ARC083 / ABC074 解説

三谷庸 (wo01)

2017 年 9 月 16 日

For International Readers: English editorial starts on page 6.

A 問題

$N \times N$ のマス目にはマスが N^2 個あります。そのうち A マスを白く塗るので、黒く塗るマスは $N^2 - A$ 個あります。

したがって、入力から N, A を読み込んで $N^2 - A$ を出力すればよいことになります。

以下に C++ によるコード例を示します。

```
#include<cstdio>

using namespace std;

int main(){
    int N, A;
    scanf("%d%d", &N, &A);
    printf("%d\n", N * N - A);
    return 0;
}
```

B 問題

i 番目のボールは i 番目のタイプ A のロボット、または i 番目のタイプ B のロボットで回収できます。タイプ A のロボットで回収するときはロボットの移動距離は $2x_i$ となり、タイプ B のロボットで回収するときはロボットの移動距離は $2(K - x_i)$ となります。

$i = 1, 2, \ldots, N$ について順にボールの位置を見ていき、$2x_i$ と $2(K - x_i)$ のうち小さい方 (同じであるときはその値) を足し合わせれば答えが求まります。

以下に C++ によるコード例を示します (このコード例ではループの各段階ではなく最後に答えを 2 倍しています)。

```
#include<cstdio>
#include<algorithm>

using namespace std;

const int MAX_N = 100;

int N, K;
int X[MAX_N];

int main(){
    scanf("%d", &N);
    scanf("%d", &K);
    for(int i = 0; i < N; ++i){
        scanf("%d", X + i);
    }
    int ans = 0;
    for(int i = 0; i < N; ++i){
        int tmp = min(X[i], K - X[i]);
        ans += tmp;
    }
    printf("%d\n", ans * 2);
    return 0;
}
```

C 問題

ビーカーにちょうど x [g] の水を入れられる必要十分条件は、非負整数 i, j が存在して $x = 100Ai + 100Bj$ となることです。同様に、ビーカーに y [g] の砂糖を入れられる必要十分条件は、非負整数 i, j が存在して $y = Ci + Dj$ となることです。

x, y としては F 以下の値だけを考えればよいので、x, y としてありうる値は $O(F^2)$ 時間かけてすべての i, j を試すことで列挙できます。

最後に、すべての (x, y) の組 ($O(F^2)$ 個しかありません) について、ビーカーに水を x [g] と砂糖を y [g] 入れたときに砂糖が水にすべて溶けているかどうか調べ、溶けているものについて濃度が最大のものをとって出力すればよいです。

D 問題

都市を頂点、道路を辺としてグラフ理論の言葉で説明します。

条件を満たすグラフが存在するとき、任意の頂点 u, v について、これらの間には辺がないか、長さ $A_{u,v}$ の辺があるかのどちらかとなります。なぜなら、

- u, v の間に $A_{u,v}$ よりも長い辺があった場合、この辺は取り除いても最短距離に影響を与えず、
- u, v の間に $A_{u,v}$ よりも短い辺があった場合、u, v 間の最短距離は明らかに $A_{u,v}$ よりも短くなる

からです。

グラフ G を任意の u, v の間に長さ $A_{u,v}$ の辺を張ったグラフとします。上の考察より、G の部分グラフに関してのみ考えればよいことになります。

どの頂点間に辺を張る必要があるか考えます。

A (の対角線上以外の要素) のうち $A_{u,v}$ が最小値であるとき、A が最短距離を表す表であるようなグラフには u と v を長さ $A_{u,v}$ で結ぶ辺が必要です。なぜなら、そうでないとすると他の頂点 w を経由して頂点 u から頂点 v まで距離 $A_{u,v}$ で移動できる必要がありますが、そのとき頂点 u と頂点 w の距離は $A_{u,v}$ よりも小さくなるからです。

同様に G のすべての辺について短い方から見ていくことで、求めるべきグラフは以下のとおりであることがわかります (G から辺 $\{u,b\}$ を取り除いたグラフを $G\backslash\{u,v\}$ と書いています)。

- ある u, v について、$G\backslash\{u,v\}$ における uv 最短距離が $A_{u,v}$ 未満であるとき、条件を満たすグラフは存在しない。
- そうでないとき、$G\backslash\{u,v\}$ における uv 最短距離が $A_{u,v}$ より大きいような u, v に限り辺を張ったグラフが求めるべきグラフである。

$G\backslash\{u,v\}$ における uv 最短路の長さを $A'_{u,v}$ と書くことにします。$A'_{u,v}$ と $A_{u,v}$ の大小を比較するために、まず G における全点対最短距離を Warshall-Floyd 法を用いて求め、その値を B とします。

$B_{u,v} < A_{u,v}$ となる u,v が存在するとき、この u,v について、$A'_{u,v} = B_{u,v} < A_{u,v}$ となります。よって条件を満たすグラフは存在しません。

あとは、任意の u,v について $A_{u,v} = B_{u,v}$ であったとして、$A'_{u,v}$ が $A_{u,v}$ と等しいか、それより大きいかを判定できればよいことになります。$A'_{u,v} = A_{u,v}$ であることは、以下と同値です。

$$\exists w, u \neq w, v \neq w, B_{u,v} = B_{u,w} + B_{w,v} \tag{1}$$

実際、$A'_{u,v} = A_{u,v}$ のとき、w として $G\backslash\{u,v\}$ における uv 最短路上の頂点をとれます。逆に、上のような w が取れるとき、G における w を通った uv 最短路は $G\backslash\{u,v\}$ における uv 最短路にもなっています。

よって、条件を満たすグラフが存在するとき、答えは条件 (1) を満たす u,v について $A_{u,v}(= A'_{u,v} = B_{u,v})$ の和をとったものになります。

N をグラフの頂点の数とするとき、B を求めるための Warshall-Floyd 法の計算量は $O(N^3)$ です。また、条件 (1) は各 u,v について $O(N)$ で確かめられるので、全体の計算量は $O(N^3)$ です。以上から、アルゴリズム全体の計算量は $O(N^3)$ となるので、十分高速です。

E 問題

頂点 v の子孫について色と重みの割り当てができているとして、頂点 v の色と重みを決めることを考えます。頂点 v の直接の子を $u_1, u_2, \ldots, u_k$ とします。

各 u_i について、u_i およびそのすべての子孫に割り当てた色を反転させても、頂点 u_i およびその子孫への色と重みの割り当ては条件を満たしたままです。また、これらの頂点のうち頂点 v と異なる色の重みの和 y はできるだけ小さくしたほうが v の先祖に色と重みを割り当てやすくなります。

よって、頂点 v を根とする部分木に含まれる頂点の色と重みを決めるには、次の問題を解けばよいことになります。

- $u_1, u_2, \ldots, u_k$ のうちいくつかを v と同じ色に、それら以外を v と異なる色に塗り、v の子孫 (v を除く) に含まれる v と同じ色の頂点の重みの和を X_v 以下にできるか。できる場合、それらの頂点のうち v と異なる色であるものの重みの和の最小値はいくらか。

この問題は、$x_1, x_2, \ldots, x_k$ (ただし $x_i = X_{u_i}$) の情報の他に、u_i を根とする部分木に含まれる頂点であって u_i と異なる色のものの重みの和 y_i の情報がわかれば、動的計画法により解くことができます。

与えられた木の頂点を根から遠い順に並べ、それぞれについて上の動的計画法を行うことで、与えられた木全体への色と重みの割り当てが可能かどうか判定できます。

F 問題

まず、与えられた点集合から、以下のようなグラフ G を構築します。

- 頂点集合は $(X,1), (X,2), \ldots, (X,N), (Y,1), (Y,2), \ldots, (Y,N)$ の $2N$ 個。これらにそれぞれ番号 $0, 1, \ldots, 2N-1$ を付ける。

- 頂点 (X, x) と頂点 (Y, y) は、ある i が存在して $x_i = x$ かつ $y_i = y$ であるときに限り辺で結ばれている。
- それら以外に辺は存在しない。

グラフ G の頂点はボールを回収するロボットに、辺は回収されるボールに対応します。また、辺 $\{u, v\}$ に対応するボールは、頂点 u または頂点 v に対応するロボットによってのみ回収できます。以下、このグラフに関する問題として考えることにします。

まず、G の連結成分に頂点の数と辺の数が異なるものが存在する場合、答えは 0 です。そうでない場合、G の連結成分ごとに順序を数えられれば十分であるので、以下では G が連結であるとして説明します。

ロボットの起動順序を考える前に、ロボットとそれが回収するボールの対応付けの方法を考えます。

$2N$ 頂点 $2N$ 辺のグラフにはサイクルがちょうど 1 つあります。このサイクルに含まれない頂点に対応するロボットについては、それが回収すべきボールは一意に決まります。また、サイクルに含まれる頂点に対応するロボットについては、回収すべきボールとの対応付けが (サイクル全体で) 2 とおりに決まります。これらを両方試すことにすると、ロボットとボールの対応付けが定まったことになります。

次に、ロボットの起動順序について考えます。

グラフ G において頂点 v は頂点 $u_1 < u_2 < \cdots, u_k$ に隣接しているとします。また、頂点 v に対応するロボットは辺 $\{v, u_i\}$ に対応するボールを回収することになっているとします。このとき、頂点 $u_1, u_2, \ldots, u_{i-1}$ に対応するロボットは頂点 v に対応するロボットよりも先に起動する必要があります。また、この条件がすべての v に対して満たされていれば、決めた対応のとおりにロボットがボールを回収できることもわかります。

グラフ G と同じ頂点集合について、各頂点 v について v から上の記法における $u_1, u_2, \ldots, u_{i-1}$ に向けた辺を張ったグラフを G' とします。このグラフ G' は森になるので、G' 上で動的計画法を行うことで求める順序の個数が得られます。

以上で答えが求まりました。

ARC083 / ABC074 Editorial

wo01

September 16, 2017

A

```
#include<cstdio>

using namespace std;

int main(){
    int N, A;
    scanf("%d%d", &N, &A);
    printf("%d\n", N * N - A);
    return 0;
}
```

B

```
#include<cstdio>
#include<algorithm>

using namespace std;

const int MAX_N = 100;

int N, K;
int X[MAX_N];

int main(){
    scanf("%d", &N);
    scanf("%d", &K);
    for(int i = 0; i < N; ++i){
        scanf("%d", X + i);
    }
    int ans = 0;
    for(int i = 0; i < N; ++i){
        int tmp = min(X[i], K - X[i]);
        ans += tmp;
    }
    printf("%d\n", ans * 2);
    return 0;
}
```

C

We can pour exactly x grams of water if there exist non-negative integers i, j such that $x = 100Ai + 100Bj$. Similarly, we can pour exactly y grams of sugar if there exist non-negative integers i, j such that $y = Ci + Dj$.

Since $x, y \leq F$, we can generate all possible values of x, y in $O(F^2)$ time.

Finally, for each possible pair (x, y) (There are $O(F^2)$ such pairs), consider the case where we put x grams of water and y grams of sugar, check whether it satisfies the conditions, and compute the maximum density.

D

If there exist three vertices u, v, w such that $A_{u,v} + A_{v,w} < A_{u,w}$, the distances are contradictory, thus you should output -1. From now on, we assume that there are no such triplets.

We call a (unordered) pair of two distinct vertices (u, v) *important* if there exists no w such that $w \neq u, w \neq v, A_{u,w} + A_{w,v} = A_{u,v}$.

- If (u, v) is important, the shortest path between u and v must not pass through other vertices. Thus, we must add an edge between u and v with cost $A_{u,v}$.
- If (u, v) is not important, we don't need to add any edges between these vertices. For some w, it satisfies $A_{u,w} + A_{w,v} = A_{u,v}$, and you can just follow the shortest path between u and w and the shortest path between w and v.

Therefore, the answer is the sum of $A_{u,v}$ over all important pairs (u, v).

E

Suppose that we've already decided the colors and weights of all descendants of v, and among those vertices, the conditions are satisfied. The only things that matter for the decision of colors and weights of other vertices (ancestors of v) are:

- The sum of weights of all black vertices in the subtree rooted at v.
- The sum of weights of all white vertices in the subtree rooted at v.

One of these two values must be exactly X_v. To make things better in the future, we want to minimize the other value. Let $dp[v]$ be the minimum of the other value.

How can we compute $dp[v]$? Let $u_1, u_2, \ldots, u_k$ be the children of v. Without loss of generality, we can assume that v is black.

We keep two values: B (the sum of weights of black vertices) and W (the sum of weights of white vertices). They are initialized to zeroes. Now, for each u_i, we do one of the following:

- Make u_i black: add X_{u_i} to B and $dp[u_i]$ to W.
- Make u_i white: add $dp[u_i]$ to B and X_{u_i} to W.

After that, we want to minimize the value of W, under the constraint that B must be at most X_v. This can be done in $O(kX_v)$ by a simple DP.

F

First, construct a graph G as follows:

- There are $2N$ vertices: $X_1, \ldots, X_N, Y_1, \ldots, Y_N$. (Also, let's number the vertices $0, 1, \ldots, 2N-1$ in this order.)
- For each i, add an edge between two vertices X_{x_i} and Y_{y_i}.

Each vertex in G corresponds to a robot, and each edge in G corresponds to a ball. An edge (ball) between two vertices u and v can be collected by either vertex (robot) u or vertex (robot) v.

For each connected component in this graph, the number of edges and vertices must be the same. Otherwise, the answer is 0. We will assume that the graph is connected (otherwise, solve the problem independently for each component and combine the results). Thus, we have a connected graph with $2N$ vertices and $2N$ edges.

How can we assign edges to vertices? The graph contains exactly one cycle, and for edges in the cycle, there are two ways to assign them to vertices. For other edges, the assignment can be uniquely determined.

Let's fix an assignment of edges to vertices. In how many ways can we activate the vertices to satisfy this assignment?

Suppose that in the graph G, a vertex v is adjacent to vertices $u_1 < u_2 < \cdots, u_k$. Also, suppose that v is assigned to the edge $\{v, u_i\}$. Then, the vertices $u_1, u_2, \ldots, u_{i-1}$ must be activated before v. These conditions are sufficient: if this is satisfied for all v, the robots can collect all balls.

Now, construct another graph G': for each v, add an edge from v to $u_1, u_2, \ldots, u_{i-1}$ in G'. The answer is the number of topological orderings in this graph. Since G' is a (directed) forest, it's east to compute the answer.