

# 日立製作所 社会システム事業部 プログラミングコンテスト 2020 解説

writer: tempura0224 heno239 beet kort0n

2020 年 3 月 8 日

*For International Readers: English editorial starts on page 8.*

## A: Hitachi String

$S$  が hi または hihi または hihihhi または hihihihhi または hihihihihhi の時は Yes で、それ以外の場合は No です。

Listing 1 c++ での実装例

---

```
1 #include<bits/stdc++.h>
2 using namespace std;
3 int main() {
4     string s; cin >> s;
5     string hitachi = "";
6     for (int i = 1; i <= 5; i++) {
7         hitachi += "hi";
8         if (hitachi == s) {
9             cout << "Yes" << endl;
10            return 0;
11        }
12    }
13    cout << "No" << endl;
14    return 0;
15 }
```

---

## B: Nice Shopping

割引券を使う時と使わない時に分けて考えます。

まず、割引券を使わない時は、最も安い冷蔵庫と最も安い電子レンジを購入するのが最適です。

次に、割引券を使う時は、全ての割引券について支払総額を計算すればよいです。

Listing 2 c++ での実装例

---

```
1 #include<bits/stdc++.h>
2 using namespace std;
3 int main() {
4     int A, B, M;
5     cin >> A >> B >> M;
6     vector<int> a(A);
7     vector<int> b(B);
8     for (int i = 0; i < A; i++)cin >> a[i];
9     for (int i = 0; i < B; i++)cin >> b[i];
10    int minA = *min_element(a.begin(), a.end());
11    int minB = *min_element(b.begin(), b.end());
12    int ans = minA+minB;
13    for (int i = 0; i < M; i++) {
14        int x, y, c;
15        cin >> x >> y >> c; x--; y--;
16        ans = min(ans, a[x] + b[y] - c);
17    }
18    cout << ans << endl;
19    return 0;
20 }
```

---

## C: THREE

「 $p_i$  と  $p_j$  の和または積が 3 の倍数である」という条件の否定は「 $p_i, p_j$  を 3 で割ったあまりがともに 1 である、または、 $p_i, p_j$  を 3 で割ったあまりがともに 2 である」です。  
つまり、3 で割ったあまりが 1 である整数どうしや、3 で割ったあまりが 2 である整数どうしが木の上で距離 3 にならないような順列をつくるのがこの問題の目標です。

まず、頂点 1 からの距離が奇数である頂点を赤色、偶数である点を青色に塗ります。すると、与えられるグラフが木であることから、距離が 3 である頂点の組は、必ず一方が赤色で一方が青色になります。

### 赤色の頂点も青色の頂点も $\frac{N}{3}$ 個より多いとき

赤色の頂点に 3 で割って 1 余る整数を、青色の頂点に 3 で割って 2 余る整数を順に割り当てていき、残った頂点に 3 の倍数を割り当てます。  
すると、3 で割ったあまりが 1 である整数どうしや、3 で割ったあまりが 2 である整数どうしが距離 3 になることはなく、条件を満たしています。

### 赤色の頂点が $\frac{N}{3}$ 個以下のとき

赤色の頂点に 3 の倍数を順に割り当てていき、残った 3 の倍数と 3 で割って 1, 2 余る整数たちを青色の頂点に割り当てます。  
この場合も、3 で割ったあまりが 1 である整数どうしや、3 で割ったあまりが 2 である整数どうしが距離 3 になることはなく、条件を満たしています。

### 青色の頂点が $\frac{N}{3}$ 個以下のとき

青色の頂点に 3 の倍数を順に割り当てていき、残った 3 の倍数と 3 で割って 1, 2 余る整数たちを赤色の頂点に割り当てます。  
この場合も、3 で割ったあまりが 1 である整数どうしや、3 で割ったあまりが 2 である整数どうしが距離 3 になることはなく、条件を満たしています。

以上より、すべての場合について、条件を満たすような順列を構成することができました。

## D: Manga Market

まず、回る店の集合を固定した場合を考えます。このとき、店  $i$  の直後に店  $j$  を訪れる場合と店  $j$  の直後に店  $i$  を訪れる場合を考えると、前者の方が所要時間が短くなる為の必要十分条件は、1 つ目の店へ向かい始める時刻を  $T$  と置くと、

$$\begin{aligned} & 1 + a_i(T + 1) + b_i + 1 + a_j(T + 1 + a_i(T + 1) + b_i + 1) + b_j \\ & < 1 + a_j(T + 1) + b_j + 1 + a_i(T + 1 + a_j(T + 1) + b_j + 1) + b_i \\ \Leftrightarrow & a_j(b_i + 1) < a_i(b_j + 1) \\ \Leftrightarrow & \frac{a_j}{b_j + 1} < \frac{a_i}{b_i + 1} \end{aligned}$$

です。これより、 $\frac{a}{b+1}$  で降順にソートを行い、この順に各要素を見て

$$dp_{i,j} = i \text{ 番目の店まで見て、} j \text{ 個の店を回るのに掛かる所要時間の最小値 } (0 \leq j \leq i \leq N)$$

として動的計画法を行うことで、最適解を得ることが出来ます。しかし、この動的計画法の時間計算量は  $O(N^2)$  であり、問題の制約のもと Time Limit に間に合いません。

以下  $a$  の値に着目して場合分けを行います。

### $a = 0$ である店について

これらの店については、 $a \geq 1$  である店を回った後に、 $b$  が小さい順に回れるだけ回るのが明らかに最適です。

### $a \geq 1$ である店について

これらの店を回る場合、訪れる店の数が増える毎に列の待ち時間が指数関数的に増加しますから、回れる店の数は  $O(\log T)$  個です。これより、前述の動的計画法で  $j$  が大きい場合を調べる必要が無くなりますから、 $O(N \log T)$  で動的計画法を行うことが出来ます。(この問題の制約の元では  $0 \leq j \leq 28$  のみ調べれば良いです。)

以上より、 $a \geq 1$  である店のみを  $j$  ( $0 \leq j \leq 28$ ) 個回る場合の所要時間の最小値を前述の動的計画法で求めた後、各場合について  $a = 0$  である店をいくつ回れるかを調べることで、元の問題の最適解を得ることが出来ます。

全体での時間計算量は  $O(N \log N + N \log T)$  です。

## E: Odd Sum Rectangles

以下では  $N \geq M$  であるとします。  $H = 2^N, W = 2^M$  とします。  
 $0 \leq i \leq H - 1, 0 \leq j \leq W - 1$  に対して、

$$f(i, j) = \left( \sum_{r=1}^i \sum_{c=1}^j a_{r,c} \right) \% 2 \quad (\text{ただし } f(0, i) = f(j, 0) = 0) \quad (1)$$

と定めると、  $S(i_1, i_2, j_1, j_2) \equiv f(i_2, j_2) + f(i_2, j_1 - 1) + f(i_1 - 1, j_2) + f(i_1 - 1, j_1 - 1) \pmod{2}$  であるため、

$S(i_1, i_2, j_1, j_2)$  が奇数であることと、  $f(i_2, j_2) + f(i_2, j_1 - 1)$  と  $f(i_1 - 1, j_2) + f(i_1 - 1, j_1 - 1)$  の偶奇が異なっていることは同値です。

今、  $j_1, j_2$  を固定すると、

$S(i_1, i_2, j_1, j_2)$  が奇数となる  $(i_1, i_2)$  の個数  
 $= (f(x, j_2) + f(x, j_1 - 1)$  が奇数となる  $x$  の個数)  $\times$   $(f(x, j_2) + f(x, j_1 - 1)$  が偶数となる  $x$  の個数)  
 です。

これは、  $f(x, j_2) + f(x, j_1 - 1)$  が奇数となる  $x$  の個数がちょうど  $\frac{H}{2}$  個であるとき最大値  $\frac{H^2}{4}$  を取ります。

$j_1, j_2$  の取り方は  $\frac{W(W-1)}{2}$  通りありますから、グリッドの奇妙さの上界として、  $\frac{H^2}{4} \times \frac{W(W-1)}{2}$  が得られます。

実際にこの上界が達成可能であることを示します。

mod 2 上で階差を取ることで  $f(i, 0) = f(0, j) = 0$  を満たすような  $f$  から (1) 式を満たす  $a$  を復元することは常に可能なので、上界を達成するような  $f$  が存在することを証明すればよいです。以下の 2 ステップに分けて示します。

### $N = M$ のとき

数学的帰納法で示します。

#### 1. $N = 1$ のとき

$f(0, 0) = f(0, 1) = f(1, 0) = 0, f(1, 1) = 1$  とすればよいです。

#### 2. $N = k$ のとき条件をみたく $f$ が存在すると仮定して、

$$\begin{aligned} F(i, j) &= F(i + 2^k, j) = F(i, j + 2^k) = f(i, j) \\ F(i + 2^k, j + 2^k) &= 1 - f(i, j) \quad (0 \leq i, j < 2^k) \end{aligned}$$

で  $F$  を定めます。この  $F$  が  $N = k + 1$  において条件を満たすことは容易に確認することができます。

$N > M$  のとき

$2^N \times 2^N$  での最大を達成する  $f$  から前の  $2^M$  行を取り出せば良いです。

以上により、グリッドの奇妙さの最大値が  $\frac{H^2}{4} \times \frac{W(W-1)}{2}$  であることと、そのようなグリッドの構築方法を示すことができました。

## F: Preserve Diameter

$G$  の直径を  $L$  と置きます。まず、 $H$  の性質について考察します。 $H$  において、直径の端点となるような組はちょうど一つです。その組を  $(x, y)$  とします。 $x$  を根とした  $H$  上の BFS 木を考えます。このとき、 $x$  からの距離の差の絶対値が 1 以下であるような頂点对全てについて、その間に辺が存在する必要があります（存在しなければその間に辺を追加しても直径は変わらないため）。また、BFS 木の性質から、距離の差の絶対値が 1 より大きいような頂点对の間には辺は存在しません。

逆に、前述した条件が満たされていれば、問題文中の四つ目の条件を満たすことは明らかです。

したがって、この問題は以下の問題に帰着されます。

以下の二つの条件を満たすように、各頂点  $v$  について  $x$  からの距離  $d_v \geq 0$  を定める方法は何通りあるか。

1.  $d_v = L$  となるような  $v$  がちょうど一つ存在する。
2. 頂点  $u, v$  が  $G$  で隣接しているなら、 $|d_u - d_v| \leq 1$

計算量を改善するため、この問題をさらに言い換えます。簡単のため、 $G$  の中心は一つであるものとし、その頂点を  $C$  とします。 $C$  は  $H$  における直径に必ず含まれることから、 $C$  を根とした BFS 木を考えます。ここで、先ほどまでとは異なり、 $C$  からの距離  $s_v$  を符号付きで考えることにします。

すると、問題は次のように変形されます。

以下の二つの条件を満たすように、各頂点  $v$  について  $C$  からの距離  $s_v$  を定める方法は何通りあるか。

1.  $s_x = +L/2, s_y = -L/2$  であるような頂点  $x, y$  がそれぞれちょうど一つずつ存在する
2. 頂点  $u, v$  が  $G$  で隣接しているなら、 $|s_u - s_v| \leq 1$

この通り数は、 $|s_v| = L/2$  となりうるような頂点に注目しながら、各辺に  $+1, -1, 0$  のいずれかを割り当てる木 DP を行うことで求められます。各頂点において、 $s_x = +L/2$  であるような頂点  $x$  が 0 個、1 個、2 個以上、 $s_y = -L/2$  であるような頂点  $y$  が 0 個、1 個、2 個以上のどれに該当するかを状態として持てばよいです。符号をつけたことにより、最後に通り数を 2 で割る必要が生じたことに注意してください。

中心が二つの場合も同様の考察により解くことができます。計算量は  $O(N)$  です。

## A: Hitachi String

The answer is Yes when  $S$  is hi,hihi,hihihi,hihihihi, or hihihihhi, and No otherwise.

Listing 3 c++ implementation example

---

```
1 #include<bits/stdc++.h>
2 using namespace std;
3 int main() {
4     string s; cin >> s;
5     string hitachi = "";
6     for (int i = 1; i <= 5; i++) {
7         hitachi += "hi";
8         if (hitachi == s) {
9             cout << "Yes" << endl;
10            return 0;
11        }
12    }
13    cout << "No" << endl;
14    return 0;
15 }
```

---



## B: Nice Shopping

There are two cases: to use a discount ticket or not.

If we don't use a ticket, we can buy the cheapest refrigerator and microwave.

If we use a ticket, for each ticket, we can compute the value  $a_{x_i} + b_{y_i} - c_i$ .

Listing 4 c++ implementation example

---

```
1 #include<bits/stdc++.h>
2 using namespace std;
3 int main() {
4     int A, B, M;
5     cin >> A >> B >> M;
6     vector<int> a(A);
7     vector<int> b(B);
8     for (int i = 0; i < A; i++)cin >> a[i];
9     for (int i = 0; i < B; i++)cin >> b[i];
10    int minA = *min_element(a.begin(), a.end());
11    int minB = *min_element(b.begin(), b.end());
12    int ans = minA+minB;
13    for (int i = 0; i < M; i++) {
14        int x, y, c;
15        cin >> x >> y >> c; x--; y--;
16        ans = min(ans, a[x] + b[y] - c);
17    }
18    cout << ans << endl;
19    return 0;
20 }
```

---

## C: THREE

Since a tree is a bipartite, let's color the vertices in red and blue such that adjacent vertices don't have same colors. Then, every pair with distance 3 have a different colors!

Let  $R$  be the number of red vertices and  $B$  be the number of blue vertices. We will assume  $R \leq B$ .

Let  $X$  be the largest integer that doesn't exceed  $N/3$ . There are two cases.

when  $R \leq X$

Assign multiples of three to red vertices. Since every pair has a red vertex, the product of the values is always a multiple of 3.

when  $R > X$

Assign all  $3k + 1$  type values to red vertices, and assign  $3k + 2$  type values to blue vertices. Then, for each pair, one of the followings holds:

1. one of the vertex is assigned a multiple of 3. In this case, the product of the values is a multiple of 3.
2. one vertex has a value of  $1 \pmod{3}$ , and another has  $2 \pmod{3}$ . In this case, the sum of the values is a multiple of 3.

## D: Manga Market

Let's say we have determined the set of stores to visit. If we visit the store  $i$  right before the store  $j$ , the following inequality should hold:

$$\begin{aligned} & 1 + a_i(T + 1) + b_i + 1 + a_j(T + 1 + a_i(T + 1) + b_i + 1) + b_j \\ & \leq 1 + a_j(T + 1) + b_j + 1 + a_i(T + 1 + a_j(T + 1) + b_j + 1) + b_i \\ \Leftrightarrow & a_j(b_i + 1) \leq a_i(b_j + 1) \\ \Leftrightarrow & \frac{a_j}{b_j + 1} \leq \frac{a_i}{b_i + 1} \end{aligned}$$

Here  $T$  denotes the time when we start moving to the store  $i$ . This inequality states that swapping the order of store  $i$  and  $j$  doesn't profit. Therefore, we can sort the stores by  $\frac{a}{b+1}$  and do the following dp:

$dp_{i,j}$  = the minimum time needed to visit  $j$  stores among the first  $i$  stores ( $0 \leq j \leq i \leq N$ )

But this dp is  $O(N^2)$  and too slow, so we need to speed it up.

when  $a = 0$

We visit the stores in increasing order of  $b$ , after visiting stores with  $a \geq 1$ .

when  $a \geq 1$

It can be seen that  $dp_{i,j}$  grows exponentially in accordance with  $j$ . So we should only maintain  $O(\log T)$  candidates of  $j$ , and then the aforementioned dp is fast enough.

After calculating this dp, we try all possible  $j$  and visit stores with  $a = 0$ . The total complexity is  $O(N \log N + N \log T)$

## E: Odd Sum Rectangles

Let's assume  $N \geq M$ . Let  $H = 2^N, W = 2^M$ . There are  $W(W - 1)/2$  pair of  $(j_1, j_2)$  such that  $(1 \leq j_1 \leq j_2 \leq W - 1)$ . Let  $f(i)$  ( $0 \leq i \leq H - 1$ ) be the parity of  $S(1, i, j_1, j_2)$ . Then, the number of odd  $S(i_1, i_2, j_1, j_2)$  is equal to (the number of  $i$  such that  $f(i)$  is 0)  $\times$  (the number of  $i$  such that  $f(i)$  is 1). The upper bound of this value is  $(H/2) \times (H/2)$ . Therefore the upper bound of the oddness of the grid is  $\frac{H^2}{4} \times \frac{W(W-1)}{2}$ .

We will prove that this upper bound can be reached in a constructive way.

when  $N = M$

We will prove it by induction.

1. When  $N = 1$ . grid with 1 satisfy the conditon.
2. Let's assume we have solution with  $N = K$ . Then we can construct a  $(2^{K+1} - 1)$  by  $(2^{K+1} - 1)$  grid board in the following manner:
  - The upper left, upper right, bottom left, and bottom right part is the same as the solution to  $N = K$ .
  - The cell  $(2^K, 2^K)$  contains 1.
  - other cells (in row  $2^K$  or column  $2^K$ ) contain 0.

It is not hard to prove this construction is correct.(Sorry, I don't have enough time to finish this part.)

when  $N > M$

Create a  $2^N - 1$  by  $2^N - 1$  grid board, and just take the first  $2^M - 1$  columns.

## F: Preserve Diameter

Let  $L$  be the diameter of  $G$ . Let's consider how  $H$  looks like. In  $H$ , exactly one pair of vertices is the endpoints of the diameter. Let  $(x, y)$  be the pair. We consider the BFS tree with root  $x$ . Here, for all pairs of vertices whose distance from  $x$  differs no more than 1, there must be an edge connecting them. Also, since we build BFS tree, there is no edge between two vertices whose distance from  $x$  differs more than 1.

It is easy to see that this necessary condition is indeed sufficient.

Therefore, we need to solve the following problem.

How many ways are there to assign value  $d_v$  to each vertex  $v$ ? Here,  $d_v$  stands for the distance from  $x$ .

1. there is exactly one  $v$  such that  $d_v = L$ .
2. If vertices  $u$  and  $v$  are adjacent in  $G$ ,  $|d_u - d_v| \leq 1$ .

We will rephrase this problem further. We will assume the diameter is even and let  $C$  be the center.  $C$  must be contained in the diameter of  $H$ . Now, we will consider signed distance  $s_v$  from  $C$ .

Then, we need to solve the following:

How many ways are there to assign value  $s_v$  to each vertex  $v$ ?

1. There is exactly one vertex  $x$  such that  $s_x = +L/2$  and one vertex  $y$  such that  $s_y = -L/2$ .
2. If vertices  $(u, v)$  are adjacent in  $G$ ,  $|s_u - s_v| \leq 1$ .

This can be solved by standard dp on tree. We just need to care about candidate vertex  $v$  that  $s_v$  can be  $L/2$  or  $-L/2$ .

If the diameter is odd, we can do the similar thing.

The total complexity is  $O(N)$ .