Tema 1 PA

- deadline 3 aprilie 2016 23:55 -

Responsabil Vicentiu Ciorbaru (cvicentiu@gmail.com)

1. Problema 1

1.1 Enunţ

Gigel doreşte să programeze primul său joc pe calculator. Neavând prea multă experienţă, s-a gândit să pornească cu ceva simplu. Şi-a imaginat un joc cu palindroame. Jocul va rearanja literele din cadrul unui palindrom. Scopul jucătorului este să readucă literele amestecate într-o ordine de palindrom. Regulile jocului sunt simple:

- Oricare 2 litere adiacente pot fi interschimbate.
- Jocul se termină când ordinea literelor este corespunzătoare cu cea de palindrom.

Gigel se gândeşte că trebuie să existe o strategie ce poate rearanja literele într-un număr minim de mutări. Pentru a se asigura că starea iniţială a jocului nu este nici prea grea, dar

nici prea usoara, Gigel ar dori să ştie în câte mutări ar putea să se termine jocul cel mai repede. El vă roagă să ii testaţi câteva valori propuse de el.

1.2 Date intrare

Fişierul de intrare **joc.in** va conţine mai multe cuvinte a căror litere au fost interschimbate.
Pe prima linie se va afla numărul *N* de cuvinte.
Pe fiecare din urmatoarele *N* linii se va afla un sir de caractere (litere mici ale alfabetului latin).

1.3 Date iesire

Fişierul de iesire **joc.out** va conţine *N* linii. Pe linia *i* se va afla numărul minim de mutări necesare pentru a transforma şirul de caractere aflat pe linia *i+1* în fisierul de intrare sau *-1* în cazul în care literele nu pot fi rearanjate pentru a se obţine un palindrom.

1.4 Precizari

1 <= N <= 1000
Dimensiune cuvânt <= 350

1.5 Exemplu

joc.in	**joc.out**
4 trtaccart iaian asd aabb	2 3 -1 2

2. Problema 2

2.1 Enunţ

Anii au trecut şi Gigel a devenit expert in programarea pe Android. El nu este foarte multumit de rezultatele funcţiei de recunoaştere vocală de la Google aşa că s-a gândit să creeze o soluţie proprie.

La rostirea unui cuvânt aplicaţia ii va returna un vector de variante de lungimi egale, apropiate de cuvantul initial. El consideră că soluţia este suficient de buna dacă **fiecare literă din cuvântul iniţial** se gaseste **in cel putin una** din variante. Pentru a evalua performanţa soluţiei sale Gigel s-a gândit la o funcţie. Funcţia de evaluare determină diferenţa dintre cuvântul iniţial şi cea mai buna combinaţie obţinută din variante, mai precis numărul de ajustări asupra cuvantului initial astfel incat cele 2 cuvinte sa devina identice.

F(cuvant, variante) = numărul minim de ajustări asupra cuvântului rostit.

O ajustare se referă la ştergerea unei litere, adăugarea unei litere sau înlocuirea unei litere cu o altă literă in cuvantul initial.

Având o serie de teste în care caracterele cuvintelor sunt codificate in cod ASCII (fiecare caracter este reprezentat de un numar cuprins intre 0 si 255 inclusiv) să se determine ce rezultate vor fi întoarse de funcţia de evaluare.

2.2 Date intrare

Fisierul **evaluare.in** va conţine pe prima linie 2 valori *NR_VAR* si *N* separate prin spaţiu, reprezentând numărul de variante, respectiv numărul de caractere din care e compusă fiecare variantă. Pe urmatoarele *NR_VAR* linii se găsesc *N* numere reprezentând codificarea în cod ASCII pentru fiecare variantă. Linia *NR_VAR+2* va conţine *M*, numărul de caractere din care e compus cuvântul rostit, iar pe linia *NR_VAR+3* se vor gasi *M* numere reprezentand
codificarea cuvântului iniţial.

2.3 Date iesire

Fisierul **evaluare.out** va conţine rezultatul întors de funcţia de evaluare aplicată pe setul de date din fisierul de intrare, evaluare.in.

2.4 Precizari

NR_VAR <= 600

M, N <= 1200

2.5 Exemple

evaluare.in	**evaluare.out**
3 5 1 5 10 8 7 0 8 9 7 7 1 2 3 4 5 5 1 2 3 4 5	0

Explicatie: Cuvântul rostit este identic cu o varianta, deci nu este nevoie de nicio ajustare.

evaluare.in	**evaluare.out**
3 5 1 5 10 8 7 0 8 9 7 7 1 2 3 4 5 4 1 11 7 5	2

Explicatie: Cuvântul rostit necesita o inserare pe pozitia 2 a unei valori dintre 5, 8 sau 2 şi o modificare a numărului 11 în una din valorile 3, 9 sau 10.

3. Problema 3

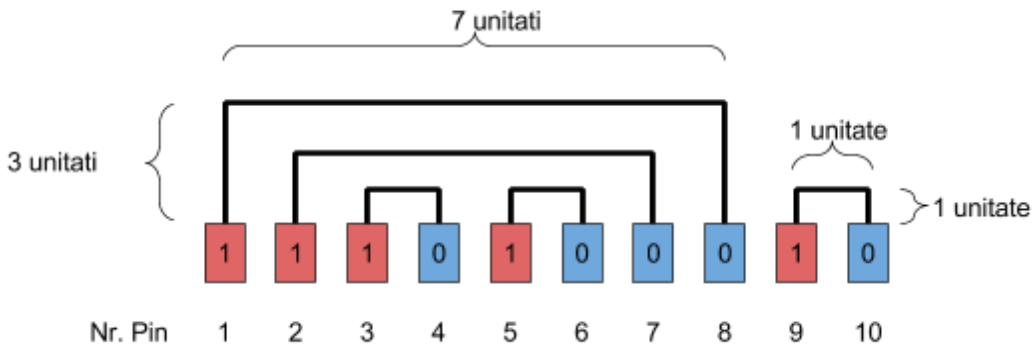
3.1 Enunt

Gigel şi-a descoperit o nouă pasiune, **circuitele electronice**. El vrea să îşi proiecteze propria sa placă de bază. Cum acest lucru este destul de complicat, s-a rezumat la a-şi conecta momentan doar un set de pini in configuratia ceruta de procesor. Gigel are câteva limitări de care trebuie să ţină cont:

- Există 2 tipuri de pini, de input (1) respectiv de output (0).
- Toţi pinii sunt poziţionaţi pe o singură linie pe placa de bază, la marginea inferioară.
- Un pin de input trebuie să se conecteze neapărat la un singur pin de output. Analog, un pin de output trebuie sa se conecteze neaparat la un singur pin de input.
- Traseele de conectare pot fi realizate doar pe verticala şi pe orizontala.
- Traseele de conectare nu se pot suprapune.

Întrucât spaţiul disponibil este mic, Gigel vă roagă să-l ajutaţi să traseze cele mai scurte trasee, dându-se configuraţia pinilor, astfel încât lungimea totală a traseelor să fie minimă.

Un exemplu de cablaj:



În exemplul de mai sus se observă cum fiecare pin de input (1), este conectat la un singur pin de output (0). Traseele nu se intersecteaza.
Pentru calculul distanţei se vor folosi următoarele distanţe:

- Distanta dintre 2 pini adiacenti este de 1 unitate.
- Cel mai mic segment de traseu pe verticala are 1 unitate.

Conform exemplului, distanţele traseelor dintre pini sunt următoarele:

- Perechea (1, 8): 3 vertical + 7 orizontal + 3 vertical = 13 unităţi.
- Perechea (2, 7): 2 vertical + 5 orizontal + 2 vertical = 9 unităţi.
- Perechea (3, 4): 1 vertical + 1 orizontal + 1 vertical = 3 unităţi.
- Perechea (5, 6): 1 vertical + 1 orizontal + 1 vertical = 3 unităţi.
- Perechea (9,10): 1 vertical + 1 orizontal + 1 vertical = 3 unităţi.
- Total: 31 unităţi.

3.2 Date intrare

Fişierul de intrare **cablaj.in** va conține pe prima linie numărul *N* al pinilor de tip 1. *N* reprezinta deasemenea şi numărul pinilor de tip 0.
Pe următoarea linie se află o succesiune de cifre 0 şi 1, neseparate prin spații, reprezentând configuraţia pinilor. În total 2 * N cifre.

3.3 Date iesire

Fişierul de ieşire **cablaj.out** va conține pe prima linie un număr natural nenul, reprezentând lungimea minimă totală a traseelor de conectare folosite, iar fiecare dintre următoarele N linii va conține două numere naturale x, y, separate printr-un spațiu, reprezentând numărul de ordine a doi pini care se vor conecta; perechile care se vor scrie în fişier vor fi ordonate crescător în funcție de valoarea x.

3.4 Precizari:

1 <= N <= 50

3.5 Exemple:

cablaj.in	**cablaj.out**
5 1110100010	31 1 8 2 7 3 4 5 6 9 10

4. Punctare

Punctajul pentru primele două probleme este de 40 de puncte fiecare. A treia problema are un punctaj de 60 de puncte. 10 puncte vor fi acordate pentru coding-style, 10 puncte pentru comentarii şi README. Punctajul pe README si comentarii este condiţionat de obţinerea a unui punctaj pozitiv pe cel puţin un test. Pentru detalii puteţi să vă uitaţi şi peste regulile generale de trimitere a temelor.

Prima problemă este obligatorie. Dintre problemele 2 si 3, puteţi rezolva una, la alegere. Punctajul maxim care va fi acordat pe tema este de 120 de puncte.

Vor exista mai multe teste pentru fiecare problemă in parte. Punctele pe teste sunt independente, punctajul pe un anumit test nefiind condiţionat de alte teste. Fiecare problemă va avea o limită de timp pe test (precizată mai jos) - dacă execuţia programului pe un test al acelei probleme va dura mai mult decât limita de timp veţi primi automat 0 puncte pe testul respectiv şi execuţia va fi întreruptă. În fişierul README va trebui să descrieţi soluţia pe care aţi ales-o pentru fiecare problemă, să precizaţi complexitatea pentru fiecare şi alte lucruri pe care le consideraţi utile de menţionat. Corectorii îşi rezervă dreptul de a scădea puncte dacă vor considera acest lucru necesar.

5. Format arhivă şi testare

Temele pot fi testate automat pe vmchecker - acesta suportă temele rezolvate in C/C++ şi Java.
Dacă doriţi să realizaţi tema în alt limbaj trebuie să-i trimiteţi un e-mail lui Traian Rebedea (traian.rebedea@cs.pub.ro) în care să îi cereţi explicit acest lucru.

Arhiva cu rezolvarea temei trebuie să fie .zip şi să conţină:

- Fişierul/fişierele sursă
- Fişierul Makefile
- Fişierul README

Fişierul pentru make trebuie denumit obligatoriu Makefile şi trebuie să conţină următoarele reguli:

- build, care va compila sursele şi va obţine executabilele
- run-p1, care va rula executabilul pentru problema 1
- run-p2, care va rula executabilul pentru problema 2
- run-p3, care va rula executabilul pentru problema 3
- clean, care va şterge executabilele generate

Atentie! Numele regulilor trebuie să fie exact cele de mai sus, în special pentru cele de run. Absenţa sau denumirea diferită a acestora va avea drept consecinţă obţinerea a 0 puncte pe testele asociate problemei rezolvate de regula respectivă.

Atenţie! Pentru cei ce folosesc C/C++ **NU** este permisă compilarea cu opțiuni de optimizare a codului (O1, O2, etc.).
Limitele de timp pentru problema 1 sunt:

- C/C++: 0.5 secunde
- Java: 0.75 secunde

Limitele de timp pentru problema 2 sunt:

- C/C++: 0.2 secunde
- Java: 1.5 secunde (Java este considerabil mai incet in implementarea noastra de referinta)

Limitele de timp pentru problema 3 sunt:

- C/C++: 0.1 secunde
- Java: 0.2 secunde