# SoundHound Inc. Programming Contest 2018 (春)

writer : hogloid

2018 年 1 月 27 日

## A: SoundHound

- C++ のコード例 : https://beta.atcoder.jp/contests/soundhound2018/submissions/2011758
- Python 3 のコード例 : https://beta.atcoder.jp/contests/soundhound2018/submissions/1991457

## B: 音量

毎秒、問題文通りの 3 通りの場合分けを行うことで解くことができます。また、$b_i = min(R, max(L, a_i))$ と表すこともできます。

- C++ のコード例 : https://beta.atcoder.jp/contests/soundhound2018/submissions/2011765
- Python 3 のコード例 : https://beta.atcoder.jp/contests/soundhound2018/submissions/1991479

## C: 広告

盤面をグリッドグラフ (頂点がマスで、隣接するマスの間に辺の張られたグラフ) と見なすと、この問題は

グリッドグラフでの最大独立集合のサイズを求めよ

という問題に言い換えられます。

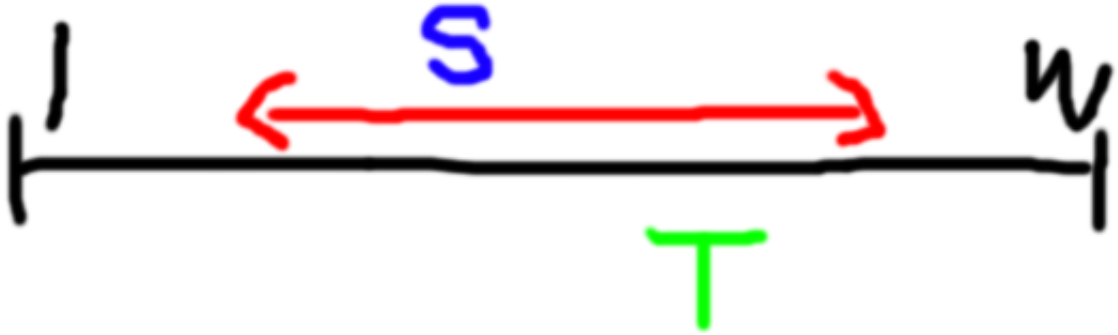
ところで、上下左右の 4 近傍に辺の張られたグリッドグラフは、二部グラフです。(∵ グリッドを市松模様に塗り分けると、異なる色の間にしか辺が張られないことが分かります)

二部グラフについて、最大独立集合のサイズは (頂点数) - (最大マッチングのサイズ) で求まることが知られています。なので、グリッドグラフを構築し、その上で最大マッチングを求めることで、答えが分かります。

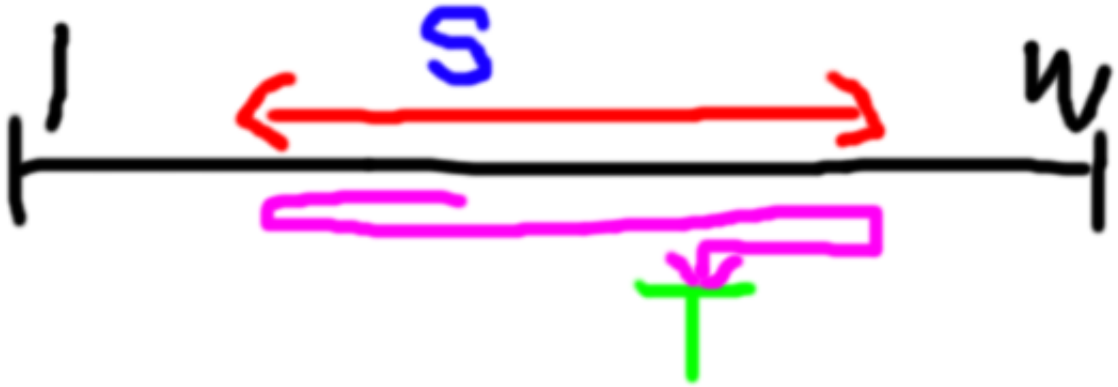
## D: 建物

DP を考えましょう。部屋 $(i, j)$ に初めて到達するときの利益の最大値を $dp_{i,j}$ とおきます。

まず、ある階での動きとしてどのようなものが考えられるか見てみましょう。$S$ 番目の部屋から始め、$T$ 番目の部屋から降りる (または出る) ときを考えます。$S \leq T$ を仮定します。お金を回収する (=通る) 区間を下図の赤線の範囲として考えましょう。




このとき、下図のピンク色の線のような順で部屋を訪れるのが最適です。




この移動を、$S$ を出てまた $S$ に戻ってくるまで、$S$ と $T$ の間、$T$ を訪れてからまた戻ってくるまで の 3 つに分割します。

一つ目の移動では、$S$ から左へいくつか行った後戻ります。お金を取る区間は被らないので、他 2 つの移動に関わらず、ここでの利益を最大化することが最適です。$S$ から始めるとき、$S$ にとどまる (左へは一切動かない) ケースと、左へ動き戻ってくるケースが考えられます。前者の利益は明らかで、後者の利益は、$S-1$ に到達し、そこから最適な一つ目の移動をした後、$S$ に戻ってくると最大となるので、結局前から順に求めることができます。この分の利益を $dp_{i,S}$ に足しこむことで、一つ目の移動抜きに残りを考えましょう。

三つ目の移動も、同様に後ろから順に求めることができます。これで、残りは二つ目の移動だけ考えれば大丈夫です。

二つ目の移動は、ただ右に進むだけなので、左から順に求めることができます。

よって、$S \leq T$ の場合の移動を考えることができました。$S > T$ の場合は、列を左右反転させて同じ処理を行うと楽です。

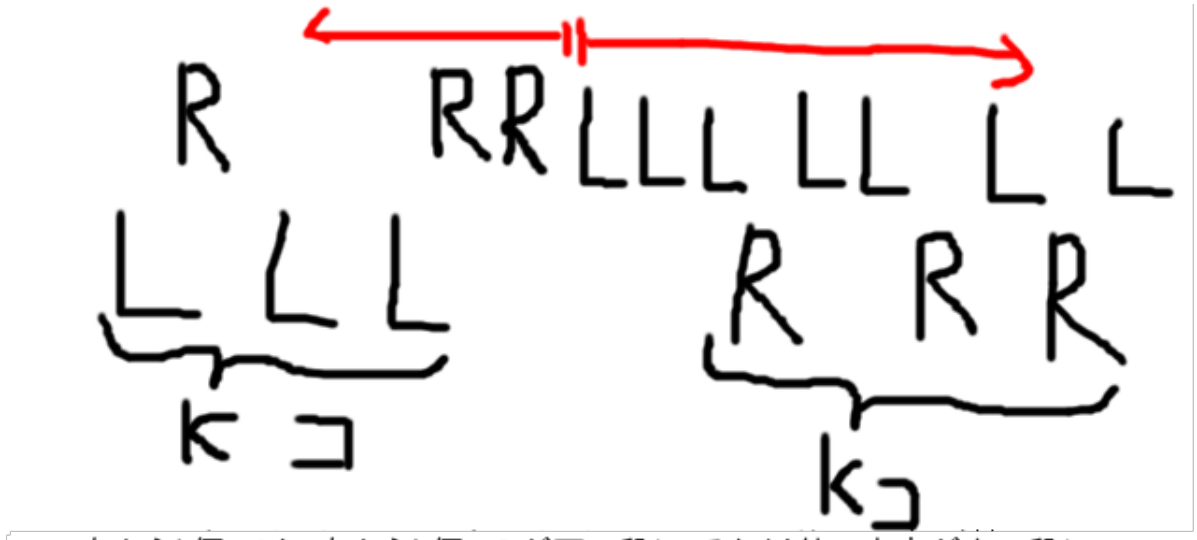
計算量は $\mathrm{O}(HW)$ です。

## E: カッコ列

以下の文では、(, ) を L, R で表します (文中に括弧があるとついそこで文を分けてしまいますよね？)。

左から $k$ 番目の L の位置が、右から $k$ 番目の R の位置より前であるようなもののうち、最も大きな $k$ を取ります。

このとき、括弧列は以下のような形になっています。




左からk個のLと、右からk個のRが下の段に、それ以外の文字が上の段に示されている。左右のk個の間には、RRLLといったようにR→Lの順で並んでいる(片方・両方ないこともありえる)

以下では、赤線で示された場所で、文字列を前半と後半に分けて呼ぶことにします。前半には $k$ 個の L があり、間に R がいくつかあります。後半は逆で、$k$ 個の R があり、間に L がいくつかあります。

答えとなる部分列に、左から $k$ 番目までの L と右から $k$ 番目までの R を含むとして構いません。もしそうでないとき、

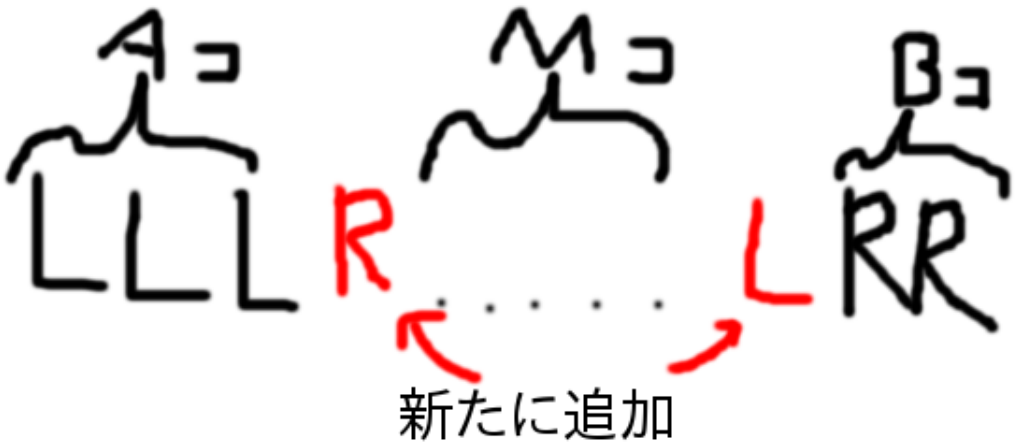
- 前半の L と後半の R のうち、取った数が違うとき
  前半の L の方が多いとします。このとき、前半で R を取っていることになりますが、そうするなら後半の取っていない R を代わりに入れたほうがスコアは高くなります。
- 前半の L と後半の R のうち、取った数が同じだが $k$ 未満のとき
  部分列に入れていない L と R をすべて入れることで、スコアは高くなります。

なので、これら $2k$ 個の文字は答えに含めるとして考えます。

残りは前半の R、後半の L です。これらを部分列に入れたほうがスコアが高くなる時もあります。R/L をそれぞれ $m$ 文字入れることにしましょう。

このとき、使う L/R は、前半と後半の境目に近い方から貪欲に取るのが最適です。これも、同様に取り替

えることによってスコアが上がることから分かります。




さらに、左から $A+1$ 文字目に R を、右から $B+1$ 文字目に L を新たに挿入し、その間に $M$ 文字あったとき、スコアは $A+B-(M+1)$ だけ上がります。また、この取り方によって prefix で L の数が R を下回ることもありません。なので、これが正の間中心から近いものを順に取ったものが、答えの部分列です。これにより、答えは左から順に L がいくつか連続し、その後 $S$ 上での連続した区間が続き、最後に R がいくつか連続する、という恰好になり、このスコアを求めることでクエリーに答えられます。

まず、$k$、$m$ を求めるのには二分探索を使います。ここで必要な操作は、ある場所より右/左で $i$ 番目の L/R の場所を求める、という操作です。これは、セグメント木を用いて $\mathrm{O}(\log n)$ で求めたり、二分探索 +BIT で $\mathrm{O}(\log^2 n)$ で求めることができます。答えを求めるには、L/R のある範囲での添字の値の総和を求めることができればよく、これはそのまま BIT で実現することができます。セグメント木を用いる場合、計算量は全体で $\mathrm{O}(N \log N + Q \log^2 N)$ になります。