제빵왕

a와 b는 라이벌 제빵사입니다. 둘은 빵 만들기로 시합을 하기로 했습니다. 두 제빵사는 같은 빵을 만들어도 만드는 레시피가 달라 사용하는 밀가루의 양이 다를 수 있습니다. 여기서 당신은 심판이고, 두 라이벌 제빵사가 치룰 시합을 만듭니다. 시합은 a와 b가 빵을 만드는 것입니다. 같은 종류의 빵을 여러개 만들거나, 여러 종류의 빵을 특정 개수씩 만드는 시합을 의미합니다.

아래의 표에 빵의 종류와 각 종류 별 a, b가 사용하는 밀가루가 제시되어 있을 때, 시합의 예를 들면, a와 b가 팥빵 1개, 메론빵 2개, 소보로빵 1개와 같이 3가지의 종류의 빵을 총 4개를 만드는 시합을 예로 들 수 있습니다.

이 과정에서, a가 사용할 총 밀가루는 5(팥빵 1개) + 23*2(메론빵 2개) + 6(소보로빵 1개) = 57g가 되고, b가 사용할 총 밀가루는 3(팥빵 1개) + 23*2(메론빵 2개) + 7(소보로빵 1개) = 56g이 됩니다. a가 사용한 밀가루가 57g, b가 사용한 밀가루가 56g이므로, 이 시합에서는 b가 승리하게 됩니다.

하지만 당신은 이 시합의 심판으로서, 둘의 시합이 무승부로 끝이 났으면 좋겠다고 생각이 듭니다. 서로의 사이를 생각해서 말이죠.. 그렇다면 a와 b가 사용할 밀가루가 같아지게끔 시합을 구성했을 때, a와 b가 사용할 밀가루의 최솟값을 구해주세요.

예를 들어 다음 표와 같이 3개의 빵 종류가 있을 시 메론빵 1개를 만드는 시합을 만든다면, 종목을 선택하면 a와 b는 둘 다 23g로 같은 양의 밀가루를 사용하므로 무승부를 유도할 수 있습니다. 또 다른 시합으로는 팥빵 1개와 소보로빵 2개를 만드는 시합을 만든다면, a가 사용할 밀가루는 5(팥빵 1개) + 6*2(소보로빵 2개) = 17g, b가 사용할 밀가루는 3(팥빵 1개) + 7*2(소보로빵 2개) = 17g으로 a와 b가 같은 양의 밀가루를 사용하므로 무승부를 유도할 수 있습니다.

첫 번째 예제에서 23g을 사용하며 무승부를 유도하는 것 보다 두 번째 예제에서 17g을 사용하며 무승부를 유도하는 것이 더 작고, 아래의 예제에서 가장 최소의 밀가루로 무승부를 유도할 수 있는 시합이 됩니다.

빵 종류	a가 사용하는 밀가루 (g)	b가 사용하는 밀가루 (g)
팥빵	5 g	3 g
메론빵	23 g	23 g
소보로빵	6 g	7 g

입력 설명

위의 표의 예시와 같이 3종류의 입력도 들어갈 수 있고, 더 적거나 많은 종류의 빵이 입력으로 사용될 수 있습니다. 마찬가지로 빵의 종류가 주어졌을 때, a와 b가 사용할 밀가루도 각각 주어지게 됩니다. 입력의 첫 번째 줄에는 빵 종류의 수 n(1<=n<=500)이 주어집니다. 그 후 n 줄에 각 두 개의 숫자로 a와 b가 빵을 만드는데 사용하는 밀가루의 양이 정수로 주어집니다. 밀가루의 양은 g단위이며, 1이상 200이하의 정수입니다.

출력 설명
무승부를 유도하는 최적의 시합을 구성했을 때, 두 제빵사가 사용하는 최소 밀가루의 양을 g 단위로 출력합니다. 만약 같은 양의 밀가루를 사용할 수 없다면 -1을 출력합니다.

입력 예제1	입력 예제2	입력 예제3
3 5 3 23 23 6 7	2 2 1 3 7	3 23 78 54 64 22 53
출력 예제1	출력 예제2	출력 예제3
17	11	-1

예제1 : 입력의 두 번째 줄(5 3)을 1번, 네 번째 줄(6 7)을 두 번 선택하면 a는 5+6+6=17 g의 밀가루를 사용하고, b는 3+7+7=17 g의 밀가루를 사용하게 되어 a와 b는 같은 양의 밀가루를 사용하며 최소한의 밀가루를 사용 할 수 있습니다. 그러므로 출력은 17입니다.

예제2 : 입력의 두 번째 줄(2 1)을 4번, 세 번째 줄(3 7)을 한 번 선택하면 a는 2+2+2+2+3=11 g의 밀가루를 사용하고, b는 1+1+1+1+7=11 g의 밀가루를 사용하게 되어 a와 b는 같은 양의 밀가루를 사용하며 최소한의 밀가루를 사용 할 수 있습니다. 그러므로 출력은 11입니다.

예제3 : a가 만들 수 있는 빵의 밀가루의 양이 b가 만들 수 있는 빵의 밀가루의 양보다 항상 적기 때문에 어떤 조합으로도 a와 b가 같은 양의 밀가루를 사용할 수 없으므로 출력은 -1입니다.

주의사항
- 모든 라이브러리는 사용가능합니다. - c, c++를 사용하신다면 practice14-c/c++, java를 사용하신다면 practice14-java, python을 사용하신다면 practice14-python에 제출해주시기 바랍니다.
- 제출하는 란만 다르고, 문제는 동일합니다. 본인이 사용하는 언어를 제출하는 란에 제출해 주시기 바랍니다.
- 시간제한: C/C++ 0.1초. java 1초, python 2초.
- 테스트 케이스는 총 10개입니다. 3개의 테스트 케이스는 공개되고, 7개의 테스트 케이스는 실습시간동안 공개되지 않습니다.

힌트
1대n 최단거리를 구하는 알고리즘을 이용해야 합니다.

모범답안 (cpp)

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

int v = 402;
vector<pair<int, int>> adj[410];
const int start = 401;
const int INF = 987654321;

int vertex(int delta) {
    return delta + 200;
}

vector<int> dijkstra(int src) {
    vector<int> dist(v, INF);
    int cost, here;
    dist[src] = 0;
    priority_queue<pair<int, int>> pq;
    pq.push(make_pair(0, src));
    while (!pq.empty()) {
        cost = -pq.top().first;
        here = pq.top().second;
        pq.pop();
        if (dist[here] < cost) continue;
        for (int i = 0; i < adj[here].size(); ++i) {
            int there = adj[here][i].first;
            int nextDist = cost + adj[here][i].second;

            if (dist[there] > nextDist) {
                dist[there] = nextDist;
                pq.push(make_pair(-nextDist, there));
            }
        }
    }
    return dist;
}
int solve(const vector<int>& a, const vector<int>& b) {
    int i, delta, next;

    for (i = 0; i < v; i++) adj[i].clear();
    for (i = 0; i < a.size(); i++) {
        delta = a[i] - b[i];
        adj[start].push_back(make_pair(vertex(delta), a[i]));
    }

    for (delta = -200; delta <= 200; ++delta) {
        for (i = 0; i < a.size(); ++i) {
            next = delta + a[i] - b[i];
            if (abs(next) > 200) continue;
            adj[vertex(delta)].push_back(make_pair(vertex(next), a[i]));
        }
    }
    vector<int> shortest = dijkstra(start);
    int ret = shortest[vertex(0)];
    if (ret == INF) return -1;
    return ret;
}
```

```cpp
int main() {
    vector<int> a, b;
    int n, g;
    cin >> n;
    for (int i = 0; i < n; i++) {
        cin >> g;
        a.push_back(g);
        cin >> g;
        b.push_back(g);
    }
    cout << solve(a, b);
    return 0;
}
```