

Problem A

最強のカードを見つけれ

Time Limit: 2 seconds

普通のトランプカードには 13 種の異なるランク、すなわちエース、キング、クイーン、ジャックと、10 から 2 までの数がある。ひと揃いのカードは各ランクが 4 枚ずつ、合計 52 枚である。多くのゲームでは、カードの強さはこのランクの順序通りで、エースが最強、2 が最弱である。

日本などいくつかの東アジアの地域で人気のあるゲーム、たとえば「大富豪」「争上游」「大老二」などでは、この強さの順序が異なっている。これらのゲームでは 13 種のランクのうちで 2 が最強で、次がエース、その後はキング、クイーンなどの通常のランク順で、3 が最も弱い。この一風変わった強さの順序を「大老二順」と呼ぶことにする。

あなたの仕事は、与えられた何枚かのカードの中から大老二順で一番強いカードのランクを見つけることである。

Input

入力は 1 個以上のテストケースからなる。各テストケースは次の形式で表される。

$$\begin{array}{l} n \\ c_1 c_2 \cdots c_n \end{array}$$

各テストケースは 2 行からなる。1 行目にはカードの枚数 n ($1 \leq n \leq 52$) が与えられる。2 行目の n 個の整数 c_1, c_2, \dots, c_n は 1 以上 13 以下で、与えられた n 枚のカードのランクを表す。ここで整数 2 から 10 はランク 2 から 10 を、1, 13, 12, 11 はそれぞれエース、キング、クイーン、ジャックを表す。同じランクのカードが最大 4 枚与えられることもある。

入力の終わりは、1 個のゼロだけからなる行で表される。テストケースの個数は 100 を超えない。

Output

各テストケースについて、与えられたカードのうち大老二順で最強のもののランクを 1 行に出力せよ。ここでもエース、キング、クイーン、ジャックについては 1, 13, 12, 11 で表し、ランク 2 から 10 はその数で表すこと。

サンプル入出力は [DOMjudge の Problemset ページ](#) からダウンロード可能である。

Sample Input 1

```
4
2 1 10 12
7
3 4 5 4 3 4 5
6
3 11 13 7 8 3
0
```

Sample Output 1

```
2
5
13
```

Problem B

自動販売機

Time Limit: 2 seconds

あなたの所属する研究所にはいくつかの部屋と、長さが 10^8 のまっすぐな廊下が一本だけある。各部屋には廊下に出る扉がひとつずつある。

この廊下に自動販売機を何台か設置することになった。研究員の利便性のため、どの扉からでも十分近くに自動販売機が少なくとも 1 台はあるように設置したい。扉の位置と、それぞれの扉から最寄りの自動販売機までの距離として許される上限が与えられるので、この条件を満たすために必要な自動販売機の最少台数を求めよ。

自動販売機は廊下のどこに置いてもよい。扉と同じ位置や、廊下の端に置いてもよい。この問題では廊下は幅を持たない線分だと見なし、扉や自動販売機はその線分上の点と見なす。

Input

入力は 1 個以上のテストケースからなる。各テストケースは次の形式で表される。

$$\begin{array}{l} n \ d \\ x_1 \ x_2 \ \cdots \ x_n \end{array}$$

テストケースは 2 行からなる。1 行目には扉の個数 n と、それぞれの扉から最寄りの自動販売機までの距離の上限を表す整数 d が与えられる ($1 \leq n \leq 100, 1 \leq d \leq 10^8$)。2 行目には扉の位置が与えられる。各 $i = 1, 2, \dots, n$ について、 x_i は、廊下の一方の端から i 番目の扉までの距離を表す整数である ($0 \leq x_1 < x_2 < \cdots < x_n \leq 10^8$)。

入力の終わりは、2 個のゼロだけからなる行で表される。テストケースの個数は 100 を超えない。

Output

各テストケースについて、必要となる最少の自動販売機の台数を 1 行に出力せよ。

サンプル入出力は [DOMjudge の Problemset ページ](#) からダウンロード可能である。

Sample Input 1

```
3 5
10 20 40
9 1
0 1 2 3 4 5 6 7 8
2 300
123 724
1 100000000
100000000
0 0
```

Sample Output 1

```
2
3
2
1
```



Sample Input 1 の最初のテストケースでは、たとえば 15 と 40 の位置に自動販売機を設置すればよい。

Problem C

残った水

Time Limit: 2 seconds

木材で水槽を作った。水槽は真上から見ると長方形で、縁の高さは均一だが、底は平らとは限らず、段々になっている。水槽の底は長手方向の一定長さごとに区切られており、ひとつの区画内の底は平らで水平だが、異なる区画の底の深さは異なっているかもしれない。

この水槽に縁までいっぱい水を溜めておいたのだが、いつの間にか両端の板が外れていて、水が流れ出てしまった。それでも底が深い部分には水が残っているかもしれない。1区画、深さ1あたりの水量を1として、残っている水の量を求めよ。

図 C.1 は水槽の状態を表す。左からそれぞれ、水が溜まっていない状態、満水の状態、両端の板が外れた状態を示している。

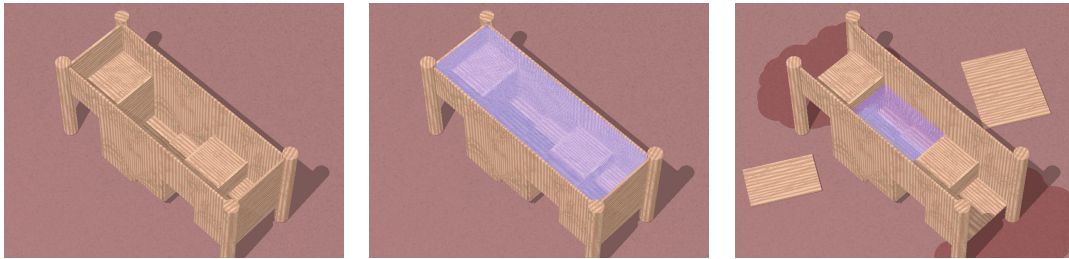


図 C.1. 水槽

Sample Input 1 の最初のテストケースには5つの区画があり、各区画の縁から底までの深さがそれぞれ3, 8, 7, 4, 6である。図 C.2 は、この水槽の断面を表しており、左図は満水の状態、右図は両端の板が外れて水が流れた状態を示している。2番目と3番目の連続する2区画は、両隣よりも深くなっているので、水が残り、その水面は4番目の区画の底と同じ高さになる。2番目の区画には $8 - 4 = 4$ の水が、3番目の区画には $7 - 4 = 3$ の水が残る。他の区画には水が残らない。したがって、残る水の量は $4 + 3 = 7$ である。

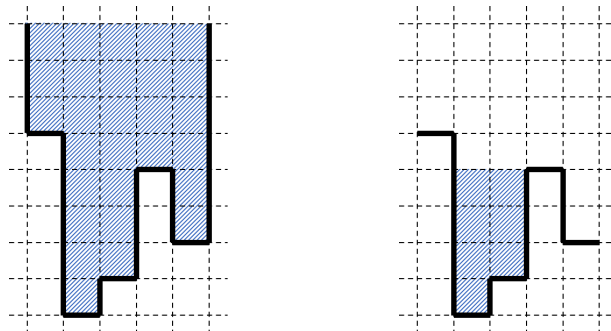


図 C.2. Sample Input 1 の最初のテストケースの断面図

Input

入力は1個以上のテストケースからなる。各テストケースは次の形式で表される。

$$\begin{array}{l} n \\ d_1 d_2 \cdots d_n \end{array}$$

テストケースは2行からなる。1行目には区画の個数 n が与えられる ($1 \leq n \leq 100$)。各 $k = 1, 2, \dots, n$ について、整数 d_k は、一方の端から k 番目の区画の、縁からの深さである ($1 \leq d_k \leq 1000$)。

入力の終わりは、1個のゼロだけからなる行で表される。テストケースの個数は100を超えない。

Output

各テストケースについて、残る水の総量を1行に出力しなさい。

サンプル入出力は [DOMjudge の Problemset ページ](#) からダウンロード可能である。

Sample Input 1

```
5
3 8 7 4 6
9
11 15 14 1 7 5 11 4 9
6
4 6 6 6 4 3
6
5 9 7 8 9 4
1
300
2
300 400
0
```

Sample Output 1

```
7
18
6
13
0
0
```

Problem D

回数数列

Time Limit: 2 seconds

正の整数 s を初項とする, 正整数の数列 a_1, a_2, \dots を以下で定義する.

- $a_1 = s$
- $a_{i+1} = |\{j \in \{1, 2, \dots, i\} \mid a_j = a_i\}|$ ($i \geq 1$)

すなわち, 第 $i+1$ 項 a_{i+1} は, 直前の項 a_i の値が第 1 項から第 i 項までの中に出現する回数である. たとえば $s = 3$ の場合, 数列の最初の 12 項は 3, 1, 1, 2, 1, 3, 2, 2, 3, 3, 4, 1 である.

正の整数 s と k が与えられるので, 数列の第 k 項である a_k の値を求めよ.

Input

入力は 1 個以上のテストケースからなる. 各テストケースは次の形式で表される.

s k

テストケースは 2 つの正の整数 s, k からなる ($1 \leq s \leq 10^9, 1 \leq k \leq 10^9$).

入力の終わりは, 2 個のゼロだけからなる行で表される. テストケースの個数は 100 を超えない.

Output

各テストケースについて, 数列の k 番目の項 a_k の値を 1 行に出力しなさい.

サンプル入出力は [DOMjudge の Problemset ページ](#) からダウンロード可能である.

Sample Input 1

Sample Output 1

3 1	3
3 2	1
3 3	1
3 4	2
3 5	1
6 100	12
100000000 100000000	50000000
123456789 987654321	5
31415926 535897932	16621598
0 0	

Problem E

買い物上手

Time Limit: 2 seconds

Icpca 王国では、コインとジェムの2種類の通貨が流通している。この国の店では、各商品の値札で指定された枚数のコインを支払うか、ジェム1個だけを支払うかのいずれかを行えば、商品を購入することができる。

あなたは Icpca 王国に観光に来ている。今日が帰国前の最終日であるため、土産物屋に行き、名産品である酒を購入することに決めた。今、その店の棚には n 本の酒のボトルが並べられている。各酒はそれぞれ別々の職人たちが手掛けたものであり、したがって価格も異なるかもしれない。あなたは今、十分な数のコインを持っているが、店独自のサービスを有効活用して、できるだけ安く n 本すべてを買いたい。

独自のサービスの内容は以下の通りである。各ボトルには特典として、いくつかのジェムが入った袋が付いていることがある。したがって、先に買ったボトルの特典のジェムを使って次のボトルを購入することで、コインを節約できるかもしれない。

ジェムを持っていない状態から始めて、 n 本すべてのボトルを買うのに必要なコインは最小何枚か？

図 E.1 は Sample Input 1 の最初のテストケースにおけるボトルの価格および各ボトルに付いているジェムを図示したものである。このケースでは1番目のボトルを400コインで購入すると、ジェム1つが手に入る。その後、3,4,2番目のボトルを順にジェムで購入すると、コインを支払わずに済む。

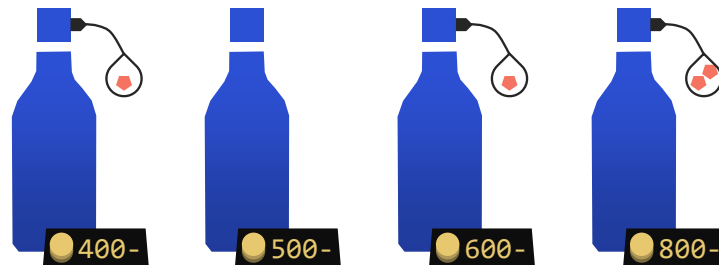


図 E.1. Sample Input 1 の最初のテストケース

Input

入力は1個以上のテストケースからなる。各テストケースは次の形式で表される。

$$\begin{array}{l} n \\ a_1 b_1 \\ a_2 b_2 \\ \vdots \\ a_n b_n \end{array}$$

1行目の整数 n ($1 \leq n \leq 10^5$) は土産物屋にある酒のボトルの本数を表す。

次の n 行の i 行目は i 番目のボトルの情報を表す ($i = 1, 2, \dots, n$). 整数 a_i ($1 \leq a_i \leq 10^9$) は i 番目のボトルの値札で指定されているコインの枚数を表し, 整数 b_i ($0 \leq b_i \leq n$) は i 番目のボトルに付いている袋に入っているジェムの個数を表す.

入力の終わりは, 1 個のゼロだけからなる行で表される. テストケースの個数は 2500 を超えない. すべてのテストケースにわたる n の総和は 10^5 を超えない.

Output

各テストケースについて, n 本すべてのボトルを購入するのに必要なコインの枚数の最小値を 1 行に出力せよ.

サンプル入出力は [DOMjudge の Problemset ページ](#) からダウンロード可能である.

Sample Input 1

Sample Output 1

4	400
400 1	6600
500 0	1747
600 1	
800 2	
5	
1540 0	
1430 0	
1320 0	
1210 0	
1100 0	
20	
861 0	
901 0	
955 1	
602 1	
882 1	
188 1	
817 0	
932 2	
669 0	
621 2	
276 0	
668 0	
825 1	
834 1	
341 2	
545 0	
218 0	
939 0	
179 1	
587 1	
0	

Problem F

地図アプリの高速化

Time Limit: 4 seconds

あなたは、Icpca 市を対象とした地図アプリの高速化を行っている。

Icpca 市の市域は正方形であり、南北方向に n 行、東西方向に n 列の均等な正方形の区画に分割されている。北から i 行目、西から j 列目 ($1 \leq i \leq n, 1 \leq j \leq n$) の区画を (i, j) と表す。たとえば北西隅の区画は $(1, 1)$ である。市には、1つの区画からなる中央駅と、いくつかの建物が存在する。各建物は1つ以上の区画を含む長方形領域を占める。

あなたの地図アプリでは、指定された2つの区画の間を徒歩で移動する際の最短所要時間を計算することができる。計算の上では、東西南北に隣接する区画に徒歩で1分かけて移動することができ、斜めには移動できないと仮定する。また、建物が占有する区画には入れず、Icpca 市の外には出られない。一方、中央駅が占有する区画には入れると仮定する。

あなたはテスト運用中、ユーザーが中央駅からの所要時間を頻繁に問い合わせていることに気づいた。そこで、中央駅から出発し、指定された区画に至るまでの最短所要時間を求める多数の問い合わせを、高速に処理するプログラムを書いてほしい。

図 F.1 は Sample Input 1 の最初のテストケースにおける Icpca 市の様子、および中央駅から区画 $(1, 7)$ までの最短経路を図示したものである。中央駅は緑色で、区画 $(1, 7)$ は水色の縞模様で、建物は濃い灰色で示されている。最短所要時間は17分となる。

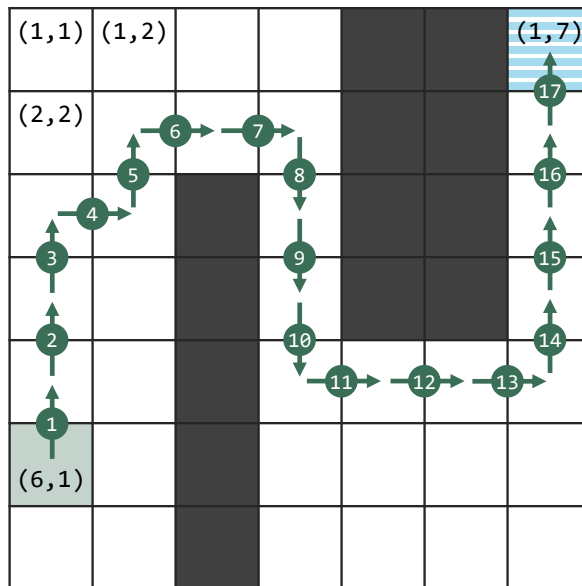


図 F.1. Sample Input 1 の最初のテストケース

Input

入力は1個以上のテストケースからなる。各テストケースは次の形式で表される。

```
n u v
m
a1 b1 c1 d1
a2 b2 c2 d2
⋮
am bm cm dm
q
s1 t1
s2 t2
⋮
sq tq
```

1行目は3つの整数 n, u, v ($2 \leq n \leq 10^9, 1 \leq u \leq n, 1 \leq v \leq n$) からなり、Icpca 市の各辺の区画の数が n 、中央駅が区画 (u, v) にあることを表す。

2行目は整数 m ($1 \leq m \leq 200$) からなり、市内にある建物の数を表す。続く m 行には各建物の占める範囲に関する情報が入っている。その中の i 行目 ($1 \leq i \leq m$) は4つの整数 a_i, b_i, c_i, d_i ($1 \leq a_i \leq b_i \leq n, 1 \leq c_i \leq d_i \leq n$) からなり、 i 番目の建物が $a_i \leq x \leq b_i$ および $c_i \leq y \leq d_i$ を満たす区画 (x, y) すべてを占めていることを示す。ここで、どの区画も複数の建物に占有されていないこと、および中央駅の区画がどの建物にも占有されていないことが保証される。

その次の行には、問い合わせの数 q ($1 \leq q \leq 2 \times 10^5$) が与えられる。続く q 行には問い合わせの情報が入っている。その中の j 行目 ($1 \leq j \leq q$) は2つの整数 s_j, t_j ($1 \leq s_j \leq n, 1 \leq t_j \leq n$) からなり、中央駅から区画 (s_j, t_j) に移動するための最短所要時間の問い合わせを表す。ここで、区画 (s_j, t_j) はどの建物にも占有されていないことが保証される。

入力の終わりは、3個のゼロだけからなる行で表される。テストケースの個数は200を超えない。すべてのテストケースにわたる m の総和は200を超えない。また、すべてのテストケースにわたる q の総和は 2×10^5 を超えない。

Output

各テストケースについて、 q 行出力せよ。その j 行目 ($1 \leq j \leq q$) には、 j 番目の問い合わせの答えを分単位で出力せよ。ただし、指定された区画に到達できない場合は、答えとして no を出力せよ。

サンプル入出力は [DOMjudge の Problemset ページ](#) からダウンロード可能である。

Sample Input 1

Sample Output 1

```

7 6 1
2
3 7 3 3
1 4 5 6
4
1 7
5 5
6 2
6 1
10 1 1
4
5 6 4 4
5 6 7 7
4 4 5 6
7 7 5 6
3
10 10
3 3
5 6
2 1 1
2
1 1 2 2
2 2 1 1
1
2 2
9 1 1
4
2 2 1 8
4 4 2 9
6 6 1 8
8 8 2 9
3
1 5
5 5
9 9
10000000 7777777 123456
1
1000001 9000000 1000001 9000000
3
2525252 9876543
10000000 5252525
123456 123456
0 0 0

```

```

17
11
1
0
18
4
no
no
4
24
48
17450060
7351292
7654321

```

Problem G

衝突回避

Time Limit: 3 seconds

縦 n マス、横 m マスからなるグリッド状の盤面がある。上から i 行目、左から j 列目のマスを (i, j) と表す。盤面の四隅以外のマスには障害物が置かれているかもしれない。

盤上に白い駒と黒い駒が 1 つずつある。はじめ、白い駒は左上隅のマス $(1, 1)$ に、黒い駒は左下隅のマス $(n, 1)$ に置かれている。

あなたは 2 つの駒を交互に動かす。最初に動かすのは白い駒である。白い駒は 1 回の移動で右または下に隣接するマスへ、黒い駒は右または上に隣接するマスへ動かす。ここで、2 つのマスが辺を共有しているとき、それらのマスは隣接しているとみなす。ただし、障害物の置かれたマスへ動かすことはできない。さらに、もう一方の駒のあるマスに駒を動かすことはできない。

2 つの駒をそれぞれ $n + m - 2$ 回動かす。白い駒を右下隅のマス (n, m) に、黒い駒を右上隅のマス $(1, m)$ に到達させたい。そのような移動経路の組の総数を 998 244 353 で割った余りを求めよ。

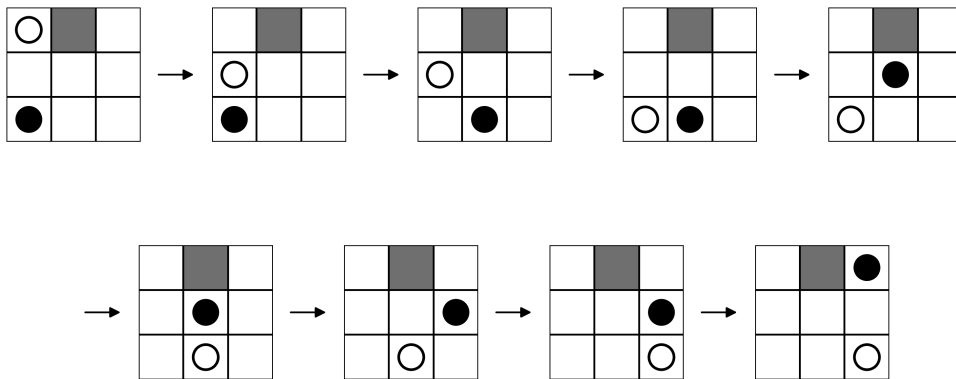


図 G.1. Sample Input 1 の最初のテストケースにおける移動の例

Input

入力は 1 個以上のテストケースからなる。各テストケースは次の形式で表される。

```

n m
s1,1s1,2⋯s1,m
⋮
sn,1sn,2⋯sn,m

```

整数 n と m は、それぞれグリッドの行数と列数を表す ($2 \leq n \leq 10^6, 2 \leq m \leq 10^6, n \times m \leq 2 \times 10^6$)。続く n 行には障害物の有無を表す '#' と '.' からなる長さ m の文字列が与えられる。マス (i, j) には、 $s_{i,j}$ が '#' ならば障害物が置かれており、'.' ならば障害物は置かれていない。 $s_{1,1}, s_{1,m}, s_{n,1}, s_{n,m}$ は必ず '.' である。

入力の終わりは、2 個のゼロだけからなる行で表される。テストケースの個数は 200 を超えない。すべてのテストケースにわたる $n \times m$ の総和は 2×10^6 を超えない。

Output

各テストケースについて、移動経路の組の総数を 998 244 353 で割った余りを 1 行に出力せよ。

サンプル入出力は [DOMjudge の Problemset ページ](#) からダウンロード可能である。

Sample Input 1

```
3 3
.#.
...
...
4 7
.....
###.###
###.###
.....
2 2
..
..
11 27
.....
.....
.....
...###...##...###...##...
...#...#...#...#...#...#...#...
...#...#...###...#...
...#...#...#...#...#...#...
...###...##...#...##...
.....
.....
.....
0 0
```

Sample Output 1

```
3
0
1
40019649
```

Problem H

浮き輪の色分け

Time Limit: 2 seconds

海開きの準備のため、色とりどりの浮き輪を整理している。地面に垂直に立てられた n 本の棒があり、すべての棒にはそれぞれ m 個の浮き輪が積み重なるように通されている。各浮き輪は 1 から n までのいずれかの色であり、どの色の浮き輪も少なくとも 1 個はある。また、浮き輪を一時的に移しておくための広い置き場がある。はじめ、この置き場に浮き輪はない。

あなたは浮き輪を移動する操作を何度でも繰り返すことができる。1 回の操作は以下のいずれかである。

- ある棒に通されている一番上の浮き輪を 1 個取り、別の棒の一番上に通す。
- ある棒に通されている一番上の浮き輪を 1 個取り、置き場に移す。
- 置き場にある任意の浮き輪 1 個と棒のうちの 1 本を選び、浮き輪をその棒の一番上に通す。

なお、浮き輪はみな同じ大きさなので、棒に通した浮き輪の順序が入れ替わることはない。

最終的に、それぞれの棒に集められている浮き輪がすべて同じ色となっていて、置き場には浮き輪が残っていない状態にしたい。どの棒にどの色の浮き輪を集めるかは自由である。この状態にするために必要な最小の操作回数を求めよ。

Input

入力は 1 個以上のテストケースからなる。各テストケースは次の形式で表される。

```
n m
c1,1 c1,2 ⋯ c1,m
c2,1 c2,2 ⋯ c2,m
⋮
cn,1 cn,2 ⋯ cn,m
```

n は棒の本数を表す整数であり、 $2 \leq n \leq 10$ を満たす。 m は各棒にはじめに通されている浮き輪の個数を表す整数であり、 $1 \leq m \leq 3 \times 10^4$ を満たす。

各 i と j ($1 \leq i \leq n, 1 \leq j \leq m$) について、整数 $c_{i,j}$ は、はじめに棒 i の下から j 番目に通されている浮き輪の色を示す番号であり、 $1 \leq c_{i,j} \leq n$ を満たす。特に、はじめに棒 i の一番上に通されている浮き輪の色は $c_{i,m}$ である。すべての色 $1, 2, \dots, n$ について、その色の浮き輪がテストケース中に少なくとも 1 個は存在する。

入力の終わりは、2 個のゼロだけからなる行で表される。すべてのテストケースにわたる n の総和は 10 を超えない。

Output

各テストケースについて、必要な操作回数の最小値を 1 行に出力せよ。

サンプル入出力は DOMjudge の Problemset ページからダウンロード可能である。

Sample Input 1

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

Sample Output 1

```
8
3
6
```

Sample Input 1 の最初のテストケースでは、まず棒 1 の一番上から 1 個、棒 2 の一番上から 2 個の色 2 の浮き輪を置き場に移す。次に、棒 1 の一番上から 2 個の色 1 の浮き輪を棒 2 の一番上に通す。最後に、置き場にある 3 個の色 2 の浮き輪を棒 1 の一番上に通す。これにより、棒 1 には色 2 の浮き輪だけが、棒 2 には色 1 の浮き輪だけが通された状態になる。操作回数は 8 回であり、これが最小である。

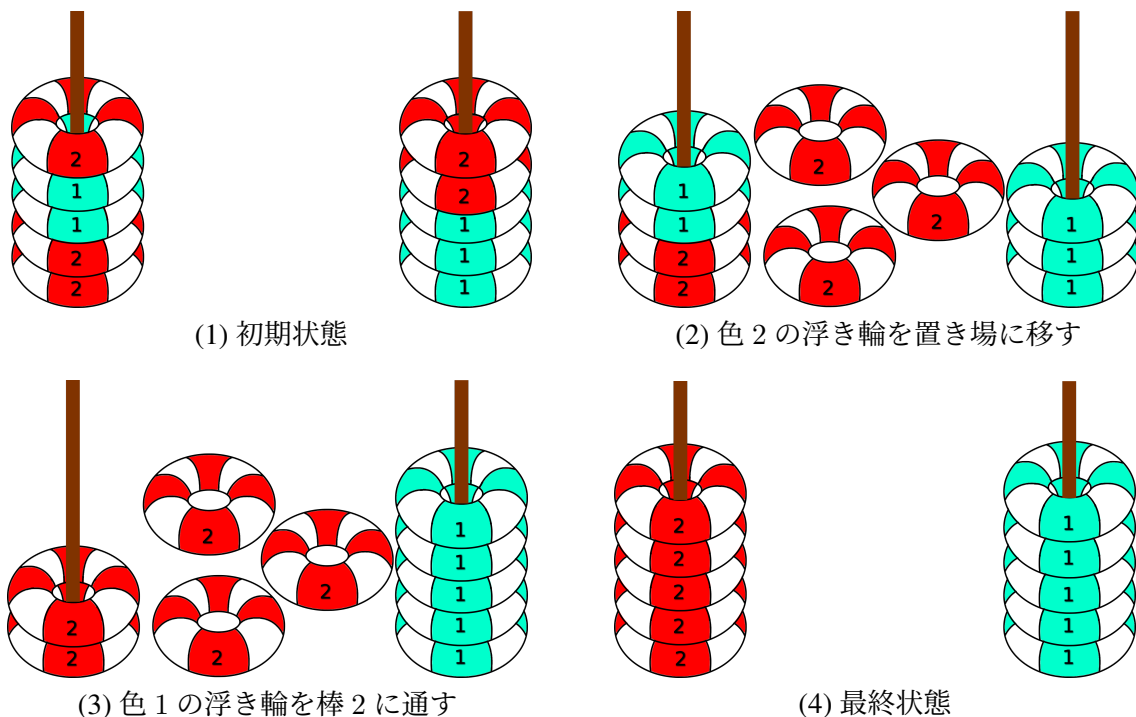


図 H.1. Sample Input 1 の最初のテストケースにおける最適な操作手順の例

Problem I

制限速度

Time Limit: 4 seconds

Icpca 王国には、一方通行の高速道路がある。この高速道路は始点から終点まで 1 km ごとの区間に分けられていて、それぞれの区間には制限速度が定められている。

あなたは車両を運転し、この高速道路の始点から終点に向かいたい。車両の速度は、任意のタイミングで瞬時に変更できる。各変更では、任意の**非負整数** v を選び、速度を v km/h にする。変更前の速度を v' km/h とすると、この変更には $|v - v'|$ のコストがかかる。始点を出る前の車両の速度は 0 km/h であり、また、車両が終点に到着した瞬間に、0 km/h にしなければならない。

始点から終点までに発生するコストの合計値が与えられた上限値を超えず、かつ、すべての区間で制限速度を超えないように走行するとき、始点から終点までの最短所要時間を求めよ。

Input

入力は 1 個以上のテストケースからなる。各テストケースは次の形式で表される。

$$\begin{array}{l} n f \\ a_1 a_2 \cdots a_n \end{array}$$

テストケースは 2 行からなる。1 行目には、高速道路の区間の個数 n と、コストの合計値の上限を表す整数 f が与えられる ($1 \leq n \leq 2 \times 10^5, 2 \leq f \leq 10^{10}$)。ここで、 f は偶数である。2 行目には、各区間の制限速度を表す整数 a_1, a_2, \dots, a_n が与えられる。 $i = 1, 2, \dots, n$ について、始点から i 番目の区間における制限速度は a_i km/h である ($1 \leq a_i \leq 10^5$)。

入力の終わりは、2 個のゼロだけからなる行で表される。テストケースの個数は 10^4 を超えない。すべてのテストケースにわたる n の総和は 2×10^5 を超えない。

Output

各テストケースについて、始点から終点までに必要な最短時間を、時間単位で 1 行に出力せよ。出力の絶対誤差または相対誤差が 10^{-4} 以下であれば正答と見なされる。

サンプル入出力は [DOMjudge の Problemset ページ](#) からダウンロード可能である。

Sample Input 1

Sample Output 1

5 120	0.1050000000
120 100 40 100 120	0.3333333333
5 100	0.4776190476
10 20 30 20 10	2.5000000000
10 160	3.0000000000
30 10 40 10 50 90 20 60 50 30	1.1166666667
3 4	5.3285714286
2 1 2	5.4968253968
3 2	
2 1 2	
5 20	
7 3 8 4 9	
15 14	
1 2 3 4 5 6 7 8 7 6 5 4 3 2 1	
15 60	
6 2 3 5 1 7 2 1 5 9 7 5 3 2 6	
0 0	

Sample Input 1 の最初のテストケースでは、以下のように車両の速度を変更すると、最短所要時間を達成できる。

1. 始点で車両の速度を 50 km/h に変更する。 $|50 - 0| = 50$ のコストが発生する。
2. 始点から 1, 2 番目の区間を 50 km/h で通過する。これには $2/50 = 1/25$ 時間かかる。
3. 2 番目の区間を通り抜けた直後に速度を 40 km/h に変更する。 $|40 - 50| = 10$ のコストが発生する。
4. 3 番目の区間を 40 km/h で通過する。これには $1/40$ 時間かかる。
5. 3 番目の区間を通り抜けた直後に速度を 50 km/h に変更する。 $|50 - 40| = 10$ のコストが発生する。
6. 4, 5 番目の区間を 50 km/h で通過する。これには $2/50 = 1/25$ 時間かかる。
7. 終点到着した時点で速度を 0 km/h に変更する。 $|0 - 50| = 50$ のコストが発生する。

コストの合計値は $50 + 10 + 10 + 50 = 120$ である。また、所要時間は $1/25 + 1/40 + 1/25 = 0.105$ 時間である。

Problem J

最大の倍率

Time Limit: 2 seconds

xy 平面上の凸多角形 P, Q が与えられる。以下の条件を満たす実数 s の最大値を求めよ。

条件: Q の各点の x, y 座標を s 倍して得られる多角形を Q' とする。 Q' を適切に平行移動することで、 Q' の周上および内部の点をすべて、 P の周上または内部に含めることができる。

上記の条件で Q' に対して行える操作は平行移動のみであり、回転や反転は行えないことに留意されたい。

図 J.1 (a)–(c) は Sample Input 1 の 3 つのテストケースを図示したものである。青い多角形が P 、赤い多角形が Q に対応する。最初のテストケースにおいて、 Q を 0.5 倍した多角形を Q' とし、さらに Q' を x 方向に 5、 y 方向に 5 だけ平行移動させてできる多角形を図 J.1 (d) の赤い多角形に示す。この多角形の周上および内部の点はすべて、 P の周上または内部に含まれている。条件を満たす s の最大値は 0.5 である。 s が 0.5 より大きいときは、 Q を s 倍した多角形をどう平行移動させても P の周上または内部に収めることができない。

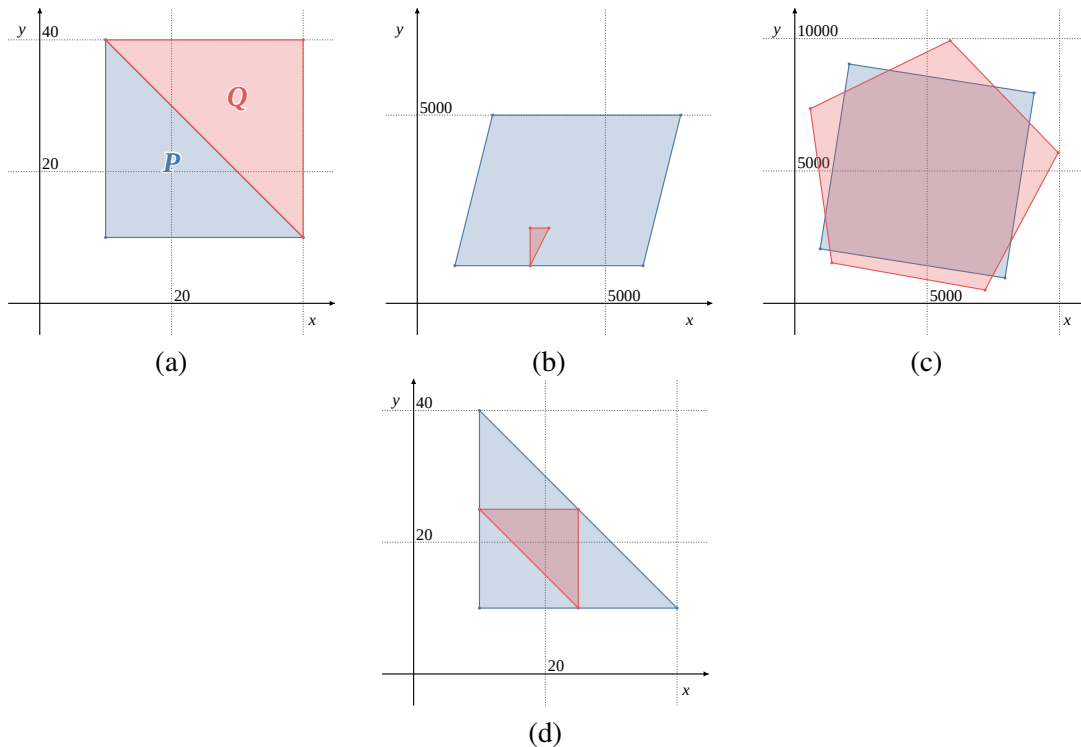


図 J.1. Sample Input 1 の図示

Input

入力は 1 個以上のテストケースからなる。各テストケースは次の形式で表される。

$$\begin{array}{l} n \ m \\ x_1 \ y_1 \\ \vdots \\ x_n \ y_n \\ x'_1 \ y'_1 \\ \vdots \\ x'_m \ y'_m \end{array}$$

テストケースの 1 行目には、凸多角形 P の頂点数を表す整数 n と、 Q の頂点数を表す整数 m が与えられる ($3 \leq n \leq 500, 3 \leq m \leq 500$)。続く n 行には、 P の頂点座標 $(x_1, y_1), \dots, (x_n, y_n)$ が反時計回り順で与えられる。同様に、次の m 行には、 Q の頂点座標 $(x'_1, y'_1), \dots, (x'_m, y'_m)$ が反時計回り順で与えられる。それぞれの座標は 0 以上 10^4 以下の整数である。

多角形 P, Q は単純多角形であり、各頂点の内角の大きさは 180 度未満であることが保証される。

入力の終わりは、2 個のゼロだけからなる行で表される。テストケースの個数は 100 を超えない。すべてのテストケースにわたる n の合計は 500 を超えない。 m についても同様である。

Output

各テストケースについて、実数 s の最大値を 1 行に出力せよ。出力の絶対誤差または相対誤差が 10^{-4} 以下であれば正答と見なされる。

サンプル入出力は [DOMjudge の Problemset ページ](#) からダウンロード可能である。

**Sample Input 1**

```
3 3
10 10
40 10
10 40
40 40
10 40
40 10
4 3
1000 1000
6000 1000
7000 5000
2000 5000
3000 1000
3500 2000
3000 2000
4 5
2056 9041
959 2056
7944 959
9041 7944
587 7351
1400 1530
7188 504
9952 5692
5872 9923
0 0
```

Sample Output 1

```
0.5
4
0.743571717879
```