7th Asia-Pacific Informatics Olympiad

Diselenggarakan oleh

National University of Singapore, Singapura

Sabtu, 11 Mei 2013

Nama Soal	**ROBOTS**	**TOLL**	**TASKSAUTHOR**
Batas Waktu	1.5 s	2.5 s	Tidak berlaku
Ukuran Heap	128 MB	128 MB	Tidak berlaku
Ukuran Stack	32 MB	32 MB	Tidak berlaku
Poin	100	100	100
Banyak Subtask	4	5	8
Catatan	Kumpulkan Program	Kumpulkan Program	Kumpulkan berkas data untuk setiap subtask

Bahasa	Versi Compiler	Opsi Compiler
C	gcc versi 4.4.7	-O2 -lm
C++	g++ versi 4.4.7	-O2 -lm
Pascal	fpc versi 2.4.0	-O2

HAPPY PROGRAMMING!

Soal 1: ROBOTS

Para insinyur dari IRV (Institut Robotika Voltron) telah membangun sekumpulan n buah robot. Antara 2 robot yang saling cocok dan berdiri pada petak lantai yang sama akan bergabung membentuk robot gabungan.

Setiap robot diberi label dengan angka 1 hingga n ($n \leq 9$). 2 robot saling cocok apabila mereka mempunyai label yang saling berurutan. Pada awalnya, masing-masing dari n robot tersebut mempunyai 1 label yang unik. Sebuah robot gabungan yang terbentuk setelah menggabungkan 2 atau lebih robot lain akan diberikan 2 buah label, yang menyatakan label terkecil dan terbesar dari robot-robot yang bergabung membentuk robot gabungan tersebut.

Sebagai contoh, robot dengan label 2 hanya bisa bergabung dengan robot dengan label 3 atau robot berlabel 1. Jika robot 2 bergabung dengan robot 3, sebuah robot gabungan berlabel 2-3 akan terbentuk. Jika robot 2-3 bergabung dengan robot 4-6, sebuah robot gabungan berlabel 2-6 akan terbentuk. Robot dengan label 1-n akan terbentuk ketika semua robot bergabung.

Para insinyur menempatkan n buah robot tersebut di dalam sebuah ruangan yang terdiri dari $w \times h$ petak lantai dan dikelilingi oleh dinding. Beberapa petak lantai diberikan penghalang sehingga tidak bisa dilalui oleh para robot. Sebuah petak lantai dapat menampung satu atau lebih robot, dan sebuah robot pasti selalu menempati tepat 1 petak lantai. Pada awalnya, setiap robot ditempatkan pada petak lantai yang berbeda-beda.

Robot-robot yang baru dibuat memiliki mekanisme gerak yang sedikit primitif. Mereka hanya bisa bergerak lurus terhadap sumbu x atau sumbu y setelah didorong oleh seorang insinyur. Setelah didorong terhadap salah satu dari 4 arah yang sejajar sumbu x atau sumbu y, robot akan terus bergerak ke arah yang didorong hingga dia membentur penghalang atau dinding, dan kemudian berhenti bergerak. Setelah robot tersebut berhenti bergerak, ia akan memindai lingkungan sekitar untuk menemukan robot lain yang saling cocok dengan dirinya sendiri dan sedang menempati petak lantai yang sama dengannya, dan kemudian bergabung membentuk sebuah robot gabungan. Proses penggabungan ini terus berlangsung hingga tidak ada lagi penggabungan yang mungkin.

Untuk membantu para robot mengganti arah gerak, beberapa insinyur telah menempatkan cakram berputar di beberapa petak lantai. Cakram ini dapat berputar searah jarum jam atau berlawanan arah jarum jam. Robot yang mencapai petak lantai dengan cakram berputar di atasnya selalu mengubah arah geraknya sebanyak 90 derajat dengan arah sesuai arah putar cakram. Jika sebuah robot didorong ketika robot tersebut sedang berhenti di atas cakram berputar, robot tersebut akan berputar sebanyak 90 derajat sebelum bergerak lurus, dengan arah tegak lurus dengan arah dorongan yang diberikan.

Hanya 1 robot yang dapat bergerak pada setiap waktu.

Tugas Anda adalah menentukan banyak dorongan sesedikit mungkin yang diperlukan agar semua n robot dapat bergabung (jika mungkin).

Format Masukan

Program Anda harus membaca masukan dari masukan standar (*standard input*). Baris pertama dari berkas masukan berisi 3 bilangan bulat, n, w, dan h, yang dipisahkan oleh tepat sebuah spasi.

Berikutnya terdapat h baris untuk mendeskripsikan ruangan tersebut, di mana masing-masing baris berisi w buah karakter.

Karakter angka ('1' hingga '9') menyatakan bahwa ada robot dengan label sesuai karakter angka bersangkutan pada petak lantai tersebut. Karakter 'x' (huruf X kecil) menyatakan ada penghalang pada petak lantai tersebut. Karakter 'A' (huruf A besar) atau 'C' (huruf C besar) menyatakan ada cakram berputar pada petak lantai tersebut, di mana 'A' menyatakan cakram tersebut berputar berlawanan arah jarum jam, sementara karakter 'C' menyatakan cakram berputar searah jarum jam. Petak lantai lain akan dinyatakan dengan karakter '.' (tanda baca titik).

Format Keluaran

Program yang dibuat harus mencetak keluaran dari keluaran standar (*standard output*). Cetak sebuah angka yang menyatakan banyak dorongan sesedikit mungkin yang diperlukan agar semua robot dapat bergabung, atau -1 jika penggabungan total tidak dimungkinkan.

Subtask

Program yang dibuat akan diuji dengan 4 set kasus uji, masing-masing dengan batasan sebagai berikut:

1. (10 poin) $n = 2, w \leq 10, h \leq 10$, tanpa cakram berputar.
2. (20 poin) $n = 2, w \leq 10, h \leq 10$.
3. (30 poin) $n \leq 9, w \leq 300, h \leq 300$.
4. (40 poin) $n \leq 9, w \leq 500, h \leq 500$.

Contoh Masukan

```
4 10 5
1.........
AA...x4...
..A..x....
2....x....
..C.3.A...
```

Contoh Keluaran

```
5
```

Penjelasan

Pada solusi optimal dibutuhkan setidaknya 5 kali dorongan untuk menggabungkan semua robot.

1. Dorong robot 3 ke arah kanan. Robot tersebut akan bergerak ke kanan, menemui cakram berputar, berbelok 90 derajat berlawanan arah jarum jam, dan melanjutkan bergerak ke atas. Robot 3 pada akhirnya akan berhenti di depan dinding.

2. Dorong robot 4 ke atas. Robot 4 akan bergerak ke atas, berhenti di depan dinding, dan bergabung dengan robot 3 untuk membentuk robot 3-4.

3. Dorong robot 2 ke atas. Robot 2 akan bergerak ke atas, menemui cakram berputar, berbelok 90 derajat berlawanan arah jarum jam, menabrak dinding dan berhenti.

4. Dorong robot 2 ke kanan. Robot 2 kembali akan berbelok 90 derajat belawanan arah jarum jam, bergerak ke atas, kemudian berhenti di pojok kiri atas ruangan dan bergabung dengan robot 1 membentuk robot 1-2.

5. Dorong robot 3-4 ke kiri. Robot gabungan ini akan bergerak lurus ke arah kiri, berhenti di pojok kiri atas ruangan, dan kemudian bergabung dengan robot 1-2.

Soal 2: TOLL

Happyland dapat dideskripsikan dengan kumpulan N kota (dinomori 1 hingga N) yang awalnya dihubungkan oleh M buah jalan dua arah (dinomori 1 hingga M). Kota 1 adalah kota utama. Dijamin seseorang dapat berkunjung dari kota 1 ke semua kota lain melalui jalan dua arah yang ada. Jalan 2 arah tersebut adalah jalan tol, sehingga pengguna jalan i harus membayar biaya tol sebesar c_i sen kepada pemilik jalan. Diketahui bahwa semua biaya tol c_i berbeda-beda. Akhir-akhir ini, sebanyak K jalan baru tambahan telah selesai dibangun dan jalan-jalan ini dimiliki oleh milyuner Tuan Greedy. Tuan Greedy dapat menentukan biaya tol (tidak harus berbeda-beda) dari jalan-jalan yang baru dibangun, dan dia akan mengumumkan biaya tol tersebut pada esok hari.

Dua minggu dari sekarang akan diadakan karnaval akbar di Happyland. Banyak peserta karnaval akan berjalan menuju kota utama dan berparade sepanjang jalan. Sebanyak total p_j peserta akan berangkat dari kota j menuju kota utama. Mereka hanya akan melalui jalan-jalan yang sudah ditentukan, dan jalan yang ditentukan akan diumumkan sehari sebelum karnaval berlangsung. Menurut tradisi lama, jalan-jalan tersebut akan ditentukan oleh orang terkaya di Happyland, yang tak lain adalah Tuan Greedy. Karena terbatasi oleh tradisi yang sama, Tuan Greedy harus menentukan kumpulan jalan yang ***meminimalkan jumlah dari biaya tol pada jalan-jalan yang ditentukan*** dan pada saat yang sama jalan yang terpilih memungkinkan perjalanan dari kota j ke kota 1 (sehingga jalan-jalan yang terpilih akan membentuk sebuah "*minimum spanning tree*" di mana biaya tol adalah bobot/ *weight* dari sisi/ *edge* yang bersangkutan). Jika ada beberapa pemilihan kumpulan jalan yang memenuhi kedua kondisi di atas, Tuan Greedy dapat memilih salah satu dari beberapa pemilihan kumpulan jalan tersebut selama total biaya tolnya seminimal mungkin.

Tuan Greedy menyadari bahwa pendapatan yang dia peroleh dari K buah jalan yang baru dibangun tidak hanya bergantung pada biaya tol saja. Pendapatan dari sebuah jalan sebenarnya merupakan total biaya yang dikumpulkan dari orang-orang yang melalui jalan tersebut. Lebih tepatnya, jika p orang berjalan melalui jalan i, maka pendapatan dari jalan i adalah $c_i \times p$. Perhatikan bahwa Tuan Greedy hanya dapat menarik biaya tol dari jalan-jalan yang baru karena dia bukanlah pemilik dari jalan-jalan lama.

Tuan Greedy punya rencana licik. Dia berencana memaksimalkan pendapatannya selama karnaval berlangsung dengan memanipulasi biaya tol dan pemilihan jalan. Dia ingin menentukan biaya tol dari jalan-jalan baru (yang mana akan diumumkan besok), dan menentukan jalan-jalan yang digunakan untuk karnaval (yang mana akan ditentukan sehari sebelum karnaval) sedemikian sehingga memaksimalkan pendapatan yang diperoleh dari K buah jalan baru. Perhatikan juga Tuan Greedy masih harus mengikuti aturan tradisi dalam memilih jalan-jalan yang digunakan sehingga meminimalkan total biaya tol.

Anda adalah seorang reporter dan ingin membongkar rencana Tuan Greedy. Untuk itu, pertama-tama Anda harus menulis program untuk menentukan berapa besar pendapatan yang dapat diperoleh Tuan Greedy dari rencana liciknya.

Format Masukan

Program Anda harus membaca masukan dari masukan standar (*standard input*). Baris pertama berisi 3 buah bilangan bulat N, M, dan K yang masing-masing dipisahkan oleh spasi. M baris berikutnya mendeskripsikan M buah jalan lama, di mana baris ke-i berisi 3 buah bilangan bulat a_i, b_i, dan c_i yang dipisahkan oleh spasi dan menyatakan ada jalan dua arah antara kota a_i dan b_i dengan biaya tol c_i. K baris berikutnya mendeskripsikan K jalan tambahan yang baru dibangun, di mana baris ke-i berisi 2 buah bilangan bulat x_i dan y_i (dipisahkan oleh spasi) yang menyatakan ada jalan baru yang menghubungkan kota x_i dan kota y_i. Baris terakhir berisi N buah bilangan bulat yang dipisahkan oleh spasi, di mana bilangan ke-j pada baris tersebut merupakan p_j yang menyatakan banyak orang dari kota j yang berjalan menuju kota 1.

Masukan yang diberikan juga akan memenuhi batasan berikut.

- $1 \leq N \leq 100000$.
- $1 \leq K \leq 20$.
- $1 \leq M \leq 300000$.
- $1 \leq c_i, p_j \leq 10^6$ untuk setiap i dan j.
- $c_i \neq c_{i'}$, jika $i \neq i'$.
- Antara 2 kota, paling banyak terdapat 1 jalan (termasuk jalan yang baru dibangun).

Format Keluaran

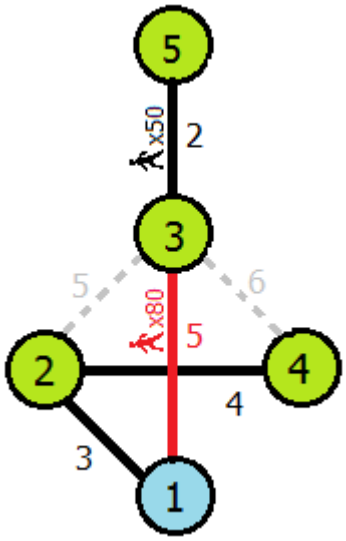
Program Anda harus mencetak ke keluaran standar (*standard output*) sebuah bilangan bulat, yang menyatakan total pendapatan maksimal yang mungkin didapat.

Contoh Masukan dan Keluaran

Masukan	Keluaran
5 5 1 3 5 2 1 2 3 2 3 5 2 4 4 4 3 6 1 3 10 20 30 40 50	400

Penjelasan

Berikut adalah ilustrasi dari contoh masukan,



Pada contoh masukan, Tuan Greedy harus menetapkan biaya tol jalan $(1, 3)$ menjadi 5 sen. Dengan biaya tol sebesar itu, dia dapat memilih jalan $(3, 5)$, $(1, 2)$, $(2, 4)$, dan $(1, 3)$ untuk meminimalkan total biaya tol, yang mana adalah 14 sen. 30 orang dari kota 3 dan 50 orang dari kota 4 akan berjalan melalui jalan tol baru menuju kota 1 sehinga Tuan Greedy dapat mengumpulkan pendapatan optimal sebesar $(30 + 50) \times 5 = 400$ sen.

Jika biaya tol dari jalan baru $(1, 3)$ diatur menjadi 10 sen, maka sesuai batasan dari tradisi, Tuan Greedy harus memilih jalan $(3, 5)$, $(1, 2)$, $(2, 4)$, dan $(2, 3)$ karena kumpulan jalan tol ini yang meminimalkan total biaya tol. Sehingga tidak ada pendapatan yang dapat dikumpulkan dari jalan baru $(1, 3)$ selama karnaval berlangsung.

Subtask

Program Anda akan diuji dengan 5 set kasus uji, masing-masing dengan batasan sebagai berikut:

1. (16 poin) $N \leq 10$, $M \leq 20$, dan $K = 1$.
2. (18 poin) $N \leq 30$, $M \leq 50$, dan $K \leq 10$.
3. (22 poin) $N \leq 1.000$, $M \leq 5.000$, dan $K \leq 10$.
4. (22 poin) $N \leq 100.000$, $M \leq 300.000$, dan $K \leq 15$.
5. (22 poin) $N \leq 100.000$, $M \leq 300.000$, dan $K \leq 20$.

Soal 3: TASKSAUTHOR

Saat ini terdapat banyak kontes pemrograman di seluruh dunia. Membuat soal kontes pemrograman yang bagus tidaklah mudah. Salah satu tantangannya adalah menyiapkan *test case* (atau kasus uji). *Test case* yang bagus haruslah dapat membedakan kode yang memenuhi tujuan, dengan kode lain yang terlihat benar namun gagal di beberapa kasus khusus.

Pada soal ini, peran Anda dalam kontes sudah ditentukan. Sebagai seorang programmer yang berpengalaman, Anda membantu panitia *Happy Programmer Contest* dalam menyiapkan *test case* mereka. Panitia telah memilih 2 soal graf dengan total 8 subtask berbeda, dan telah menulis beberapa kode yang mungkin dapat menyelesaikan soal graf tersebut. Dalam menentukan subtask, panitia bertujuan agar beberapa kode bisa mendapatkan semua poin, sementara kode lain mungkin mendapat 0 atau beberapa poin. Anda diberikan semua kode tersebut dalam versi C, C++, dan Pascal[1]. Untuk setiap subtask, tugas Anda adalah menghasilkan *test case* X yang dapat ***membedakan*** kedua kode yang diberikan, yaitu kode **A** dan kode **B.** Lebih spesifik, kedua kondisi berikut haruslah terpenuhi:

1. Pada masukan X, kode **A** tidak boleh menghasilkan *verdict* Time Limit Exceeded (TLE) (program berjalan melebihi batas waktu yang ditentukan).
2. Pada masukan X, kode **B** harus menghasilkan *verdict* Time Limit Exceeded (TLE).

Sebagai tambahan, panitia lebih menyukai *test case* yang berukuran kecil, dengan target paling banyak *T* buah bilangan bulat dalam *test case*.

Dua soal yang dipilih oleh panitia persoalan *Single-Source Shortest Paths* (SSSP, atau Jarak Terpendek dari 1 Sumber), dan sebuah soal graf yang diberi nama soal Mystery. Pseudo-code dari kode yang ditulis oleh panitia terdapat pada bagian lampiran dan implementasi dari pseudo-code tersebut dalam C++ dan Pascal dapat ditemukan pada lampiran berkas .zip pada soal 3 di server grader.

Subtask

Perhatikan Tabel 1. Setiap baris mendeskripsikan sebuah subtask. Perhatikan bahwa subtask 1 hingga 6 merupakan subtask dari soal SSSP sementara subtask 7 dan 8 merupakan subtask dari soal Mystery. Alokasi poin untuk setiap subtask dapat dilihat di kolom *S*.

Subtask	Poin *S*	Target *T*	Soal	Kode **A**	Kode **B**
1	3	107	SSSP	ModifiedDijkstra	FloydWarshall
2	7	2222	SSSP	FloydWarshall	OptimizedBellmanFord
3	8	105	SSSP	OptimizedBellmanFord	FloydWarshall
4	17	157	SSSP	FloydWarshall	ModifiedDijkstra
5	10	1016	SSSP	ModifiedDijsktra	OptimizedBellmanFord
6	19	143	SSSP	OptimizedBellmanFord	ModifiedDijkstra
7	11	3004	Mystery	Gamble1	RecursiveBacktracking
8	25	3004	Mystery	RecursiveBacktracking	Gamble2

Tabel 1: Deskripsi 8 Subtask.

[1]Semua kode mengimplementasikan algoritma yang diperbolehkan dalam silabus IOI.

Untuk mendapatkan poin pada sebuah subtask, *test case* X yang Anda buat harus dapat membedakan kode **A** dan kode **B** yang berkorespondensi dengan subtask tersebut. Sebagai tambahan, poin yang Anda dapat bergantung pada banyaknya bilangan bulat bertanda dalam X. Misalkan X mengandung F bilangan bulat, S poin dialokasikan untuk subtask tersebut, dan T adalah batasan ukuran yang diharapkan, maka poin yang diterima dikalkulasikan dengan formula berikut:

$$\lfloor 0.5 + S \times \min\{T/F, 1\} \rfloor$$

di mana $\lfloor \ \rfloor$ menyatakan operasi pembulatan ke bawah. Sehingga jika *test case* Anda mengandung tidak lebih dari T bilangan bulat, maka S poin akan secara penuh diberikan.

Penilaian

Anda harus menamai setiap *test case* dengan nama `taskauthor.outX.1` dengan `X` adalah nomor dari subtask. Sebelum mengumpulkan, Anda harus meng-kompresi berkas *test case* Anda menggunakan `gzip`. Dalam sistem UNIX, perintah berikut akan menghasilkan file yang sudah dikompresi:

```
tar -cvzf tasksauthor.tgz tasksauthor.out*.1
```

Pada sistem Windows, gunakan *software* seperti 7-Zip atau Winzip untuk menghasilkan berkas tar-gzipped. Kumpulkan hanya berkas `taskauthor.tgz` ke server grader.

Server grader akan membuka file yang sudah dikompresi. Untuk setiap subtask, misalkan subtask X, grader akan melakukan prosedur langkah berikut untuk menentukan banyaknya poin yang diberikan untuk subtask X:

1. Jika *test case* `tasksauthor.outX.1` tidak ada, berhenti dan tidak ada poin yang diberikan.

2. Periksa format dari `tasksauthor.outX.1`.
   Jika format masukan tidak valid, berhenti dan tidak ada poin yang diberikan.

3. Jalankan kode **A** dengan `tasksauthor.outX.1` sebagai masukan.
   Jika terjadi TLE, berhenti dan tidak ada poin yang diberikan.

4. Jalankan kode **B** dengan `tasksauthor.outX.1` sebagai masukan.
   Jika terjadi TLE, berhenti dan berikan poin dengan nilai sesuai hasil kalkulasi formula:

$$\lfloor 0.5 + S \times \min\{T/F, 1\} \rfloor$$

Setiap kode yang diberikan menggunakan *counter* (variabel `counter`) yang melacak banyaknya operasi yang dilakukan. Selama eksekusi kode, ketika nilai dari *counter* sudah melebihi 1.000.000, dianggap kode tersebut sudah menghasilkan *verdict* TLE.

Deskripsi Soal 1: Single-Source Shortest Paths (SSSP)

Diberikan sebuah graf berbobot berarah (*directed weighted graph*) G dan dua simpul s dan t dalam G, misalkan $p(s,t)$ menyatakan bobot jalur terpendek dari "tempat asal" s menuju "tempat tujuan" t. Jika t tidak mungkin tercapai dari s, maka $p(s,t)$ didefinisikan menjadi 1.000.000.000. Pada soal ini, masukan adalah graf G dan kumpulan Q pertanyaan $(s_1,t_1),(s_2,t_2),\ \ldots,(s_Q,t_Q)$. Keluaran adalah hasil dari pertanyaan yang bersesuaian pertanyaan $p(s_1,t_1),p(s_2,t_2),\ \ldots,p(s_Q,t_Q)$. .

Format Berkas Masukan/ Keluaran

Berkas masukan terdiri dari 2 bagian. Bagian pertama mendeskripsikan senarai ketetanggaan (*adjacency list*) dari graf berbobot berarah G. Bagian kedua mendeskripsikan pertanyaan jalur terpendek dalam G.

Bagian pertama dimulai dengan sebuah bilangan bulat V pada satu baris, yang merupakan banyaknya simpul dalam G. Simpul-simpul tersebut diberi label $0,1,\ldots,V-1$. Berikutnya terdapat V baris di mana masing-masing baris berkorespondensi dengan sebuah simpul, dimulai dari simpul 0. Setiap baris dimulai dengan bilangan bulat n_i yang menyatakan berapa banyak sisi keluar yang dimiliki simpul i. Berikutnya, terdapat n_i buah pasangan bilangan bulat (j,w) di mana setiap pasangan mendeskripsikan sebuah sisi berarah keluar. Bilangan bulat pertama j dalam pasangan bilangan bulat adalah label dari simpul yang ditunjuk oleh sisi tersebut, sementara bilangan bulat kedua w adalah bobot dari sisi tersebut.

Bagian kedua diawali dengan sebuah bilangan bulat Q dalam satu baris. Berikutnya terdapat Q baris, di mana baris ke-k berisi dua bilangan bulat s_k dan t_k, yang secara berturut-turut menyatakan simpul asal dan simpul tujuan.

Setiap dua buah bilangan bulat dalam satu baris harus dipisahkan oleh setidaknya satu karakter spasi. Sebagai tambahan, masukan memenuhi batasan berikut:

1. $0 < V \leq 300$,
2. n_i adalah bilangan bulat non-negatif $\forall i \in [0..V-1]$,
3. $0 \leq j < V$,
4. $|w| < 10^6$ di mana $|w|$ menyatakan nilai mutlak dari w,
5. $0 \leq \sum_{i=0}^{V-1} n_i \leq 5000$,
6. $0 < Q \leq 10$,
7. $0 \leq s_k < V, 0 \leq t_k < V, \forall k \in [1..Q]$, dan
8. graf G tidak boleh mempunyai siklis berbobot negatif (*negative weight cycle*) yang dapat dicapai dari simpul s_k mana pun yang muncul pada bagian kedua masukan.

Ingat kembali bahwa server grader akan memeriksa batasan di atas pada langkah C2.

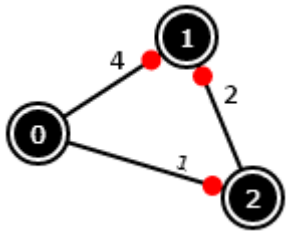
Format berkas keluaran kurang relevan dengan soal ini. Keluaran berisi tepat Q baris, dan baris ke-k berisi bilangan bulat $p(s_k,\ t_k)$. Untuk kemudahan, kode yang diberikan akan mencetak nilai dari variabel `counter` di akhir keluaran.

Contoh Berkas Masukan[2]

```
3
2 1 4 2 1
0
1 1 2
2
0 1
1 0
```

Contoh Berkas Keluaran[3]

```
3
1000000000
The value of counter is: 5
```



Gambar 1: Graf Berbobot Berarah dari Contoh Berkas Masukan

[2]Ada 15 bilangan bulat pada berkas masukan ini, sehingga $F = 15$.

[3]Nilai dari *counter* adalah 5 ketika berkas contoh masukan di atas dijalankan dalam ModifiedDijkstra.cpp/ pas.

Deskripsi Soal 2: Mystery

Diberikan sebuah graf tidak berarah G dengan V buah simpul dan E buah sisi, beri label kepada setiap simpul dalam G dengan sebuah bilangan bulat $\in [0..(X-1)]$ sehingga tidak ada 2 ujung dari sisi mana pun dalam G yang memiliki label yang sama. Nilai dari X haruslah nilai terkecil yang mungkin namun tetap mempertahankan syarat pelabelan setiap simpul dalam graf G.

Format Berkas Masukan/ Keluaran

Berkas masukan dimulai dengan dua buah bilangan bulat V dan E dalam satu baris. Berikutnya terdapat E baris, di mana setiap baris berisi dua buah bilangan bulat a dan b yang menyatakan sisi *tidak berarah* (a, b) ada dalam G. Sebagai tambahan, masukan yang diberikan memenuhi batasan berikut (yang akan diperiksa pada langkah C2):

1. $70 < V < 1000$,
2. $1500 < E < 10^6$, dan
3. untuk setiap sisi $(a, b), a \neq b, 0 \leq a < V, 0 \leq b < V$, dan sisi tersebut hanya muncul sekali dalam G.

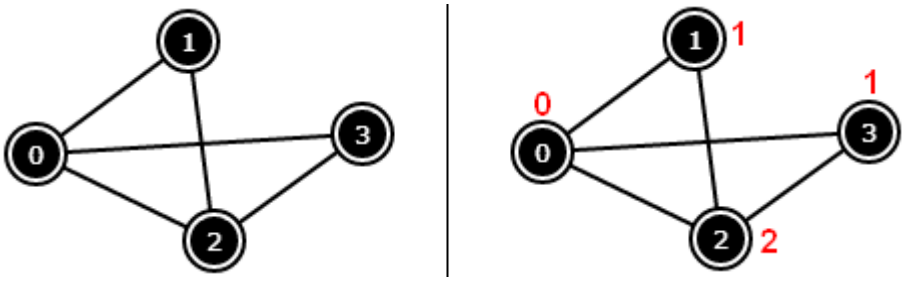
Berkas keluaran dimulai dengan sebuah bilangan bulat X pada satu baris yang menyatakan bilangan terkecil sehingga pelabelan simpul dimungkinkan. Baris berikutnya berisi V buah bilangan bulat yang mendeskripsikan label dari simpul 0, simpul 1, ..., simpul $V-1$, dan baris terakhir adalah nilai dari `counter`.

Contoh Berkas Masukan[4]

```
4 5
0 1
0 2
0 3
1 2
1 3
```

Contoh Berkas Keluaran[5]

```
3
0 1 2 1
The value of counter is: 18
```



Gambar 2: Kiri: Graf tidak berarah dari Contoh Berkas Masukan; Kanan: Label semua simpul dengan $\boldsymbol{X}$ = 3

[4]Ada 12 bilangan bulat pada berkas masukan, sehingga F = 12. Namun, contoh berkas masukan ini hanya sebagai ilustrasi. Berkas masukan tersebut tidak valid karena nilai V dan E terlalu kecil.

[5]Nilai dari *counter* adalah 18 ketika berkas contoh masukan di atas dijalankan dalam RecursiveBacktracking.cpp/ pas.

Lampiran: Pseudo-code

Berikut adalah algoritma dari kode yang disediakan. Variabel counter "mengira-ngira" *runtime* dengan melacak beberapa operasi. Server grader menggunakan versi C++ dari implementasi kode berikut.

FloydWarshall.cpp/ pas

```
// kondisi awal: informasi graf disimpan dalam matriks ketetanggaan M
counter = 0;
for k = 0 to V-1
   for i = 0 to V-1
      for j = 0 to V-1
         tambahkan counter dengan 1;
         M[i][j] = min(M[i][j], M[i][k] + M[k][j]);
for each pertanyaan p(s, t)
   cetak M[s][t];
```

OptimizedBellmanFord.cpp/ pas

```
// kondisi awal: informasi graf disimpan dalam senarai ketetanggaan L
counter = 0;
for each pertanyaan p(s, t)
   dist[s] = 0; // s adalah simpul tempat awal
   loop V-1 kali
      change = false;
      for each sisi (u, v) in L
      tambahkan counter dengan 1;
      if dist[u] + weight(u, v) < dist[v]
         dist[v] = dist[u] + weight(u, v);
         change = true;
      if change adalah false // ini adalah Bellman Ford yang 'dioptimasi'
         keluar dari loop paling luar;
   cetak dist[t];
```

ModifiedDijkstra.cpp/ pas

```
// kondisi awal: informasi graf disimpan dalam senarai ketetanggaan L
counter = 0;
for each pertanyaan p(s, t)
   dist[t] = 0;
   pq.push(pair(0, s)); // pq adalah sebuah priority queue
   while pq not empty
      tambahkan counter dengan 1;
      (d, u) = elemen paling atas dari pq;
      buang elemen paling atas dari pq;
      if (d == dist[u])
         for each sisi (u, v) in L
            if (dist[u] + weight(u, v) ) < dist[v]
               dist[v] = dist[u] + weight(u, v);
               masukkan pair (dist[v], u) ke dalam pq;
   cetak dist[t];
```

Gamble1.cpp/ pas

```
Beri nilai X = V dan beri label simpul i dalam [0..V-1] dengan i;
Beri nilai counter = 0; // tidak akan pernah TLE
```

Gamble2.cpp/ pas

```
Beri nilai X = V dan beri label simpul i dalam [0..V-1] dengan i;
Beri nilai counter = 1000001; // akan selalu mendapat TLE
```

RecursiveBacktracking.cpp/ pas

```
Algoritma ini mencoba nilai X dari 2 hingga V satu per satu
dan berhenti pada nilai X pertama yang valid.

Untuk setiap nilai X, fungsi backtracking memberi label simpul 0 dengan 0,
lalu untuk setiap simpul u yang telah diberi label,
fungsi backtracking mencoba memberi
nilai label sekecil mungkin hingga label X-1 untuk setiap tetangga v,
dan melakukan backtrack jika diperlukan.

// Silakan periksa RecursiveBacktracking.cpp/pas untuk melihat
// pada baris keberapa variabel counter ditambah dengan nilai 1.
```