

# ABC 144 解説

kort0n, kyopro\_friends, sheyasutaka

2019 年 10 月 27 日

*For International Readers: English editorial starts on page 7.*

## A: 9x9

$A, B$  がともに 9 以下であるかどうかで場合分けをすればよいです。複数の条件を満たすかどうかの判断には論理積 (AND) 演算子を用います。以下は C 言語での実装例です。

---

```
1 #include<stdio.h>
2 int main(){
3     int a,b;
4     scanf("%d%d",&a,&b);
5     if(a<=9 && b<=9)printf("%d\n",a*b);
6     else printf("-1\n");
7 }
```

---

## B: 81

$A \times B = N$  となるような 1 以上 9 以下の整数  $A, B$  が存在するかどうか全探索で確かめます。以下は C 言語での実装例です。

---

```
1 #include<stdio.h>
2 int main(){
3     int n;
4     scanf("%d",&n);
5     for(int a=1;a<=9;a++)for(int b=1;b<=9;b++){
6         if(a*b==n){
7             puts("Yes");
8             return 0;
9         }
10    }
11    puts("No");
12 }
```

---

### C: walk on multiplication table

$(1, 1)$  から  $(a, b)$  に至るまでの移動回数は  $a + b - 2$  です。したがって問題は、「 $a \times b = N$  となるような  $(a, b)$  について、 $a + b - 2$  の最小値を求めよ」となります。対称性から  $a \leq b$  としてよいので、 $a \leq \sqrt{N}$  までを全探索することにより  $O(\sqrt{N})$  で解くことができます。

## D: Water Bottle

### 解法 1

水筒を  $\theta$  傾けた際に水が溢れるか否かには明らかに単調性がありますから、二分法により解を求めることが出来ます。

水筒を  $\theta$  傾けた際に水筒の中に存在出来る水量  $f(a, b, \theta)$  を計算し、 $x$  と比較することにより、解の範囲を狭めることが出来ます。

ただし、 $f(a, b, \theta)$  の計算では、場合分けが必要であることに注意してください。

具体的な計算式については、以下の解答例コードをご確認ください。

C++ による解答例:<https://atcoder.jp/contests/abc144/submissions/8133803>

### 解法 2

更に計算を進めると、答えを直接計算することが出来ます。

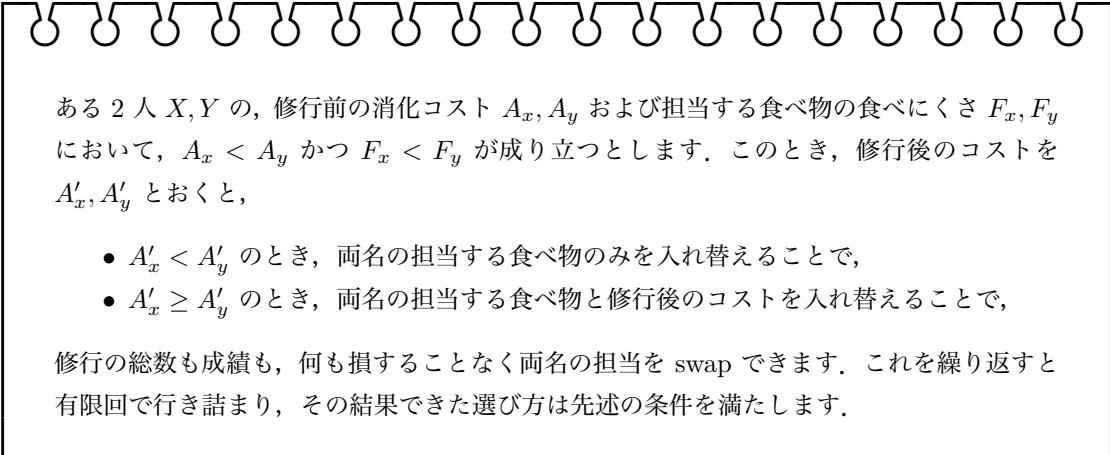
こちらも  $a, b, x$  の値に応じて場合分けが必要であることに注意してください。

具体的な計算式については、以下の解答例コードをご確認ください。

C++ による解答例:<https://atcoder.jp/contests/abc144/submissions/8133804>

## E: Gluttony

メンバーを  $A_i$  で昇順に、食べ物を  $F_i$  で降順にソートし、 $i$  番目のメンバーに  $i$  番目の食べ物を担当させるのが最適です。以下にそれを示します。



ある 2 人  $X, Y$  の、修行前の消化コスト  $A_x, A_y$  および担当する食べ物の食べにくさ  $F_x, F_y$  において、 $A_x < A_y$  かつ  $F_x < F_y$  が成り立つとします。このとき、修行後のコストを  $A'_x, A'_y$  とおくと、

- $A'_x < A'_y$  のとき、両名の担当する食べ物のみを入れ替えることで、
- $A'_x \geq A'_y$  のとき、両名の担当する食べ物と修行後のコストを入れ替えることで、

修行の総数も成績も、何も損することなく両名の担当を swap できます。これを繰り返すと有限回で行き詰まり、その結果できた選び方は先述の条件を満たします。

あとは「全てのメンバーについて完食にかかる時間を  $x$  以下にできるか」という問題を考えて、二分探索をすることで解けます。時間計算量は  $\Theta(N \log X)$  です (ここで  $X$  は答えの値)。

## F: Fork in the Road

塞ぐ通路を決めた際 (及び、通路を塞がないと決めた際) に部屋  $N$  に到達するまでの移動回数の期待値は、

$$dp_i = \text{部屋 } i \text{ から部屋 } N \text{ へ到達するまでの移動回数の期待値}$$

と  $dp$  テーブルを定義し、部屋番号について降順に計算して動的計画法を行うことで、時間計算量  $O(M)$  で計算することが出来ます。しかし、 $M$  個の通路全てについてこの値を計算しようとすると、全体での時間計算量が  $O(M^2)$  となり、間に合いません。

ここで、部屋  $i$  から出ていく通路を塞ぐを考えます。このとき、塞ぐ通路の選択に依らず、 $j = i + 1, \dots, N$  について  $dp_j$  の値は変わりません。塞ぐ通路を選択する際は、 $dp_i$  の値が最小となるように決めるのが最適です。これは、部屋  $i$  から通路が伸びてる部屋の集合を  $S_i$  と定義したとき、部屋  $\arg \min_{k \in S_i} dp_k$  へ進む通路を塞ぐことで達成出来ます。則ち、部屋  $i$  から出ていく通路を塞ぐ際には、これ以外の通路を塞いだ際のことを考慮する必要はありません。

以上より、前述の動的計画法を  $N$  回実行すれば最適解を求めることが出来ます。全体での時間計算量は  $O(NM)$  です。

C++ による解答例:<https://atcoder.jp/contests/abc144/submissions/8133816>

## A: 9x9

You can use conditional statement by checking whether both  $A, B$  are no more than 9. When checking if multiple conditions holds, you can use logical product (AND) operator. The following is an implementation example in C.

---

```
1 #include<stdio.h>
2 int main(){
3     int a,b;
4     scanf("%d%d",&a,&b);
5     if(a<=9 && b<=9)printf("%d\n",a*b);
6     else printf("-1\n");
7 }
```

---

## B: 81

Check if there exists a pair of integers  $A, B$  in a range of 1 to 9 (inclusive) such that  $A \times B = N$  by brute forcing. The following is an implementation example in C.

---

```
1 #include<stdio.h>
2 int main(){
3     int n;
4     scanf("%d",&n);
5     for(int a=1;a<=9;a++)for(int b=1;b<=9;b++){
6         if(a*b==n){
7             puts("Yes");
8             return 0;
9         }
10    }
11    puts("No");
12 }
```

---



## C: walk on multiplication table

The number of moves needed to reach  $(a, b)$  from  $(1, 1)$  is  $a + b - 2$ . Therefore, the problem can be rephrased as "find the minimum value of  $a + b - 2$  for all  $(a, b)$  such that  $a \times b = N$ ." By symmetry you can assume that  $a \leq b$ , so by performing brute-force searching for all  $a \leq \sqrt{N}$ , the problem can be solved in a total of  $O(\sqrt{N})$  time.

## D: Water Bottle

### Solution 1

Whether the water will be spilled or not when the bottle is tilted  $\theta$  degrees is apparently monotonic, so the answer can be found by binary searching.

By calculating  $f(a, b, \theta)$ , the maximum volume of water that can be inside the bottle when the bottle is tilted  $\theta$  degrees, and comparing with  $x$ , you can narrow the range of solution.

Note that when calculating  $f(a, b, \theta)$  you have to divide into cases.

For the specific formula, please refer to the following sample code.

Sample answer in C++:<https://atcoder.jp/contests/abc144/submissions/8133803>

### Solution 2

If you calculate further, you can directly calculate the answer.


Again, note that you need to divide into cases depending on the value of  $a, b, x$ .

For the specific formula, please refer to the following sample code.

Sample answer in C++:<https://atcoder.jp/contests/abc144/submissions/8133804>

## E: Gluttony

It is optimal to sort the members in the increasing order of  $A_i$ , sort the foods in the decreasing order of  $F_i$ , and assign the  $i$ -th food to the  $i$ -th member. The following is the proof.



For some two members  $X, Y$ , assume that their consumption coefficients,  $A_x, A_y$ , and the difficulty of foods assigned to them,  $F_x, F_y$ , meet such conditions that  $A_x < A_y$  and  $F_x < F_y$ . Let  $A'_x, A'_y$  be their consumption coefficient after training, then they can swap their assigned food without losing the number of total number of training and their score. It can be achieved by performing the following operation:

- If  $A'_x < A'_y$ , swap the foods assigned to them.
- If  $A'_x \geq A'_y$ , swap the foods assigned to them, and also swap their consumption coefficients.

This operation can be repeated finite times, and the resulting assignment meets the condition mentioned above.

All the left is doing binary search by solving problem of "whether or not it is possible to let the longest time of finishing food to be less than  $x$ ." So this problem can be solved in a total of  $\Theta(N \log X)$  time (where  $X$  is the answer).

## F: Fork in the Road

If you fix any passage to block (or decide not to block any passage), the expected number of passages he takes before he reaches Room  $N$  can be calculated by defining the following dp table

$dp_i$  = The expected number of passages when he travels from Room  $i$  to Room  $N$

and calculating them in the decreasing order of room index, in which case the whole dp table can be calculated in a total of  $O(M)$  time. However, if you try to calculate this value for each of  $M$  passages, the overall time complexity will be  $O(M^2)$ , so it won't fit in the time limit.

Now let's consider blocking a passage starting from Room  $i$ . Then, for each  $j = i + 1, \dots, N$ , the value of  $dp_j$  is independent from the choice of which road to block. When choosing the road to block, the optimal choice is such that  $dp_i$  is minimum. Let  $S_i$  be the set of rooms such that there exists a road from  $i$  to the room, then it can be achieved by blocking the road towards  $\arg \min_{k \in S_i} dp_k$ . In other words, when you block a road starting from Room  $i$ , you don't have to consider the cases when the other roads are blocked.

Therefore, you can find the optimal answer by performing the dynamic programming mentioned above  $N$  times. The problem can be solved in a total of  $O(NM)$  time.

Sample answer in C++:<https://atcoder.jp/contests/abc144/submissions/8133816>