**Computer Vision & Image Processing Lab**

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING | INTERNATIONAL UNIVERSITY

**IU TOP CODER 2016**

**Official Problems + Solutions**

* **Scoreboard Address:**

.................................................................................................................................................................

.................................................................................................................................................................

.................................................................................................................................................................

* **Problem:**

|  |  |  |  |
| --- | --- | --- | --- |
| **PROBLEM** | **NAME** | **SCORE** | **PAGE** |
| Problem 01 | Số Đối Xứng | 10 | 2 |
| Problem 02 | Số Rõ Ràng | 10 | 3 |
| Problem 03 | Tam Giác | 10 | 4 |
| Problem 04 | Trò Chơi | 15 | 5 |
| Problem 05 | Tin Nhắn | 20 | 6 |
| Problem 06 | Café Sữa | 15 | 7 |
| Problem 07 | Sự Trả Thù Của Người Sith | 10 | 9 |
| Problem 08 | Rào Cản | 10 | 11 |

HCMIU Online Judge | 01

Số Đối Xứng

IU Top Coder 2016

Timelimit: 1s

Cho số tự nhiên . Số là đối xứng nếu đọc từ trái qua phải và từ phải qua trái là như nhau. Hãy kiểm tra xem số có phải là đối xứng hay không? Giới hạn: .

**Input**:

* + - * Dòng đầu tiên chứa số nguyên - số lượng Tests,
      * Mỗi dòng trong dòng sau chứa một số nguyên

**Output**: gồm dòng, dòng thứ chứa kết quả kiểm tra đối xứng của Tests thứ . Nếu đối xứng trả lời YES, ngược lại trả lời NO.

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 3  1  232  34 | YES  YES  NO |

**Solution:**

#include <iostream>

using namespace std;

bool isSymmetric(string str) {

for (int i = 0, j = str.length() - 1; i < j; i++, j--) {

if (str.at(i)!= str.at(j)) return false;

}

return true;

}

int main(int argc, char\*\* argv) {

int length;

bool result;

string str;

cin >> length;

for (int i = 0; i < length; i++) {

cin >> str;

result = isSymmetric(str);

if (result) {

cout << "Yes" << endl;

} else {

cout << "No" << endl;

}

}

return 0;

}

HCMIU Online Judge | 02

Số Rõ Ràng

IU Top Coder 2016

Timelimit: 1s

Các nhà toán học đưa vào nhiều cách phân loại số, ví dụ, với các số nguyên ta có số chẵn và số lẻ, số chính phương và không chính phương. Bob cũng muốn đặt dấu ấn của mình trong lĩnh vực phân loại số. Bob chia các số nguyên dương thành 2 loại: rõ ràng và luẫn quẫn. Việc xác định một số thuộc loại nào được thực hiện theo giải thuật sau: với số nguyên dương , ta tạo số mới bằng cách lấy tổng bình phương các chữ số của nó, với số mới này ta lặp lại công việc trên. Nếu trong quá trình trên, ta nhận được số mới là 1, thì số ban đầu gọi là số rõ ràng.

Ví dụ với , ta có: . Như vậy, 19 là số rõ ràng.

Không phải mọi số đều rõ ràng. Ví dụ, với , ta có:

Rất thú vị với cách phân loại của mình, Bob muốn biết, trong thực tế, số rõ ràng nhiều hay ít?

**Yêu cầu**: Cho hai số nguyên dương và . Hãy xác định - số lượng số rõ ràng nằm trong đoạn

**Input**: gồm 1 dòng chứa hai số nguyên và

**Output**: số nguyên .

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 2 20 | 4 |

**Solution:**

#include <iostream>

#include <fstream>

#include <algorithm>

#include <vector>

#include <cmath>

#define SIZE 568

#define MAX 10000000

using namespace std;

//------------------------------------------------------------------------------

int lookupTable[] = {0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0,

0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,

0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0,

0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0,

0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,

0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,

0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,

0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1,

0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0,

0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,

0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,

0, 0, 0, 0, 0, 1, 0, 1, 1, 0};

long allNumber[MAX];

void findLookupTable();

//------------------------------------------------------------------------------

int main(int argc, char\*\* argv) {

int numberOfClearNumber = 0;

long A, B;

cin >> A >> B;

for (int i = 0; i <= B; i++) {

if (i < 10) {

allNumber[i] = i \* i;

} else {

allNumber[i] = allNumber[i / 10] + (i % 10)\*(i % 10);

}

}

for (int i = A; i <= B; i++) {

if (lookupTable[allNumber[i]] == 1) {

numberOfClearNumber++;

}

}

cout << numberOfClearNumber << endl;

return 0;

}

HCMIU Online Judge | 03

Tam Giác

IU Top Coder 2016

Timelimit: 1s

Cho điểm trên mặt phẳng, không có ba điểm nào thẳng hàng, các điểm được đánh số từ đến . Người ta nối tất cả các cặp điểm bằng sợi dây màu xanh hoặc màu vàng theo nguyên tắc: Nếu là số nguyên tố thì điểm nối với điểm bằng sợi dây màu xanh, ngược lại nếu không phải số nguyên tố thì nối bằng sợi dây màu vàng. Sau đó người ta muốn khảo sát xem có bao nhiêu hình tam giác mà ba đỉnh là điểm trong điểm được nối với nhau bằng các sợi dây cùng màu.

**Yêu cầu:** Cho n, hãy đếm số hình tam giác mà ba đỉnh là 3 điểm trong n điểm được nối với nhau bằng các sợi dây cùng màu.

**Input**: Dòng đầu tiên ghi số nguyên dương là số lượng bộ dữ liệu. Tiếp đến là dòng, mỗi dòng tương ứng với một bộ dữ liệu chứa một số nguyên .

**Output**: Gồm dòng, mỗi dòng chứa một số nguyên là số tam giác đếm được tương ứng với bộ dữ liệu vào.

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 2  3  5 | 0  1 |

**Solution:**

#include <stdio.h>

#include <iostream>

#include <algorithm>

using namespace std;

#define long long long

#define f1(i,n) for (int i=1; i<=n; i++)

#define f0(i,n) for (int i=0; i<n; i++)

#define N 2000006

int n, s[N];

bool np[N];

void eratos(){

np[0]=np[1]=true;

for (int i=2; i\*i<N; i++) if (!np[i])

for (int j=i\*i; j<N; j+=i)

np[j]=true;

f1(i,N-1) s[i]=s[i-1]+(!np[i]);

}

main(){

eratos();

int T; scanf("%d", &T);

while (T-->0) {

scanf("%d", &n);

long Sum = 0LL;

f1(i,n) Sum += 1LL \* (s[i+n]-s[i+(i==1)]) \* (n-(s[i+n]-s[i])-(i!=1));

cout << 1LL\*n\*(n-1)\*(n-2)/6 - Sum/2 << endl;

}

}

HCMIU Online Judge | 04

Trò Chơi

IU Top Coder 2016

Timelimit: 1s

Trên mỗi một trong số n quân bài có ghi một số nguyên dương nhỏ hơn 1000. Các quân bài được xếp thành một dãy, quân nọ nối tiếp quân kia. Bờm vừa phát minh ra một trò chơi mới. Trò chơi đòi hỏi thực hiện một số bước. Tại mỗi bước phải loại bỏ một trong các quân bài trong dãy, không được phép loại quân bài đầu tiên cũng như quân bài cuối cùng. Đối với mỗi quân bài bị loại bỏ sẽ nhận được số điểm bằng tích của số ghi trên quân bài này với tổng các số ghi trên hai quân bài xếp cạnh nó. Trò chơi kết thúc khi dãy chỉ còn hai quân bài đầu tiên và cuối cùng.

**Yêu cầu**: Hãy tìm số lượng điểm lớn nhất có thể đạt được từ trò chơi mà Bờm vừa phát hành.

**Input**:

* Dòng đầu tiên chứa số nguyên .
* Dòng tiếp theo chứa n số nguyên là các số viết trên các quân bài được liệt kê theo thứ tự chúng xuất hiện trên dãy quân bài.

**Output**: ghi ra một số nguyên là số lượng điểm lớn nhất tìm được.

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 4  4 5 6 2 | 86 |

Solution:

#include <stdio.h>

#include <iostream>

#include <algorithm>

using namespace std;

const int N = 702;

int n, a[N];

long F[N][N];

bool FF[N][N];

int f(int ll, int rr) {

if (FF[ll][rr]++) return F[ll][rr];

int Max = 0;

for (int i=ll+1; i<=rr-1; i++)

Max = max(Max, f(ll, i) + f(i, rr) + a[i]\*(a[ll]+a[rr]));

return F[ll][rr] = Max;

}

main() {

scanf("%d", &n);

for (int i=1; i<=n; i++)

scanf("%d", &a[i]);

cout << f(1, n) << endl;

}

HCMIU Online Judge | 05

Tin Nhắn

IU Top Coder 2016

Timelimit: 1s

Trong mét m¹ng gåm m¸y tÝnh ®¸nh sè tõ 1 ®Õn . S¬ ®å nèi m¹ng ®­îc cho bëi hÖ thèng gåm kªnh nèi trùc tiÕp gi÷a mét sè cÆp m¸y trong m¹ng. BiÕt chi phÝ truyÒn mét ®¬n vÞ th«ng tin theo mçi kªnh nèi cña m¹ng.

Ng­êi ta cÇn chuyÓn mét bøc th«ng ®iÖp tõ m¸y ®Õn m¸y . §Ó ®¶m b¶o an toµn, ng­êi ta muèn chuyÓn bøc th«ng ®iÖp nµy theo hai ®­êng truyÒn tin kh¸c nhau (tøc lµ kh«ng cã kªnh nµo cña m¹ng ®­îc sö dông trong c¶ hai ®­êng truyÒn tin). Chi phÝ cña mét ®­êng truyÒn tin ®­îc hiÓu lµ tæng chi phÝ trªn c¸c kªnh cña nã.

**Yªu cÇu:** Gi¶ sö bøc th«ng ®iÖp cã ®é dµi lµ 1 ®¬n vÞ th«ng tin, h·y t×m c¸ch chuyÓn th«ng ®iÖp tõ ®Õn sao cho tæng chi phÝ chuyÓn th«ng ®iÖp (b»ng tæng chi phÝ theo c¶ hai ®­êng truyÒn tin) lµ nhá nhÊt.

**Input**:

* Dßng ®Çu tiªn ghi bèn sè c¸ch nhau bëi dÊu c¸ch ();
* Mçi dßng thø trong sè dßng tiÕp theo ghi th«ng tin vÒ kªnh nèi thø cña m¹ng gåm ba sè , trong ®ã lµ chØ sè cña hai m¸y t­¬ng øng víi kªnh nµy vµ ( nguyªn d­¬ng) lµ chi phÝ ®Ó truyÒn mét ®¬n vÞ th«ng tin tõ m¸y di ®Õn m¸y (vµ ng­îc l¹i) theo kªnh nµy .

**Output**:

* + - * Dßng ®Çu tiªn ghi chi phÝ truyÒn th«ng ®iÖp theo c¸ch truyÒn tin t×m ®­îc;
      * Dßng thø hai ghi ®­êng truyÒn tin thø nhÊt d­íi d¹ng d·y cã thø tù c¸c m¸y b¾t ®Çu tõ m¸y kÕt thóc ë m¸y ;
      * Dßng thø ba ghi ®­êng truyÒn tin thø hai d­íi d¹ng d·y cã thø tù c¸c m¸y b¾t ®Çu tõ m¸y kÕt thóc ë m¸y .

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 5 7 1 5  1 2 3  1 4 8  2 3 5  2 4 4  3 5 5  4 3 8  4 5 3 | 24  1 2 3 5  1 4 5 |

HCMIU Online Judge | 06

Cafe Sữa

IU Top Coder 2016

Timelimit: 2s

Sau trận chung kết Woburn Challenge 2015-16, các lập trình viên trẻ từ khắp nơi trong cả nước sẽ được mời tham dự một bữa tiệc bí mật tại Woburn! Nhà trường đã quyết định chiêu đãi cho các lập trình viên bằng những cốc cafe sữa.

Hội sinh viên (SAC) đã mua bình chứa có dung tích không xác định và đặt chúng ở những độ cao khác nhau (so với mặt đất). Các bình chứa được đánh số từ ​​ đến theo thứ tự độ cao đã đặt bình; từ thấp nhất đến cao nhất. Mỗi bình (với ) sẽ có (lít) (với ) cafe sữa được bơm vào mỗi giây, trực tiếp từ trang trại sữa địa phương Scarborough. Ngoài ra, sẽ có một ống dẫn từ mỗi bình xuống một bình khác. Dưới tác động của trọng lực, cafe sữa chỉ có thể chảy từ bình cao xuống bình thấp hơn, do đó, đảm bảo rằng i cao hơn . Các ống dẫn ra từ bình sẽ đủ rộng để cho phép tối đa (lít) (với ) cafe sữa chảy qua mỗi giây. cafe sữa sẽ chảy theo hệ thống đã được thiết kế - cho tất cả các bình chứa, lượng cafe sữa chảy ra sẽ không vượt quá tổng lượng sữa chảy vào bình (bao gồm từ trang trại và từ các bình khác).

Tuy nhiên, năm nay PEG đã nhận đủ kinh phí để nâng cấp ống trong ống. Sau khi nâng cấp, mỗi đường ống sẽ có khả năng chứa được bất kỳ lượng cafe nào chảy qua.

Bình 1 nằm ở căn tin Woburn và các sinh viên đều được phép uống ở đó! Lập trình là một môn thể thao vất vả, vì vậy chúng tôi biết tất cả mọi người đều vô cùng khát nước. Do đó càng nhiều cafe e sữa được cung cấp thì mọi người sẽ được uống nhiều hơn. Và càng uống nhiều cafe sữa, kỹ năng lập trình sẽ càng được cải thiện cho giải năm sau.

Lập sơ đồ bể chứa và đường ống, và với điều kiện nâng cấp tối ưu cho ống dẫn, tính lượng sữa tối đa (lít/giây) chảy đến bình 1?

**Input**:

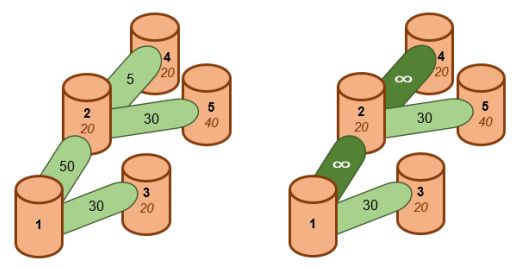
* Dòng 1 sẽ chứa hai số nguyên là và , lần lượt đại diện cho số bình chứa và số ống được nâng cấp.
* Dòng 2 đến N sẽ chứa thông tin của bình chứa. Cụ thể, dòng bao gồm ba số nguyên là và , tương ứng với lượng sữa được bơm vào bình (mỗi giây), bình nhận được sữa từ bình và tốc độ dòng chảy tối đa (lít / giây) chảy qua ống dẫn này.

**Output**: Là số nguyên duy nhất: lượng sữa tối đa chảy đến bình 1 (lít / giây)

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 5 2  20 1 50  20 1 30  20 2 5  40 2 30 | 90 |

**Explanation:**

Có 5 bình chứa và 2 đường ống được nâng cấp. Sơ đồ bình chứa và đường ống được mô tả trong hình dưới. Các con số trên ống là tốc độ dòng chảy tối đa. Những con số trên bình là lượng sữa được bơm vào mỗi giây. Sơ đồ bên trái là thiết kế ban đầu, còn bên phải là sau khi đã được nâng cấp.



Bằng cách nâng cấp các đường ống từ bình 2 và 4, bình 1 tại căn tin có thể nhận trực tiếp 20 lít/giây từ bình 3 và 20 lít/giây từ bể bình 2, và gián tiếp 50 lít/giây từ bình 2 (20 lít từ bình 4 và 30 lít từ bình 5). Tổng cộng, bình 1 nhận được 20 + 20 + 50 = 90 lít / giây.

**Solution:**

#include <iostream>

using namespace std;

const int MAXN = 205;

int N, K, P[MAXN], C[MAXN], F[MAXN];

int DP[MAXN][MAXN], DP2[MAXN][MAXN];

int main() {

cin >> N >> K;

for (int i = 2; i <= N; i++)

cin >> P[i] >> C[i] >> F[i];

for (int i = N; i >= 1; i--) {

int n = C[i];

for (int j = 0; j <= K; j++) {

for (int k = 0; k <= K - j; k++)

DP2[n][j + k] = max(DP2[n][j + k],

DP[n][k] + min(F[i], P[i] + DP[i][j]));

for (int k = 1; k <= K - j + 1; k++)

DP2[n][j + k] = max(DP2[n][j + k],

DP[n][k - 1] + P[i] + DP[i][j]);

}

for (int j = 0; j <= K; j++) {

DP[n][j] = max(DP[n][j], DP2[n][j]);

}

}

int ans = 0;

for (int i = 0; i <= N; i++) ans = max(ans, DP[1][i]);

cout << ans << endl;

return 0;

}

HCMIU Online Judge | 07

Sự Trả Thù Của Người Sith

IU Top Coder 2016

Timelimit: 2s

“Nếu ngươi không là đồng minh, thì ngươi là kẻ thù của ta”

“Chỉ duy nhất lời cam kết của Sith là có giá trị”

Obi-Wan Kenobi và Anakin Skywalker, từng là bạn bè thân thiết, đã phải quyết đấu trên hành tinh núi tủa trong hệ thống Mustafar. Cuộc chiến của họ sẽ diễn ra tại trụ sở của quân ly khai, cơ sở có buồn và hành lang. Hành lang thứ i kết nối 2 buồn là và , và có thể di chuyển theo 8 hướng. Mỗi 2 buồn đều có tối đa một hành lang kết nối. Đầu tiên, Obi-Wan và Anakin phải tìm thấy nhau, khi mà Anakin đang đợi ở một buồn ngẫu nhiên và Obi-Wan cũng đang chời ở một buồn ngẫu nhiên khác. Với mỗi địa điểm được chọn ngẫu nhiên như trên từ tập buồn cho trước, có tất cả trường hợp cặp buồn khác nhau.

Khi đã ở vị trí buồn đầu tiên, Obi-Wan và Anakin sẽ bắt đầu truy tìm lẫn nhau. Chính xác là mỗi phút, mỗi người sẽ di chuyển qua một hành lang kết nối buồn hiện tại, trừ khi buồn hiện tại không có hành lang kết nối nào, trong trường hợp đó họ sẽ đứng yên tại chính buồn đó. Nếu 2 người bọn họ có thể tìm thấy nhau tại một buồn trong thời gian cho trước, họ sẽ bắt đầu cuộc quyết đấu. Ngoài ra

, họ sẽ tiếp tục cuộc truy lùng mãi mãi (Jedi có cuộc sống rất lâu dài). Ghi chú rằng, với sức mạnh của Năng Lực, học có thể di chuyển qua hành lang một cách nhanh chóng trong vòng 1 phút, nên họ có thể gặp nhau trên hành lang, thậm chí là khi cả hai đang di chuyển theo chiều đối ngược nhau trên 1 hành lang.

Tính tỉ lệ Obi-Wan và Anakin có thể gặp nhau và có cuộc quyết đấu.

**Input**:

* + - * Dòng đầu tiên là 2 số và
      * M dòng tiếp theo là 2 số và , buồn kết nối của hành lang thứ .

**Output**: Là một số thực từ 0 đến 1, tỉ lệ mà Obi-Wan và Anakin có thể cí cuộc quyết đấu. Kết quả của bạn phải có độ sai lệch không vược quá .

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 6 7  2 3  2 4  2 5  3 4  3 5  4 5  6 1 | 0.500000000 |

**Explanation:**

Có 36 cặp buồn cho Obi-Wan và Anakin. Và nó có thể sảy ra cuộc quyết đấu cho 18 cặp buồn có xác suất cho một cuộc gặp thành công là 100% (bao gồm cả cặp (1,1) và (2,4)), và 18 cặp còn lại xác suất gặp là 0% (bao gồm cả cặp (1,6) và (5,6)).

**Solution:**

#include <iostream>

#include <vector>

using namespace std;

int N, M, a, b, c1, c2, ans = 0;

int col[46000] = { 0 };

vector<int> adj[46000];

bool bipartite;

void dfs(int n, int c) {

col[n] = c;

(c == 1 ? c1 : c2)++;

for (int j = 0; j < adj[n].size(); ++j) {

if (col[adj[n][j]] == 0) {

dfs(adj[n][j], c == 1 ? 2 : 1);

}

else if (col[adj[n][j]] == c) {

bipartite = false;

}

}

}

int main() {

cin >> N >> M;

for (int i = 0; i < M; ++i) {

cin >> a >> b;

adj[a].push\_back(b);

adj[b].push\_back(a);

}

for (int i = 1; i <= N; ++i) {

if (col[i] == 0) {

c1 = 0;

c2 = 0;

bipartite = true;

dfs(i, 1);

if (bipartite) {

ans += (c1\*c1) + (c2 \* c2);

}

else {

ans += (c1 + c2) \* (c1 + c2);

}

}

}

cout.precision(9);

cout << fixed << (double)ans / (N\*N) << endl;

return 0;

}

HCMIU Online Judge | 08

Rào Cản

IU Top Coder 2016

Timelimit: 2s

Đàn bò của nông dân John đang luyện tập cho 1 cuộc chạy vượt rào sắp diễn ra. Nông trại của John có cánh đồng và con đường , mỗi con đường được kí hiệu bởi 2 chỉ số và mô tả 1 đường đi 1 chiều từ đến . Ngoài ra, trên mỗi con đường, nông dân John đặt 1 chướng ngại vật độ cao .

Lũ bò sẽ tập chạy trong vòng , mỗi vòng chúng sẽ chạy từ đồng cỏ đến , khác ). Do chúng rất lười biếng, nên chúng sẽ chọn con đường mà độ cao của chướng ngại vật cao nhất cần phải vượt qua là thấp nhất. Nói cách khác, nếu gọi là kết quả của lượt chạy này, không có chướng ngại vật nào trên đường chạy này có độ cao lớn hơn .

Hãy giúp lũ bò chọn ra đường chạy tối ưu.

**Input**:

* Dòng đầu tiên chứa 3 số nguyên và .
* dòng tiếp theo, mỗi dòng chứa 3 số nguyên tương ứng
* dòng tiếp theo, mỗi dòng chứa 2 số nguyên và

**Output**: Đưa ra dòng, là độ cao của đường chạy tối ưu trong vòng chạy tương ứng, hoặc trong trường hợp không có đường đi từ đến

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 5 6 3 1 2 12 3 2 8 1 3 5 2 5 3 3 4 4 2 4 8 3 4 1 2 5 1 | 4  8 -1 |

**Solution:**

#include <stdio.h>

#include <vector>

#include <iostream>

#include <algorithm>

using namespace std;

#define long long long

#define f1(i,n) for (int i=1; i<=n; i++)

#define f0(i,n) for (int i=0; i<n; i++)

#define N 302

const int oo = 0x3c3c3c3c;

int n, m, q, c[N][N];

main(){

scanf("%d%d%d", &n, &m, &q);

f1(i,n) f1(j,n) c[i][j] = (i==j ? 0 : oo);

f1(i,m) {

int w, x, y;

scanf("%d%d%d", &x, &y, &w);

c[x][y]=w;

}

f1(k,n) f1(i,n) f1(j,n)

c[i][j] = min(c[i][j], max(c[i][k], c[k][j]));

while (q-->0) {

int x, y;

scanf("%d%d", &x, &y);

printf("%d\n", c[x][y]==oo ? -1 : c[x][y]);

}

}