# ARC 085/ABC 078 解説

Kohei Morita(yosupo)

平成 29 年 11 月 11 日

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

## A: HEX

16 進数に直さずとも，実は入力した文字の ASCII コードを比較するだけで十分です (C, C++ならば char 型のまま比較すれば ASCII コードで比較できます)。

想定解

## B: ISU

人が $n$ 人座るためには椅子の幅は何センチメートルあればよいでしょうか?

これは $Y \times n + Z \times (n+1)$ センチメートルです。$Y \times n$ が人の占有する幅で，$Z \times (n+1)$ が人と人，人と椅子の間に開ける幅です。

よって $n = 1, 2, ...$ と順に増やしていき，座れる人数のうち最も多い人数を調べればよいです。なお，人数の最大は入力例 3 の 49999 人です。

想定解

## C: HSI

1 回の実行にかかる時間を $x$ ms，全てのケースに正解する確率を $p$，この問題の答えを $y$ ms とします。$x = 1900M + 100(N - M), p = 1/2^M$ です。

まず，一回目の提出で必ず $x$ ms はかかります。その後は，確率 $p$ で終了します，そして確率 $1 - p$ でその提出には失敗し，更に提出を繰り返します。確率 $1 - p$ で失敗した場合の，その時点からかかる時間の期待値は $y$ ms です。

よって，$y = x + (1 - p) \times y$ が成立し，これを解くと $y = x/p$ が得られます。

よって答えは $x/p = (1900M + 100(N - M)) \times 2^M$ です。

# D: ABS

まず，N=1 の場合の答えは $|a_1 - W|$ です。

$2 \leq N$ の場合を考えます。X は初手で全部のカードを引くことにより，$|a_N - W|$ が達成できます。また，初手で $N - 1$ 枚のカードを引くことで $|a_{N-1} - a_N|$ が達成できます。

実は，初手で行うべき操作はこの 2 通りのみ，つまり 2 枚以上カードを残す理由はありません。なぜならば，その直後の手番で Y さんは，残り 1 枚になるようにカードを引くと $|a_{N-1} - a_N|$ を達成できます。よって 2 枚以上カードを残してもスコアが $|a_{N-1} - a_N|$ より高くなることはないからです。

以上より，この問題の ($2 \leq N$ の時の) 答えは $max(|a_N - W|, |a_{N-1} - a_N|)$ です。

# E: MUL

結論から言うと，この問題は最大流問題の双対問題である最小カットを使い解くことができます。

具体的には，頂点 $S, V, 1, 2, ..., N$ を用意し，各 $i$ について，

- $a_i \leq 0$ ならば，$S \to i$ に容量 $-a_i$ の辺を張る
- $a_i > 0$ ならば，$i \to T$ に容量 $a_i$ の辺を張る
- 各 $j = 2i, 3i, 4i, ...$ について，$i \to j$ に容量 $\infty$ の辺を張る

と辺を張り，$a_i > 0$ なる $a_i$ の総和から，構築したグラフの最小カットを引いたものが答えです。

最小カット問題の答えは最大流問題を解くことで得られます。頂点数が O(N) で，変数は $O(N \log N)$，そして $N \leq 100$ と小さいため，計算量が流量に依存しない最大流アルゴリズムならだいたい間に合うと思います。

## 理由

最小カット問題とは，S, T 以外の頂点を S 側と T 側に振り分け，S 側から T 側へ張られる辺の容量の総和を最小化する問題です。

ここで，S 側を 0，T 側を 1 として見ると，S には 0, T には 1，それ以外の頂点には 0, 1 を割り振り，(0 が割り振られた頂点) から (1 が割り振られた頂点) へ張られる辺の容量の総和を最小化する問題，と考えられます。

これを更に言い換えると，以下のような問題が解けることがわかります。

- 変数 $q_s, q_t, q_1, q_2, ..., q_v$ を考える
- 罰金条件 $(x, y, z)$ がたくさん与えられる。これは，$q_x$ が 0 で $q_y$ が 1 なら罰金 $z$ 円，という意味である。
- あなたは変数にそれぞれ $0, 1$ のどちらかを割り当て，罰金を最小化したい。ただし $q_s = 0, q_t = 1$ は固定である。

当然，変数が頂点，罰金条件が辺に対応します。なお，罰金条件の罰金 $z$ は必ず非負でないといけないことに注意してください。非負を許した場合，この問題は最大カット問題も含み，NP-hard となります。

では，元の問題をこのような問題に変形することを考えます。

まず，ある $i, j$($j$ は $i$ の倍数) について，$i$ が破壊されていて，$j$ が破壊されていない場合，明らかにおかしいです。

よって，この条件を表すために，$i$ が破壊されていて $j$ が破壊されていないと罰金 $\infty$ 円とすることを考えます。

これは，(0 を割り振る)=破壊されている，(1 を割り振る)=破壊されていない，とし，罰金条件 $(i, j, \infty)$ を足せばよいです。

実は，全ての $i, j$(j は i の倍数) について,「$i$ が破壊されているならば，$j$ が破壊されている」が成立する場合，このような破壊の方法が存在します。

よって，あとは破壊されていないと $a_i$ 円，つまり $q_i = 1$ ならば $a_i$ 貰える，という処理です。

$a_i \leq 0$ ならばこれは罰金と考えられるので問題ないです。罰金条件 $(S, i, -a_i)$ を足します。

$a_i > 0$ ならば，言い換えて,「$i$ が破壊されていると，$a_i$ 円が貰えない=$a_i$ 円罰金」と考えます。すると事前に答えに $a_i$ を貰っておくことにし，罰金条件 $(i, T, a_i)$ を足します。

以上より，この問題は最小カット問題に帰着できました。

今回の問題は $N$ 個の宝石に破壊された，破壊されていない，の 2 つの状態を割り振る問題でした。このように，幾つかの要素に 2 つの状態を割り振り，何かを最大化/最小化する問題で，更に DP ではとても解けそうにない場合，最小カットを疑うとよいです。

# F: MUL

この問題は，$a_i \neq b_i$，つまり $(a_i, b_i) = (0,1), (1,0)$ の個数を最小化する問題です。

$((a_i, b_i) = (0,1), (1,0)$ の個数)
$=$ ($(a_i, b_i) = (0,1)$ の個数) + ($(a_i, b_i) = (1,0)$ の個数)
$=$ ($(a_i, b_i) = (0,1)$ の個数) + ($b_i = 0$ の個数) − ($(a_i, b_i) = (0,0)$ の個数)

であり，$b_i = 0$ の個数は定数であるため，($(a_i, b_i) = (0,1)$ の個数) − ($(a_i, b_i) = (0,0)$ の個数) を最小化することを考えても良いです。

言い換えると，$a_i = 1$，つまり塗った場合のコストは 0 で，塗らなかった場合のコストは 1 だったり $-1$ するので，コストの総和を最小化してくださいという問題です。

操作それぞれについて行うかどうかを全探索していてはとても間に合いません。うまく探索する必要があります。

操作を行うかどうかとスコアの計算を並び換え，

$l_i = 1$ の操作たちを行うかどうか決め，$a_1$ のスコアを計算する
→ $l_i = 2$ の操作たちを行うかどうか決め，$a_2$ のスコアを計算する
→ $l_i = 3$ の操作たちを行うかどうか決め，$a_3$ のスコアを計算する
→ $l_i = 4$ の操作たちを行うかどうか決め，$a_4$ のスコアを計算する
→ …

という風に，操作を行うかどうかとスコアの計算を交互に行うことを考えます。このような順番にしても答えは変わりません。

なぜこのような順番にするかというと，$a_x$ のスコアを計算した後は $a_1, a_2, ..., a_x$ の色は忘れてよく，更に $a_{x+1}, a_{x+2}, ..., a_N$ については，あるところまでは 1，その後全て 0 という形になっているからです。

よって，この並び替えを行うだけで (まだ TLE ですが)DP を行い，計算量を多項式にすることができます。DP のキーは $(x, y)$ の 2 つで，「$a_x$ のスコアまで計算し，$a_y$ まで 1 になっている」という状態を現します。

$(x, x), (x, x+1), (x, x+2), ..., (x, N)$ から $(x+1, x+1), (x+1, x+2), (x+1, x+3), ..., (x, N)$ まで同時に遷移することを考え，この遷移を SegTree で高速化します。

区間 $l(= x), r$ の操作については，$(x, x), (x, x+1), ..., (x, r)$ の最小値を $(x+1, r)$ に遷移させれば良いです。$a_x$ のスコアを計算する，のパートは簡単です。最初の考察により，$a_x = 0$，つまり $y = x$ の場合以外はスコアが 0 だからです。

以上より，区間の $min$ を取るのが本質であり，この操作は $SegTree$ で簡単に実装できます。

# ARC 085/ABC 078 Editorial

Kohei Morita(yosupo)

November 11, 2017

## A: HEX

https://abc078.contest.atcoder.jp/submissions/1751369

## B: ISU

https://abc078.contest.atcoder.jp/submissions/1751697

## C: HSI

Let $x = 1900M + 100(N - M)$ be the time required to submit (and judge) one solution. Let $p = \frac{1}{2^M}$ be the probability that one submission passes. Let $y$ be the answer of this problem.

It takes $x$ ms to submit once. With probability $p$, it passes and the process ends, and with probability $1 - p$, you need more submissions and you need $y$ ms more on average.

Thus, you get $y = x + (1 - p) \times y$, and $y = x/p$.

Therefore, the answer is $y = x/p = (1900M + 100(N - M)) \times 2^M$.

## D: ABS

If $N = 1$, the answer is $|a_1 - W|$.

Suppose that $2 \leq N$. X can achieve the final result of $|a_N - W|$ all cards in his first turn. Also, X can achieve the final result of $|a_{N-1} - a_N|$ by drawing $N - 1$ cards in his final turn.

It turns out that X can't do better than the two strategies above. Suppose that X leaves at least two cards ($k$ cards) in his first turn. Then, Y can take $k - 1$ cards in the next turn and the result will be $|a_{N-1} - a_N|$.

Therefore, the answer is $max(|a_N - W|, |a_{N-1} - a_N|)$.

# E: MUL

Let's solve this problem as a minimum S-T cut problem.

We divide all integers $1, 2, \ldots, N$ into two groups. Group S is a set of integers that are smashed, and Group T is a set of integers that are not smashed.

Ideally, we should smash all integers with negative costs, and we shouldn't smash any integers with non-negatove costs. Let $X$ be the total money you can get in this ideal situation.

However, this situation is not always possible. If for some $i, j$ such that $j$ is a multiple of $i$, $i$ is smashed and $j$ is not smashed, we get a contradiction. (Note that this is a sufficient condition: if a set of integers satisfies this condition, we can perform operations on all smashed integers and still we don't smash anything else.)

Now, you get the following penalties:

- If for some $i, j$ such that $j$ is a multiple of $i$, $i$ is in group S and $j$ is in group T, we get an infinite penalty.
- If $a_i \leq 0$ and $i$ is in group $T$, we get a penalty of $-a_i$.
- If $a_i > 0$ and $i$ is in group $S$, we get a penalty of $a_i$.

To represent these penalties, construct the following graph:

Construct a graph with $N + 2$ vertices. The vertices are labelled with $S, T, 1, 2, ..., N$, and for each $i$, we add the following edges:

- If $a_i \leq 0$, add an edge $S \to i$ with capacity $-a_i$.
- If $a_i > 0$, add an edge $i \to T$ with capacity $a_i$.
- For each $j = 2i, 3i, 4i, ...$, add an edge $i \to j$ with capacity $\infty$.

The answer is $X$ minus the minimum S-T cut of the graph above.

Since there are $O(N)$ vertices and $O(N \log N)$ edges, it works in time.

# F: NRE

In this problem, we want to minimize the number of $i$ such that $a_i \neq b_i$, that is, $(a_i, b_i) = (0, 1) or (1, 0)$.

The number of $i$ such that $(a_i, b_i) = (0, 1), (1, 0)$
= (The number of $i$ such that$(a_i, b_i) = (0, 1)$) + (The number of $i$ such that $b_i = 0$) - (The number of $i$ such that $(a_i, b_i) = (0, 0)$)

Here, since the number of $i$ such that $b_i = 0$ is a constant, we want to minimize (The number of $i$ such that$(a_i, b_i) = (0, 1)$) - (The number of $i$ such that $(a_i, b_i) = (0, 0)$)

In other words, when you paint the $i$-th cell ($a_i = 1$), there's no cost, and when you don't paint it ($a_i = 0$), you get a cost of 1 or $-1$.

Let's perform the operations and compute the cost in the following order:

Perform some operations with $l_i = 1$ and compute the cost of $a_1$
→ Perform some operations with $l_i = 2$ and compute the cost of $a_2$
→ Perform some operations with $l_i = 3$ and compute the cost of $a_3$
→ Perform some operations with $l_i = 4$ and compute the cost of $a_4$
→ ⋯

Here, after you compute the cost of $a_x$, you can forget about $a_1, a_2, ..., a_x$, and at this point, $a_{x+1}, a_{x+2}, ..., a_N$ is of the form "111...111000...000".

Therefore, we get the following DP, and it works in polynomial time. $dp(x, y)$ means the minimum possible cost you get from $a_1, \ldots, a_x$, given that at this point $a_{x+1}, \ldots, a_y = 1$ and the remaining elements are zeroes.

Now, how can we get the DP values of $(x+1, x+1), (x+1, x+2), (x+1, x+3), ..., (x, N)$ from $(x, x), (x, x+1), (x, x+2), ..., (x, N)$?

To handle an operation in the interval $[l(= x), r]$, we replace $(x + 1, r)$ with the minimum of $(x, x), (x, x + 1), ..., (x, r)$. It's easy to compute the cost of $a_x$ because when $a_x = 0$, you get no cost.

Therefore, the only data structure you need is to get the minimum of an interval, and this can be done in segment tree (or RMQ).