numărul de noduri și numărul de muchii separate prin câte un spațiu) pe o linie și $m_i$ muchii, pe câte o linie fiecare (capetele muchiilor fiind separate prin câte un spațiu), $1 \le i \le T$.

**Date de ieşire**

Dacă valoarea lui **p** este **1**, **se va rezolva numai punctul 1)** din cerință.

În acest caz, în fişierul de ieşire *grafuri.out* se va scrie *numerele de ordine (în ordine crescătoare) ale grafurilor neorientate care sunt 2-neconexe echilibrate*.

Dacă valoarea lui **p** este **2**, **se va rezolva numai punctul 2)** din cerință.

În acest caz, în fişierul de ieşire *grafuri.out* se va scrie *numerele de ordine (în ordine crescătoare) ale grafurilor neorientate care sunt aproape 2 – neconex echilibrate.*

**Restricţii şi precizări**

- $1 \le T \le 20$.
- $1 \le n_i \le 200$, $1 \le i \le T$.
- Orice graf *2-neconex echilibrat* este *aproape 2 – neconex echilibrat*.
- Pentru rezolvarea corectă a cerinței 1 se acordă 40% din punctaj și a cerinței 2 se acordă 60% din punctaj.

**Exemple**

| `grafuri.in` | `grafuri.out` | Explicaţie |
|---|---|---|
| **1**<br>3<br>4 2<br>**1 2**<br>**3 4**<br>7 4<br>**1 3**<br>**2 6**<br>**1 7**<br>**7 5**<br>6 1<br>**1 4** | **1** | p = 1<br>Primul graf neorientat este *2-neconex echilibrat*, celelalte două grfuri nu sunt *2-neconex echilibrate.*<br>**Atenție! Pentru acest test se rezolvă doar cerința 1).** |

| `grafuri.in` | `grafuri.out` | Explicaţie |
|---|---|---|
| **2**<br>3<br>4 2<br>**1 2**<br>**3 4**<br>7 4<br>**1 3**<br>**2 6**<br>**1 7**<br>**7 5**<br>6 1<br>**1 4** | **1 3** | p = 2<br>Primul și al doilea graf neorientat sunt *aproape 2 – neconex echilibrate.* Primul graf este *2-neconex echilibrat* și implicit aproape *2 – neconex echilibrat*. Al doilea graf are două componente conexe, una cu 4 noduri și alta cu 2 noduri => nu este graf *aproape 2 – neconex echilibrat*. La al treilea graf, dacă adăugăm, de exemplu muchiile [1,2], [3,5], [5,6] se obține un graf cu două componente conexe, fiecare cu câte 3 noduri, deci graful este *aproape 2 – neconex echilibrat.*<br>**Atenție! Pentru acest test se rezolvă doar cerința 2).** |

**Timp maxim de execuţie: 0.1 secunde/test. Memorie totală disponibilă 4 MB. Dimensiunea maximă a sursei: 5 KB.**

## Problema 2–music 100 puncte

Luca este un talentat cântăreț de pian. Mai nou, a și început să compună melodii. O melodie este reprezentată ca un șir de note muzicale (DO, RE, MI, FA, SOL, LA, SI). El folosește note din oricare dintre octavele 1-K. (Adică notele pe care le poate adăuga sunt – în ordine crescătoare: DO_1, RE_1, MI_1, FA_1, SOL_1, LA_1, SI_1, ..., DO_K, RE_K, MI_K, FA_K, SOL_K, LA_K, SI_K).

Mai mult, el și-a creat chiar și propria metrică pentru calitatea unei melodii, bazată pe un "grad de armonie". Acesta se determină astfel: se găsește cea mai înaltă notă din melodie(dacă sunt mai multe astfel de note, se alege cea mai din dreapta) și se calculeazălungimea maximă a unui subșir de note crescătoare din intervalul [prima notă, nota cea mai înaltă] (prologul melodiei). Apoi se găsește cea mai joasă notă din melodie(dacă sunt mai multe astfel de note, se alege cea mai din dreapta) și se calculează lungimea maximă a unui subșir descrescător din intervalul [prima notă, nota cea mai joasă]. Gradul de armonie al melodiei va fi suma celor 2 lungimi + lungimea melodiei.

Luca și-a propus să compună o melodie cu gradul de armonie minim H. Însă deoarece era prea entuziasmat, a ajuns să creeze o melodie mult prea lungă (și nu vrea să plictisească pe nimeni cu o melodie prea lungă ☺). Așa că își dorește să scurteze melodia compusă prin eliminarea unui sufix al său. (0 sau mai multe note de la final).
Vă roagă să îl ajutați să determine care este lungimea maximă a unui sufix ce poate fi eliminat astfel încât melodia rezultată să aibă încă gradul de armonie minim H.

**Cerință**

Cunoscând N (lungimea melodiei inițiale), H, precum și secvența de note care compun melodia inițială, se cere:

*1. Lungimea maximă a unui subșir de note crescătoare din prologul melodiei inițiale.*

*2. Lungimea maximă a unui sufix ce poate fi eliminat astfel încât melodia rezultată să își păstreze gradul de armonie cel puțin H.*

**Date de intrare**

Fișierul de intrare *music.in* conține pe prima linie un număr natural **p**. Pentru toate testele de intrare, numărul **p** poate avea doar valoarea **1**sau **2**.

Urmează o linie conținând cele 3 numere N, Kși H, separate prin câte unspațiu.

A 3-a linie conține cele N note ce compun melodia inițială, separate prin câte un spațiu.

**Date de ieșire**

Dacă valoarea lui **p** este **1**, **se va rezolva numai punctul 1)** din cerință.

În acest caz, în fișierul de ieșire *music.out* se va scrie un singur număr -lungimea maximă a unui subșir de note crescătoare din prologul melodiei inițiale.

Dacă valoarea lui **p** este **2**, **se va rezolva numai punctul 2)** din cerință.

În acest caz, în fișierul de ieșire se va scrie un singur număr - lungimea maximă a unui sufix ce poate fi eliminat astfel încât melodia rezultată să își păstreze gradul de armonie >= H.

**Restricții și precizări**

- **3**≤ N, H≤$10^5$, K ≤ 1000
- pentru 40% din teste, **3** ≤N, H ≤ 4000
- se garantează că melodia inițială are gradul de armonie ≥ H.
- pentru rezolvarea corectă a cerinței 1 se acordă 40%din punctaj, iar pentru cerința 2 se acordă 60%din punctaj

**Exemple**

| **music.in** | **music.out** | Explicație |
|---|---|---|
| **1**<br>7 10 6<br>DO_3 MI_1 RE_5 SI_7<br>MI_4 LA_2 SI_1 | **3** | p = 1<br>Nota cea mai înaltă este SI_7. Cele mai lungi subșiruri crescătoare din prolog sunt DO_3, RE_5, SI_7 și MI_1, RE_5, SI_7, de lungime 3.<br>**Atenție! Pentru acest test se rezolvă doar cerința 1).** |
| **music.in** | **music.out** | Explicație |
| **2**<br>7 10 6<br>DO_3 MI_1 RE_5 SI_7<br>MI_4 LA_2 SI_1 | **4** | p = 2<br>Gradul inițial de armonie = 3 (subșirul crescător) + 2 (subșirul descrescător) + 7 (lungimea melodiei) = 12.<br>Dacă eliminăm un sufix mai lung de 4, gradul de armonie devine ≤ 5.<br>**Atenție! Pentru acest test se rezolvă doar cerința 2).** |

**Timp maxim de execuție: 0.25 secunde/test. Memorie totală disponibilă:16MB. Dimensiunea maximă a sursei: 5 KB**