算法设计与应用基础：Homework No.4

16337341 朱志儒

1. Preparing Olympiad

You have n problems. You have estimated the difficulty of the i-th one as integer ci. Now you want to prepare a problemset for a contest, using some of the problems you've made.

A problemset for the contest must consist of at least two problems. You think that the total difficulty of the problems of the contest must be at least l and at most r. Also, you think that the difference between difficulties of the easiest and the hardest of the chosen problems must be at least x.

Find the number of ways to choose a problemset for the contest.

Input

The first line contains four integers n, l, r, x ($1 \le n \le 15$, $1 \le l \le r \le 109$, $1 \le x \le 106$) — the number of problems you have, the minimum and maximum value of total difficulty of the problemset and the minimum difference in difficulty between the hardest problem in the pack and the easiest one, respectively.

The second line contains n integers c1, c2, ..., cn ($1 \le ci \le 106$) — the difficulty of each problem.

Output

Print the number of ways to choose a suitable problemset for the contest.

解题思路：

首先使用位运算得出整个 ci 集合的所有子集，再判断哪些子集符合条件，最后计算这些子集的数量即可。

代码如下：

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

int main() {
    long n, l, x, r;
    int count = 0;
    vector<long> problems;
```

```
cin >> n >> l >> r >> x;
for (long i = 0; i < n; ++i) {
        long tmp;
        cin >> tmp;
        problems.push_back(tmp);}
sort(problems.begin(), problems.end());
for (int i = 0; i < pow(2, n); ++i) {
        long sum = 0, mini = LONG_MAX, maxi = -1;
        for (int j = 0; j < problems.size(); ++j) {
                if (i & 1 << j) {
                        sum += problems[j];
                        mini = min(problems[j], mini);
                        maxi = max(problems[j], maxi);}}
        if (sum >= l && sum <= r && maxi - mini >= x) count++;}
cout << count;
return 0;}
```

结果：

40025504	2018-07-06 12:23:44	zhuzhr3	550B - Preparing Olympiad	GNU C++17	Accepted	31 ms	300 KB

2. Little Pony and Summer Sun Celebration

Twilight Sparkle learnt that the evil Nightmare Moon would return during the upcoming Summer Sun Celebration after one thousand years of imprisonment on the moon. She tried to warn her mentor Princess Celestia, but the princess ignored her and sent her to Ponyville to check on the preparations for the celebration. Twilight Sparkle wanted to track the path of Nightmare Moon. Unfortunately, she didn't know the exact path. What she knew is the parity of the number of times that each place Nightmare Moon visited. Can you help Twilight Sparkle to restore any path that is consistent with this information?

Ponyville can be represented as an undirected graph (vertices are places, edges are roads between places) without self-loops and multi-edges. The path can start and end at any place (also it can be empty). Each place can be visited multiple times. The path must not visit more than 4n places.

Input

The first line contains two integers n and m (2 n 105; 0 m 105) — the number of 2 places and the number of roads in Ponyville. Each of the following m lines contains two integers ui, vi (1 ui, vi n; ui /= vi), these integers describe a road between places ui and vi. The next line contains n integers: x1, x2, ..., xn (0 xi 1) — the parity of the number of times that each place must be visited. If xi = 0, then the i-th place must be visited even number of times, else it must be visited odd number of times.

Output

Output the number of visited places k in the first line (0 k 4n). Then output k integers — the numbers of places in the order of path. If xi = 0, then the i-th place must appear in the path even number of times, else i-th place must appear in the path odd number of times. Note, that given road system has no self-loops, therefore any two neighbouring places in the path must be distinct. If there is no required path, output -1. If there multiple possible paths, you can output any of them.

解题思路：

从任意一个奇结点开始，使用深度优先搜索遍历没有经过的结点，回溯时判断之前走过的点是否满足条件，若不满足则再走一次。

代码如下：

```
#include <iostream>
#include <vector>
using namespace std;

int ans = 0, s = 0;
int n, m, c[100005], v[100005] = { 0 }, rec[100005 * 4];
vector<int> g[100005];

void set(int u) {
	c[u] ^= 1;
	rec[ans++] = u;}

void dfs(int u) {
	set(u);
	v[u] = 1;
	for (int i = 0; i < g[u].size(); ++i) {
		if (v[g[u][i]]) continue;
		dfs(g[u][i]);
		set(u);
		if (c[g[u][i]]) {
			set(g[u][i]);
			set(u);}}}

bool judge() {
	for (int i = 1; i <= n; ++i)
		if (c[i]) return false;
	return true;}

int main() {
```

```
cin >> n >> m;
for (int i = 0; i < m; ++i) {
        int a, b;
        cin >> a >> b;
        g[a].push_back(b);
        g[b].push_back(a);}
for (int i = 1; i <= n; ++i) {
        cin >> c[i];
        if (c[i]) s = i;}
dfs(s);
if (c[s]) {
        c[s] = 0;
        ans--;}
if (judge()) {
        cout << ans << endl;
        if (ans) {
                cout << rec[0];
                for (int i = 1; i < ans; ++i)
                        cout << ' ' << rec[i];
                cout << endl;}}
else
        cout << -1 << endl;
return 0;}
```

结果：

40169920	2018-07-10 14:56:52	zhuzhr3	453C - Little Pony and Summer Sun Celebration	GNU C++17	Accepted	155 ms	5600 KB

3. Freelancer's Dreams

Mikhail the Freelancer dreams of two things: to become a cool programmer and to buy a flat in Moscow. To become a cool programmer, he needs at least p experience points, and a desired flat in Moscow costs q dollars. Mikhail is determined to follow his dreams and registered at a freelance site.

He has suggestions to work on n distinct projects. Mikhail has already evaluated that the participation in the i-th project will increase his experience by ai per day and bring bi dollars per day. As freelance work implies flexible working hours, Mikhail is free to stop working on one project at any time and start working on another project. Doing so, he receives the respective share of experience and money. Mikhail is only trying to become a cool programmer, so he is able to work only on one project at any moment of time.

Find the real value, equal to the minimum number of days Mikhail needs to make his dream come true.

For example, suppose Mikhail is suggested to work on three projects and $a1 = 6$, $b1 = 2$, $a2 = 1$, $b2 = 3$, $a3 = 2$, $b3 = 6$. Also, $p = 20$ and $q = 20$. In order to achieve his

aims Mikhail has to work for 2.5 days on both first and third projects. Indeed, $a1 \cdot 2.5 + a2 \cdot 0 + a3 \cdot 2.5 = 6 \cdot 2.5 + 1 \cdot 0 + 2 \cdot 2.5 = 20$ and $b1 \cdot 2.5 + b2 \cdot 0 + b3 \cdot 2.5 = 2 \cdot 2.5 + 3 \cdot 0 + 6 \cdot 2.5 = 20$.

Input

The first line of the input contains three integers n, p and q ($1 \le n \le 100\,000$, $1 \le p, q \le 1\,000\,000$) — the number of projects and the required number of experience and money.

Each of the next n lines contains two integers ai and bi ($1 \le ai, bi \le 1\,000\,000$) — the daily increase in experience and daily income for working on the i-th project.

Output

Print a real value — the minimum number of days Mikhail needs to get the required amount of experience and money. Your answer will be considered correct if its absolute or relative error does not exceed 10 - 6.

Namely: let's assume that your answer is a, and the answer of the jury is b. The checker program will consider your answer correct, if $\frac{|a-b|}{\max(1,b)} \le 10^{-6}$.

解题思路：

设每个项目分得的天数分别为 x1, x2, …，由题意可得：sum(ai * xi) >= p, sum(bi * xi) >= q，需要求得 sum(xi)的最小值，可由线性规划中的单纯形法得出答案。然而实现起来很麻烦，所以我在网上搜索了一下，学习了一个更优的解法：

根据对偶性，原问题的对偶问题就是求 max(p*y1 + q*y2)， 满足 ai * y1 + bi * y2 <= 1。于是可以直接用三分求最大值。

代码如下：

```
#include <iostream>
#include <algorithm>
#include <iomanip>
using namespace std;

int a[100010], b[100010];
int n, p, q;

double find(double x) {
    double y = 1;
    for (int i = 0; i < n; ++i)
        y = min(y, (1.0 - x * b[i]) / a[i]);
```

```
    return y * p + x * q;}

int main() {
    cin >> n >> p >> q;
    for (int i = 0; i < n; ++i)
        cin >> a[i] >> b[i];
    double l = 0, r = 1.0 / (*max_element(b, b + n));
    for (int i = 0; i < 50; i++) {
        double ll = (r + l) * 0.5, rr = (ll + r) * 0.5;
        if (find(ll) > find(rr)) r = rr;
        else l = ll;}
    cout << fixed << setprecision(10) << find((l + r) * 0.5);
    return 0;}
```

结果：

40106506	2018-07-09 11:50:50	zhuzhr3	606E - Freelancer's Dreams	GNU C++17	Accepted	358 ms	1100 KB

4. Red and Black Tree

You have a weighted tree, consisting of n vertices. Each vertex is either painted black or is painted red. A red and black tree is called beautiful, if for any its vertex we can find a black vertex at distance at most x.

The distance between two nodes is the shortest path between them.

You have a red and black tree. Your task is to make it beautiful in the minimum number of color swap operations. In one color swap operation, you can choose two vertices of different colors and paint each of them the other color. In other words, if you choose a red vertex p and a black vertex q, then in one operation you are allowed to paint p black and paint q red.

Print the minimum number of required actions.

Input

The first line contains two integers n and x ($2 \le n \le 500$; $1 \le x \le 109$). The next line contains n integers, each of them is either a zero or one. If the i-th number equals 1, then vertex i of the tree is black, otherwise vertex i is red. Next n - 1 lines contain the tree edges. The j-th line contains integers uj vj wj ($1 \le uj, vj \le n$; $uj \ne vj$; $1 \le wj \le 109$) which means that the tree has an edge of weight wj between vertices vj and uj.

Assume that the tree vertices are numbered from 1 to n.

Output

Print a single integer — the minimum number of required swap operations.

If it is impossible to get a beautiful tree at any number of operations, print -1.

解题思路：

代码如下：

结果：