



Проблем 1: Мађарски круг

Временско ограничење: 0.5 секунди
Меморијско ограничење: 32 MB

Текст проблема

Ерно Рубик је познати мађарски изумитељ и архитекта, најпознатији по тзв. Рубиковој коцки. Међутим, мало је познато да је он такође изумео круг, који ћемо звати “мађарски круг”. То је логичка загонетка у којој треба одредити удаљеност између две најближе и две најдаље тачке на кружници и то урадити у што мањем броју погледа на круг.

Играч ове загонетке у једном погледу на круг може да сазна растојање између неке две тачке на кругу - растојање се дефинише као дужина најкраћег пута од једне до друге тачке, при чему је дозвољено само кретање по обиму круга. Све тачке се, наравно, налазе на обиму и **никоје две тачке нису на истој позицији**.

Описи функција

Потребно је имплементирати функцију

$$Resi(N, L, H)$$

где је N број тачака у загонетки, L адреса променљиве у коју треба уписати тражено минимално растојање и H адреса променљиве у коју треба уписати тражено максимално растојање. На располагању вам је и функција

$$Rastojanje(u, v)$$

која враћа растојање између тачака са индексима u и v . Тачке су индексиране бројевима од 0 до $N - 1$.

Пример:

Нека је $N = 4$. Нека је растојање од тачке 2 до тачака 0, 1 једнако 2 а растојање од тачке 3 до тачака 0, 1 једнако 10. Може се закључити да су тачке 2 и 3 на различитим странама у односу на праву која спаја 0 и 1, као и да је тетива 2 – 3 пречник круга. Највећа удаљеност је онда једнака половини обима, односно 12 а најмања је 2.

Ограничења

- $2 \leq N \leq 100,000$
- Обим круга је највише 10^9

Подзадаци и бодовање

Тест примери су подељени у два подзадатка, у којима важе следећа додатна ограничења

- **ПОДЗАДАТАК 1 [30 ПОЕНА]:** $N \leq 1000$
- **ПОДЗАДАТАК 2 [70 ПОЕНА]:** Нема додатних ограничења

У зависности од броја позива функцији *Rastojanje* (означимо овај број са K), за сваки подзадатак добија се следећи проценат поена:



- **[100 %]**: Ако је у сваком тест примеру $K \leq 2N$
- **[60 %]**: Ако је у сваком тест примеру $K \leq 4N$
- **[30 %]**: Ако је у сваком тест примеру $K \leq 30N$
- **[10 %]**: Уколико ниједно од претходних ограничења није испоштовано

Детаљи имплементације

Потребно је да пошаљете тачно један фајл, под називом **krug.c**, **krug.cpp** или **krug.pas**, који имплементира горе поменућу функцију. Осим тражене функције, ваш фајл може садржати и додатне глобалне променљиве, помоћне функције и додатне библиотеке.

Зависно од програмског језика који користите, ваша функција/процедура мора бити следећег облика:

C/C++	<code>void Resi(int N, int* L, int* H)</code>
Pascal	<code>procedure Resi(N : longint; var L, H : longint);</code>

Функција *Rastojanje* биће следећег облика:

C/C++	<code>int Rastojanje(int u, int v)</code>
Pascal	<code>function Rastojanje(u, v : longint) : longint</code>

Уколико радите у C/C++-у, потребно је на почетку фајла ставити **#include "krug.h"** а уколико радите у Pascal-у, потребно је на почетку фајла ставити **Unit krug;** (ово је већ додато у фајловима који су вам обезбеђени).

Тестирање и експериментисање

Уз задатак, обезбеђени су вам *template* фајлови (**krug.c**, **krug.cpp**, **krug.pas**) које можете користити и мењати по потреби. Такође су вам обезбеђени програми (**grader.c**, **grader.cpp**, **grader.pas**) који служе да лакше тестирате кодове.

Ови програми читавају са стандардног улаза број N , затим позивају вашу функцију *Resi*. Сваки пут када се позове функција *Rastojanje* на стандардни излаз испишу се параметри u, v позива те функције, а након тога на стандардни улаз треба унети тражено растојање. По изласку из функције *Resi* испишују се вредности L, H .



Проблем 2: Игра

Временско ограничење: 0.6 секунди
Меморијско ограничење: 256 MB

Текст проблема

Мало ко зна да су Никола Тесла и Алберт Ајнштајн били јако добри пријатељи. Сваке недеље је један од њих постављао загонетке другоме. Овога пута је дошао ред на Алберта да постави загонетку. Он је задао загонетку у облику игре.

Да би се играла игра потребна је табла на којој се налази N поља која су нумерисана бројевима од 1 до N . Та поља су повезана уз помоћ M путева, где сваки пут повезује нека два поља. Табла је таква да је могуће кренути из било ког чвора и користећи дате путеве доћи до било ког другог чвора. На почетку игре, свака од K фигурица (означене бројевима од 1 до K) се постави на њено почетно поље и такође се за сваку од тих фигурица обележи и крајње поље, тј. поље где та фигурица мора да се нађе на крају игре.

У једном потезу, Тесла **мора** да помери **сваку** фигурицу са поља где се та фигурица налази на неко суседно поље. Два поља су суседна уколико међу датих M путева, постоји пут који спаја та два поља. На једном пољу у исто време може да се налази више фигурица. Поента игре је да се све фигурице у исто време нађу на њиховим крајњим позицијама. Приметимо да уколико је нека фигурица пре краја на својој крајњој позицији, она у следећем потезу мора да се помери са те позиције.

Добро је познато да су њих двојица рачунали дане другачије него што смо навикли, њихових недељу дана је трајало D дана, уместо нама добро познатих 7.

Први дан у њиховој недељи, Алберт је послао поставку игре Тесли, тј. таблу и обележена почетна и крајња поља за сваку фигурицу. Поставио му је услов да сваки дан мора да одигра тачно један потез и тражио је од њега да му каже за сваки дан у њиховој недељи који је најмањи број потеза (дана) које Тесла мора да одигра како би све фигурице биле на својим крајњим позицијама баш на тај дан у недељи.

Добили сте поставку игре и знате да је Тесла успешно дао одговор за сваки дан у њиховој недељи. Ваш задатак је да откријете које је одговоре Тесла послао Ајнштајну.

Описи функција

Потребно је имплементирати функцију

$$Igra(N, M, K, D, U[\dots], V[\dots], S[\dots], F[\dots], R[\dots])$$

где је:

- N – број поља на табли



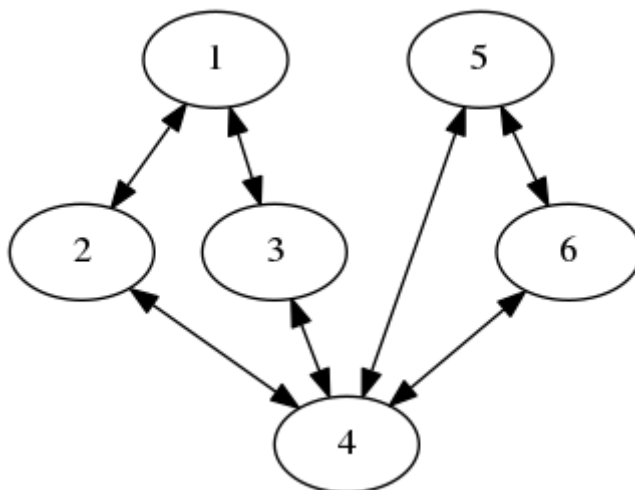
- M – број путева између поља
- K – број фигурица на табли
- D – број дана у Албертовој и Николиној недељи
- U и V – низови дужине M , где знамо да су поља означена бројевима $U[i]$ и $V[i]$ повезана путем на табли
- S и F – низови дужине K , где i -ти елементи низа означавају редни број почетног и крајњег поља за i -ту фигурицу, редом.

Функцији се такође прослеђује и параметар R који представља низ дужине D , кроз који треба да вратите ваше решење. По завршетку функције, i -ти елемент низа R треба да има вредност минималног броја дана (потеза) тако да све фигурице буду на својим крајњим позицијама на дан i у недељи. Уколико за неки дан у недељи није могуће да се све фигурице нађу на истом месту, решење за тај дан треба да буде -1 .

Сви низови су индексирани од 0, што значи да је први елемент низова онај са индексом 0. Водити рачуна да пошто су сви низови индексирани од 0, $R[0]$ треба да представља решење за дан 1, $R[1]$ за дан 2, итд...

Пример:

Нека постоји $N = 6$ поља, повезана уз помоћ $M = 7$ путева који су описани уз помоћ низова $U = \{1, 1, 2, 3, 4, 4, 5\}$ и $V = \{2, 3, 4, 4, 5, 6, 6\}$. Нека постоји $K = 2$ фигурице где су почетне и крајње позиције дате уз помоћ низова $S = \{1, 4\}$ и $F = \{2, 5\}$, и нека недеља траје $D = 3$ дана. На слици испод можете видети приказ такве табле.



Прва фигурица почиње на пољу 1 и треба да заврши на пољу 2, док друга фигурица почиње на пољу 4 и треба да заврши на пољу 5.

Недеља има 3 дана и за сваки од тих дана треба да испишемо најмањи број потеза да би обе фигурице биле на крајњим позицијама баш на тај дан у недељи.



- Дан 1 - најмањи број потеза је 3. Једно могуће решење је да прва фигурица прави шетњу: поље 1 (дан 1) → поље 3 (дан 2) → поље 4 (дан 3) → поље 2 (дан 1), док друга прави шетњу: поље 4 (дан 1) → поље 6 (дан 2) → поље 4 (дан 3) → поље 5 (дан 1).
- Дан 2 - најмањи број потеза је 1. Једно могуће решење је да прва фигурица прави шетњу: поље 1 (дан 1) → поље 2 (дан 2), док друга прави шетњу: поље 4 (дан 1) → поље 5 (дан 2).
- Дан 3 - најмањи број потеза је 5. Једно могуће решење је да прва фигурица прави шетњу: поље 1 (дан 1) → поље 3 (дан 2) → поље 4 (дан 3) → поље 3 (дан 1) → поље 4 (дан 2) → поље 2 (дан 3), а друга шетњу: поље 4 (дан 1) → поље 6 (дан 2) → поље 4 (дан 3) → поље 6 (дан 1) → поље 4 (дан 2) → поље 5 (дан 3).

Ограничења

- $1 \leq N \leq 1000$
- $1 \leq M \leq 100.000$
- $1 \leq K \leq 100$
- $1 \leq D \leq 1000$
- $1 \leq U[i], V[i] \leq N$, за свако $0 \leq i \leq M$
- $1 \leq S[i], F[i] \leq N$, за свако $0 \leq i \leq K$

Подзадаци и бодовање

- ПОДЗАДАТАК 1 [10 ПОЕНА]:** Између свака два поља на табли постојаће тачно један пут (дато је стабло) и важи $K = 1$.
- ПОДЗАДАТАК 2 [15 ПОЕНА]:** Између свака два поља на табли постојаће тачно један пут (дато је стабло).
- ПОДЗАДАТАК 3 [25 ПОЕНА]:** $D \leq 10, K \leq 10$ и D је паран број.
- ПОДЗАДАТАК 4 [10 ПОЕНА]:** $D \leq 10, K \leq 10$
- ПОДЗАДАТАК 5 [40 ПОЕНА]:** Нема додатних ограничења

Детаљи имплементације

Потребно је да пошаљете тачно један фајл, под називом **igra.c**, **igra.cpp** или **igra.pas**, који имплементира горе поменућу функцију. Осим тражене функције, ваш фајл може садржати и додатне глобалне променљиве, помоћне функције и додатне библиотеке.

Зависно од програмског језика који користите, ваша функција/процедура мора бити следећег облика:

C/C++	<code>void Igra(int N, int M, int K, int D, int* U, int* V, int* S, int* F, int* R);</code>
Pascal	<code>procedure Igra(N, M, K, D : longint; var U, V, S, F, R : array of longint);</code>



Уколико радите у C/C++-у, потребно је на почетку фајла ставити **#include "igra.h"** а уколико радите у Pascal-у, потребно је на почетку фајла ставити **Unit igra;** (ово је већ додато у фајловима који су вам обезбеђени).

Тестирање и експериментисање

Уз задатак, обезбеђени су вам "template" фајлови (igra.c, igra.cpp, igra.pas) које можете користити и мењати по потреби. Такође су вам обезбеђени програми (grader.c, grader.cpp, grader.pas) који служе да лакше тестирате кодове. Ови програми читавају са стандардног улаза следеће податке:

У првом реду бројеве N , M , K и D .

У следећих M редова бројеве $U[i]$ и $V[i]$, редом, раздвојене размаком

У наредних K редова бројеве $S[i]$ и $F[i]$, редом, раздвојене размаком

а затим позивају вашу функцију **Igra** из одговарајућег фајла (igra.c, igra.cpp, igra.pas) са учитаним параметрима и на крају вредност коју ваша функција враћа исписују на стандардни излаз. Кодове ових програма можете мењати по потреби.



Проблем 3: Мигови

Временско ограничење: 2 секунде
Меморијско ограничење: 256 MB

Текст проблема

У првој половини 20. века, уместо великог научног напретка, доминатни су били војни сукоби. Познато је да су велика имена науке, попут Алана Тјуринга и Џона фон Нојмана, дали велики допринос за бржи завршетак ових сукоба. Нажалост, људи лако заборављају историју, па и дан-данас често није битно да ли неко има британак ум и високо образовање већ да ли има велику војску и модерне Мигове 323...

Један од најпознатијих генерала данашњице, Дими Гави, највише од свега воли да гледа параду својих N најмодернијих Мигова 323 на небу и замишља да је Маверик из Топ Ган-а. Међутим, Дими Гави не воли свеж ваздух па их он не гледа на небу већ на монитору свог радара на коме се мигови виде као тачкице. Сваки од мигова се креће у једном од **4 смера (гледано на екрану, ти смерови су: горе, десно, доле и лево)** и **сви се крећу истом брзином** од 1 километра у секунди. Дими Гави каже да је растојање између два мига, која се налазе на позицијама (x_1, y_1) и (x_2, y_2) у координатном систему екрана, једнако $|x_1 - x_2| + |y_1 - y_2|$ и сви то поштују.

Дими Гави посматра **свака два мига** и у тренутку када они буду на **најмањем могућем међусобном растојању** (то може бити у почетном тренутку или у неком каснијем тренутку) он узвикне "Danger Zone!" и забележи то растојање. **Затим сва та растојања сабере**. Затим оде кући. На вама је да, знајући почетне позиције и смерове кретања свих мигова, **одредите који је резултат добио Дими Гави**.

Описи функција

Потребно је да имплементирате функцију

$$ZbirRastojanja(N, x[...], y[...], s[...])$$

где је N – број мигова а x, y и s низови који описују мигове: i -ти миг се налази на позицији $(x[i], y[i])$ и, уколико је $s[i] = 0$, он се креће нагоре, уколико је $s[i] = 1$, он се креће удесно, уколико је $s[i] = 2$, он се креће надоле и уколико је $s[i] = 3$ он се креће улево. **Сви низови су индексирани од 0**. Ова функција **мора да врати један цео број** – збир најмањих могућих растојања између свака два мига.

Пример:

Нека $N = 3$, $x = [0, 3, 5]$, $y = [3, 0, 4]$ и $s = [0, 1, 2]$. Ови подаци означавају да имамо 3 мига: први је на позицији (0,3) и иде нагоре, други је на позицији (3,0) и иде удесно и трећи је на позицији (5,4) и иде надоле. Најмање међусобно растојање између првог и другог мига се достиже у нултом тренутку и износи $|0 - 3| + |3 - 0| = 6$. Најмање растојање између првог и трећег мига је 5 и достиже се после пола секунде (ако претпоставимо да су јединична растојања – километри). Најмање растојање између другог и трећег мига је 2 (достиге се у више тренутака, нпр. у трећој секунди). Према томе, у овом случају ваша функција мора да врати број $6 + 5 + 2 = 13$.



Ограничења

- $2 \leq N \leq 200.000$
- За свако $i = \overline{0, N-1}$ важи $-5 \cdot 10^7 \leq x[i], y[i] \leq 5 \cdot 10^7$ и $s[i] \in \{0, 1, 2, 3\}$
- Све координате су цели бројеви
- Никоја два мига немају исте почетне координате

Подзадаци и бодовање

- **ПОДЗАДАТАК 1 [9 ПОЕНА]:** $N \leq 1.000$.
- **ПОДЗАДАТАК 2 [18 ПОЕНА]:** Сви мигови се крећу нагоре тј. $s[i] = 0$ за свако i .
- **ПОДЗАДАТАК 3 [24 ПОЕНА]:** Сваки миг се креће или лево или десно тј. $s[i] \in \{1, 3\}$ за свако i .
- **ПОДЗАДАТАК 4 [49 ПОЕНА]:** Нема додатних ограничења.

Детаљи имплементације

Потребно је да пошаљете тачно један фајл, под називом **migovi.c**, **migovi.cpp** или **migovi.pas**, који имплементира горе поменућу функцију. Осим тражене функције, ваш фајл може садржати и додатне глобалне променљиве, помоћне функције и додатне библиотеке.

Зависно од програмског језика који користите, ваша функција/процедура мора бити следећег облика:

C/C++	<code>long long ZbirRastojanja(int N, int* x, int* y, int* s);</code>
Pascal	<code>function ZbirRastojanja(N : longint; var x, y, s : array of longint) : int64;</code>

Уколико радите у C/C++-у, потребно је на почетку фајла ставити **#include "migovi.h"** а уколико радите у Pascal-у, потребно је на почетку фајла ставити **Unit migovi;** (ово је већ додато у фајловима који су вам обезбеђени).

Тестирање и експериментисање

Уз задатак, обезбеђени су вам "template" фајлови (migovi.c, migovi.cpp, migovi.pas) које можете користити и мењати по потреби. Такође су вам обезбеђени програми (grader.c, grader.cpp, grader.pas) који служе да лакше тестирате кодове. Ови програми учитавају са стандардног улаза следеће податке:

- У првом реду број N
- У следећих N редова бројеве $x[i], y[i], s[i]$, редом, раздвојене размаком

а затим позивају вашу функцију **ZbirRastojanja** из одговарајућег фајла (migovi.c, migovi.cpp, migovi.pas) са учитаним параметрима и на крају вредност коју ваша функција враћа исписују на стандардни излаз. Кодове ових програма можете мењати по потреби.