PA - TEMA 2

Responsabili:
Budulan Ștefania, Niculescu Irina, Sevastian Denisse,
Vasilescu Eugen, Vișan Radu

Deadline soft: **07.05.2017**
Deadline hard: **14.05.2017**

0.1 Modificări

- 24.04.2017 - Tema a fost configurată pe vmchecker
- 24.04.2017 - Tema poate fi rezolvată și în Python
- 24.04.2017 - Adăugare restricție pentru problema 2 (pag. 6)

CUPRINS

1 PROBLEMA 1: PERMUTĂRI

1.1 Enunț

Gigel tocmai ce a învățat la matematică ce sunt permutările și cum să le genereze. Ca temă de casă, a primit însă o problemă puțin diferită: i se dă o listă de cuvinte și trebuie să găsească o permutare a literelor din alfabetul englez conform căreia toate cuvintele date sa fie ordonate lexicografic. Cu alte cuvinte, dacă alfabetul ar fi reconstruit conform permutării generate a literelor, atunci cuvintele ar fi sortate lexicografic. Evident, acest lucru **nu** este totdeauna posibil, fiind parte din tema lui Gigel să-și dea seama **când** acest lucru nu se poate realiza.

Ordinea lexicografică a 2 cuvinte este definită în felul următor: se caută prima poziție pe care cele 2 cuvinte diferă. Dacă o astfel de poziție nu există, cuvântul mai scurt este considerat mai mic. În caz contrar, se compară pozițiile celor două litere în alfabetul ales (de exemplu aa < ab < abc folosind alfabetul englez standard).

1.2 Date de intrare

Datele se vor citi din fișierul **permutari.in**. Prima linie va conține numărul de cuvinte N, iar următoarele N linii vor conține câte un cuvânt.

1.3 Date de ieșire

În fișierul **permutari.out** se va afișa un șir format din toate cele 26 de caractere ale alfabetului englez într-o ordine ce ne satisface constrângerile problemei, sau șirul "Imposibil" dacă acest lucru nu este posibil.

1.4 Restricții și precizări

- Numărul de cuvinte $\leqslant 10^3$
- Cuvintele sunt formate din maxim 100 de caractere (doar litere mici ale alfabetului englez)
- **Se acceptă orice permutare ce respectă constrângerile problemei**

1.5 Testare și punctare

- Punctajul maxim este de **35** puncte.
- Timpul de execuție:
  - C/C++: **0.5 s**
  - Java: **0.5 s**
  - Python: **1 s**

- Sursa care conține funcția **main** trebuie obligatoriu denumită: **permutari.c**, **permutari.cpp**, **Permutari.java**, sau **permutari.py**.

1.6 Exemple

1.6.1 *Exemplu 1*

Exemplu 1		
permutari.in	permutari.out	Explicație
5 cal bal bac banca banchet	zyxwvutsrqpom**lcnb**kjigfed**ah**	Avem următoarele constrângeri: • c apare înaintea lui b • l apare înaintea lui c • c apare înaintea lui n • a apare înaintea lui h Trebuie deci să afișăm orice permutare a literelor care să respecte aceste constrângeri. Una dintre multele posibilități este cea oferită ca exemplu.

1.6.2 *Exemplu 2*

Exemplu 2		
permutari.in	permutari.out	Explicație
6 ax ay by bz cz cx	Imposibil	Avem urmatoarele constrângeri: • x apare înaintea lui y • a apare înaintea lui b • y apare înaintea lui z • b apare înaintea lui c • z apare înaintea lui x Afirmațiile 1, 3 si 5 nu pot fi îndeplinite simultan, deci ne este imposibil să găsim o soluție validă.

2 PROBLEMA 2: POKEMONI

2.1 Enunț

Din cauza temelor mult prea grele de la PA, Boolănel s-a lăsat de facultate și s-a decis să devină un maestru pokemon.

Acesta se află în orașul **1** și deține un număr de **P** pokemoni, cu ajutorul cărora trebuie să ajungă cât mai repede la o arenă, aflată în orașul **N**.

Drumul până la destinație nu este unul ușor, însă Boolanel se poate folosi de pokemonii săi pentru a depăși obstacolele întâmpinate. Astfel, pe fiecare drum parcurs în călătoria sa spre arenă are nevoie de un pokemon care să il însoțească. Boolanel cunoaște pentru **fiecare drum** existent și pentru **fiecare pokemon** care este timpul de parcurgere al drumului **dacă** este insoțit de pokemonul respectiv.

El este liber să își aleagă la început ce pokemon il va insoți, iar pe parcursul călătoriei are voie să își schimbe pokemonul care îl insoțește de **maxim K** ori. Aceste schimbări pot fi efectuate **doar în orașe**.

Boolanel ar dori să afle care este cel mai scurt timp în care poate ajunge la arenă, dar are nevoie de ajutorul vostru pentru a găsi răspunsul.

2.2 Date de intrare

Datele vor fi citite din fișierul **pokemoni.in**.

Pe prima linie se vor găsi 4 numere: **N**, **M**, **P** și **K**, reprezentând numărul total de orașe, numărul de drumuri ce conecteaza orașele, numărul de pokemoni și numărul total de schimbări de pokemoni permise pe parcursul călătoriei.

Pe următoarele **M** linii se află câte $\mathbf{P}+\mathbf{2}$ numere: primele 2 numere reprezintă indicii orașelor legate de acest drum, iar următoarele P timpii facuți pe acest drum cu fiecare dintre cei P pokemoni.

2.3 Date de ieșire

Rezultatul se va scrie în fișierul **pokemoni.out**. Acolo trebuie să se regăsească un singur număr, reprezentând timpul minim petrecut de Boolănel pentru a ajunge la destinație.

2.4 Restricții și precizări

- $\mathbf{N} \leqslant 100$ (numărul de orașe)
- $\mathbf{M} \leqslant 1000$ (numărul de drumuri)
- $\mathbf{P} \leqslant 20$ (numărul de pokemoni)
- $\mathbf{K} \leqslant 20$ (numarul de schimbari de pokemoni permise)
- Timpul petrecut cu orice pokemon intre două orașe adiacente $\leqslant 10^5$

- Drumurile sunt bidirecționale și au același cost indiferent de direcție
- Un pokemon poate fi folosit de oricâte ori este nevoie pe parcursul călătoriei
- Se garantează că există cel puțin un drum între nodurile **1** și **N**
- Pot exista mai multe drumuri între 2 orașe
- 30% din teste au **K** = 0
- 40% din teste au **K** = 1
- Restul testelor au **K** $\geqslant$ 2

2.5 Testare și punctare

- Punctajul maxim este de **40** puncte.
- Timpul de execuție:
  - C/C++: **1.5 s**
  - Java: **2 s**
  - Python: **2.5 s**
- Sursa care conține funcția **main** trebuie obligatoriu denumită: **pokemoni.c**, **pokemoni.cpp**, **Pokemoni.java** sau **pokemoni.py**.

2.6 Exemple

2.6.1 *Exemplu 1*

Exemplu 1		
pokemoni.in	pokemoni.out	Explicație
4 5 2 3 1 2 1 3 1 3 5 5 2 3 3 1 2 4 5 5 3 4 4 2	4	Avem un număr de 4 orașe, 5 drumuri între ele și 2 pokemoni. Ne dorim să ajungem de la orașul 1 la orașul 4, schimbând de maxim 3 ori pokemonul ce ne însoțește. Drumul minim este următorul: • Pornim cu pokemonul 1 și mergem in orașul 2 ⇒ timp petrecut = 1 • Schimbăm cu pokemonul 2 și mergem in orașul 3 ⇒ timp petrecut = 1 • Păstrăm pokemonul 2 și mergem in orașul 4 ⇒ timp petrecut = 2 Astfel, timpul total petrecut pe drum este 4. Cu toate că mai avem la dispoziție o schimbare, observăm ca aceasta nu ne ajută.

3 PROBLEMA 3: PATRULA

3.1 Enunț

Boolănel a reușit să ajungă la arenă in timpul cel mai scurt cu ajutorul vostru. Din păcate, el nu este nici un bun strateg, așa că nu a ales cei mai buni pokemoni pe care îi putea folosi în cadrul luptelor din arenă și a pierdut. Cu toate acestea, a decis că ar fi o idee bună să fure premiul și dorește să ajungă cu el cât mai repede acasă.

Arena se află în orasul **N**, iar casa lui se află in orașul **1**. Boolănel va folosi **întotdeauna** una dintre cele mai **scurte** căi între orașul **N** si orasul **1**, timpul de deplasare între 2 orașe vecine fiind de **o unitate de timp** (nu va mai fi însoțit de pokemonii săi).

Singurul obstacol pe care îl mai are de depășit este faptul că polițiștii sunt pe urmele sale. Se știe că aceștia patrulează toate străzile întunecate și, dacă îl vor prinde pe Boolănel, îl vor aresta. Inițial, toate străzile sunt întunecate.

Boolănel are posibilitatea de a aprinde luminile dintr-un oraș ales de el și, astfel, toate străzile care leagă orașul respectiv de un alt oraș vecin devin luminate, deci nu vor mai fi patrulate de polițiști.

Scopul său este să aleagă orașul în care să aprindă luminile, în așa fel încât **numărul mediu** de străzi luminate pentru toate **căile de lungime minimă** posibile să fie **maxim**.

Pentru a afla ce îl așteaptă pe drum și a putea lua cele mai bune alegeri, Boolănel vă roagă să îl mai ajutați cu următoarele informații:

- Numărul total de căi posibile de deplasare între orașul N și orașul 1 de lungime minimă. Două căi de deplasare sunt considerate distincte dacă la cel puțin un moment de timp se alege o stradă diferită.
- Numărul mediu maxim de străzi luminate pentru alegerea optimă menționată mai sus.

3.2 Date de intrare

Datele de intrare se vor citi din fișierul **patrula.in**. Acesta are următorul format:

Prima linie conține valorile **N** și **M**, reprezentând numărul de orașe, respectiv numărul de străzi existente.

Următoarele **M** linii conțin câte **2** numere **X** și **Y**, cu semnificația că există o stradă între orașul **X** și orașul **Y**.

3.3 Date de ieșire

În fișierul **patrula.out** se va scrie pe prima linie numărul total de căi minime de la orașul **N** la orașul **1**.

Pe a doua linie se va găsi numărul mediu maxim de străzi luminate, **afișat cu exact 3 zecimale**.

3.4 Restricții și precizări

- **N** $\leqslant$ 100 (numărul de orașe)
- **M** $\leqslant$ 1000 (numărul de străzi)
- Străzile sunt bidirecționale, iar traversarea unei astfel de străzi durează exact **0 unitate de timp**
- Se garantează că există cel puțin un drum între nodurile **1** și **N**
- **Pentru 40% din teste, numărul total de căi minime este <= 20000**
- **Pot exista mai multe străzi între aceleași orașe**

3.5 Testare și punctare

- Punctajul maxim este de **50** puncte.
- Timpul de execuție:
  - C/C++: **0.5 s**
  - Java: **0.5 s**
  - Python: **1 s**
- Sursa care conține funcția **main** trebuie obligatoriu denumită: **patrula.c**, **patrula.cpp**, **Patrula.java** sau **patrula.py**.

3.6 Exemple

3.6.1 *Exemplu 1*

Exemplu 1		
patrula.in	patrula.out	Explicație
4 4 1 2 2 4 1 3 3 4	2 1.000	Exista 2 căi de lungime minimă între 4 și 1, și anume 4-2-1, respectiv 4-3-1. Observăm că dacă alegem să aprindem luminile din orașul 2, avem 2 străzi luminate pe calea 4-2-1, respectiv 0 străzi luminate pe calea 4-3-1. Media este $(2+0)/2 = 1$. Aceeași medie se obține și dacă alegem să aprindem luminile din orașul 1, având o stradă luminată pe calea 4-2-1 și una pe calea 4-3-1, media fiind $(1+1)/2 = 1$.

3.6.2 *Exemplu 2*

Exemplu 1		
patrula.in	patrula.out	Explicație
5 6 1 2 1 3 2 4 3 4 4 5 4 5	4 2.000	Avem 4 căi de lungime minimă: 5-4-2-1 (folosind prima stradă de la 5 la 4), 5-4-2-1 (folosind a doua stradă de la 5 la 4), 5-4-3-1 (folosind prima stradă de la 5 la 4), 5-4-3-1 (folosind a doua stradă de la 5 la 4). Pentru a obține media maximă trebuie să luminăm orasul 4 ce ne va lumina câte 2 străzi pe fiecare din cele 4 drumuri minime. Media va fi (2 + 2 + 2 + 2) / 4 = 2. Dacă am fi ales orice alt oraș media ar fi fost doar 1.

4 PUNCTARE

- Punctajul este distribuit astfel:
  - Problema 1: 35p
  - Problema 2: 40p
  - Problema 3: 50p
  - 7.5 puncte vor fi acordate pentru coding style
  - 7.5 puncte vor fi acordate pentru comentarii și README

Punctajul pe README, comentarii și coding style este condiționat de obținerea a unui punctaj strict pozitiv pe cel puțin un test.

Tema valoreaza 125 de puncte, restul de 15 puncte fiind acordate bonus pentru rezolvarea tuturor taskurilor. Bonusul se împarte între problemele 2 si 3. În total se pot obține 140 de puncte (**NU** se trunchiază).

Pentru detalii puteți să vă uitați și peste regulile generale de trimitere a temelor.

- O temă care **NU** compilează va fi punctată cu 0.
- O temă care **NU** trece niciun test pe vmchecker va fi punctată cu 0.
- Vor exista mai multe teste pentru fiecare problemă în parte. Punctele pe teste sunt independente, punctajul pe un anumit test nefiind condiționat de alte teste.
- Fiecare problemă va avea o limită de timp pe test (precizată mai jos și pe pagina cu enunțul). Dacă execuția programului pe un test al acelei probleme va dura mai mult decât limita de timp, veți primi automat o puncte pe testul respectiv și execuția va fi întreruptă.
- În fișierul README va trebui să descrieți soluția pe care ați ales-o pentru fiecare problemă, să precizați complexitatea pentru fiecare și alte lucruri pe care le considerați utile de menționat.

4.1 Checker

- Arhiva se va trimite pe vmchecker, unde tema se va testa folosind un set de teste private.
- Pentru testarea locala, aveți disponibil un set de teste publice (de aceeași dificultate) pe pagina cu resurse a temei.
- **Punctajul pe temă** este cel de pe vmchecker și se acordă rulând tema doar cu testele private.
- Checkerul verifică doar existența unui README cu denumire corectă și conținut nenul. **Punctajul final pe README și comentarii** se acordă la corectarea manuală a temei.
- La corectare se poate depuncta pentru **erori de coding style** care nu sunt semnalate de checker.
- Corectorii își rezervă dreptul de a scădea puncte pentru orice problemă găsită în implementare, dacă vor considera acest lucru necesar.

5 FORMAT ARHIVĂ

- Temele pot fi testate automat pe vmchecker. Acesta suportă temele rezolvate în C/C++, Java și Python.

  Dacă doriți să realizați tema în alt limbaj, trebuie să-i trimiteți un email lui Traian Rebedea (traian.rebedea@cs.pub.ro), în care să îi cereți explicit acest lucru.
- Arhiva cu rezolvarea temei trebuie să fie **.zip**, având un nume de forma **Grupa_NumePrenume_Tema2.zip** (ex: 399CX_PuiuGigel_Tema2.zip sau 399CX_BucurGigel_Tema2.zip) și va conține:

  - Fișierul/fișierele sursă
  - Fișierul **Makefile**
  - Fișierul **README** (fără extensie)

- Fișierul pentru make trebuie denumit obligatoriu **Makefile** și trebuie să conțină următoarele reguli:

  - **build**, care va compila sursele și va obține executabilele (dacă acest lucru este necesar)
  - **run-p1**, care va rula problema 1
  - **run-p2**, care va rula problema 2
  - **run-p3**, care va rula problema 3
  - **clean**, care va șterge executabilele generate

- **ATENȚIE!** Funcția **main** din rezolvarea unei probleme se va găsi într-o sursă ce trebuie obligatoriu denumită astfel:

  - **permutari.c**, **permutari.cpp**, **Permutari.java** sau **permutari.py** - pentru problema 1
  - **pokemoni.c**, **pokemoni.cpp**, **Pokemoni.java** sau **pokemoni.py** - pentru problema 2
  - **patrula.c**, **patrula.cpp**, **Patrula.java** sau **patrula.py** - pentru problema 3

- **ATENȚIE!** Tema va fi compilată și testată **DOAR pe Linux**.
- **ATENȚIE!** Numele regulilor și a surselor trebuie să fie exact cele de mai sus. Absența sau denumirea diferită a acestora va avea drept consecință obținerea a 0 puncte pe testele asociate problemei rezolvate de regula respectivă.
- **ATENȚIE!** Pentru cei ce folosesc C/C++ **NU** este permisă compilarea cu opțiuni de optimizare a codului (O1, O2, etc.).
- **ATENȚIE!** Orice nerespectare a restricțiilor duce la pierderea punctajului (după regulile de mai sus).

6 LINKS

- Regulament general teme PA
- Google C++ Style Guide
- Google Java Style Guide
- Debugging și Structuri de Date