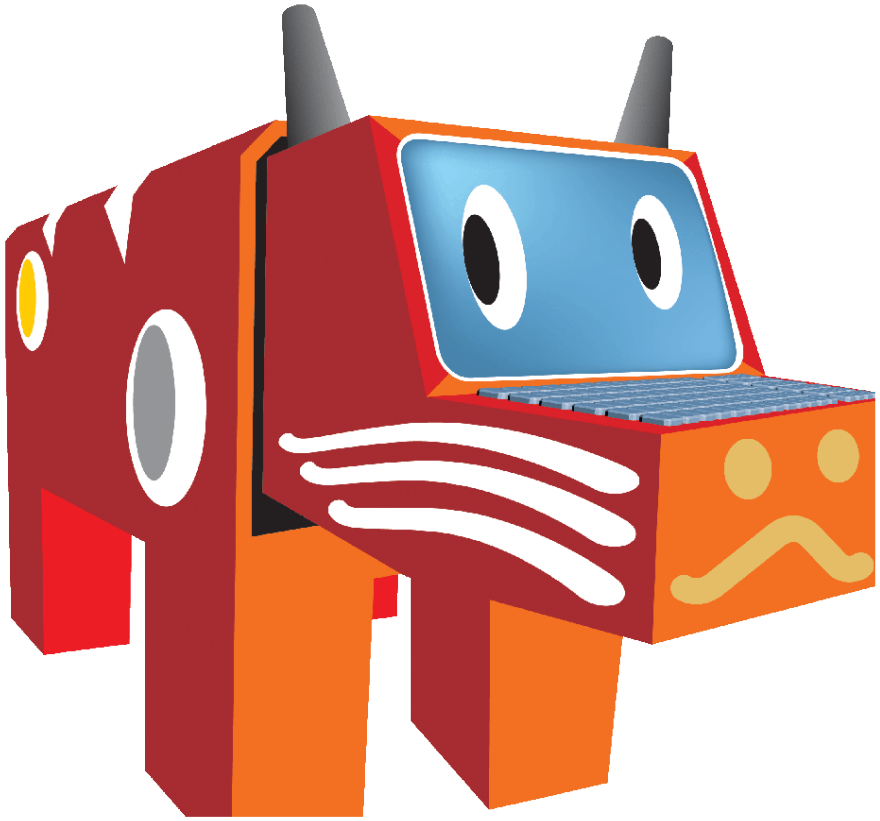
パソコン甲子園２０１９本選

プログラミング部門　解説



会津大学

2019年11月9-10日

概要

AIZU
THE UNIVERSITY OF AIZU

#	タイトル	概要	実装	思考	得点
1	目盛りのないストップウォッチ	基礎	☆	☆	2
2	ガソリンスタンド	シミュレーション	★☆	☆	4
3	海苔	幾何基礎	★	★☆	5
4	へびの脱皮	文字列・数学	★	★★	5
5	デジットK	貪欲法・データ構造	★★	★★☆	8
6	２つの多角形	動的計画法	★★☆	★★★	8
7	コンピュータシステムの不具合	動的計画法	★★☆	★★★	10
8	決まり事の多いジム	グラフ	★★☆	★★★★	10
9	山へ帰そう	グラフ	★★★☆	★★★	12
10	化学物質アルファ	平方分割	★★★	★★★☆	12
11	三角形の個数の和	数え上げ・整数	★★☆	★★★★	12
12	惑星ヤナイヅの資源	データ構造	★★★★	★★★★☆	12

問１　目盛りのないストップウォッチ

AIZU
THE UNIVERSITY OF AIZU

問題概要

- 一定の割合で時計回りに回転するストップウォッチがある.
- 目盛りがないので，起動からの経過時間を直接読み取ることができない.
- 時計回りに図った針の角度がa度のとき、起動からの経過時間がt秒になることがわかっている.
- aとt、図りたい角度rが与えられるので，その経過時間を出力せよ.
- $1 \leqq a \leqq 359$, $1 \leqq t \leqq 1000$, $0 \leqq r \leqq 359$.

問１　目盛りのないストップウォッチ

AIZU
THE UNIVERSITY OF AIZU

考察

- 1度あたりの秒数 t/a に，角度rをかければ経過時間が求まる.

解答例

```
#include<iostream>
#include<cstdio>
using namespace std;

main(){
  int a, t, r;
  cin >> a >> t >> r;
  printf("%.8lf¥n", (1.0*t/a)*r);
}
```

問2　ガソリンスタンド

AIZU
THE UNIVERSITY OF AIZU

問題概要

- N本のレーンがあるガソリンスタンドがある.
- 各レーンに車が列をなし，先頭の車が給油を行う.
- 給油を終了した車はレーンから出る.
- ガソリンスタンドに入ってきた車は，並んでいる車が最も少ないレーンの末尾に並ぶ. そのようなレーンが複数ある場合は，番号が一番小さいレーンに並ぶ.
- 車の入場と給油終了の情報が与えられるので，退場した車のナンバーを順番に出力せよ.
- $1 \leqq N \leqq 10$, $2 \leqq$ 情報の数 $\leqq 10{,}000$.

問2　ガソリンスタンド

AIZU
THE UNIVERSITY OF AIZU

解法

- キューをN個準備して、シミュレーションを行う.
- レーンの選択は，線形探索で十分.

問2 ガソリンスタンド

AIZU
THE UNIVERSITY OF AIZU

解答例

```
int N;
queue<int> L[10];

int findLane(){
  int res = 0;
  for ( int i = 0; i < N; i++ )
    if ( L[i].size() < L[res].size() ) res = i;
  return res;
}
main(){
  int Q, com, num;
  cin >> N >> Q;
  for ( int i = 0; i < Q; i++ ){
    cin >> com >> num;
    if ( com ){
      L[findLane()].push(num);
    } else {
      assert( L[num-1].size() > 0 );
      cout << L[num-1].front() << endl;
      L[num-1].pop();
    }
  }
}
```

問3　海苔

AIZU
THE UNIVERSITY OF AIZU

問題概要

- 軸に平行な２つの長方形が与えられる.
- 長方形の部分であって，重なっていない部分の面積を求めよ.
- 0≦x, y≦1000.
- 1≦w, h≦1000.

問3 海苔

AIZU
THE UNIVERSITY OF AIZU

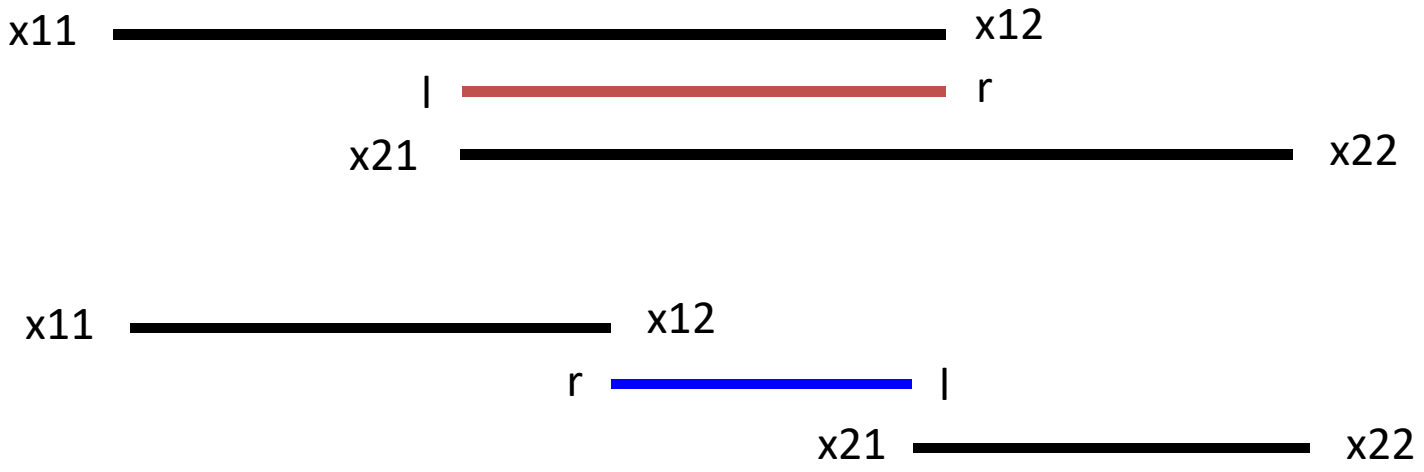
解法1

- 一つ目の長方形の面積をA，二つ目の長方形の面積をB，重なっている部分の面積をCとすると，重なっていない面積は A + B - 2C となる.
- Cを求めることを考える．一次元の問題に簡略化してみよう.
- 直線上にある，端点の位置がそれぞれ(x11, x12)，(x21, x22)である２つの線分が重なっている部分の長さを求める関数を定義する.

問3 海苔

解法1

- 図のように，重なっている部分の左端の位置をl，右端の位置をrとすると，
  - l は線分の左端の大きい方，つまりmax(x11, x21)
  - r は線分の右端の小さい方，つまりmin(x12, x22)
  - 重なっている部分の長さは r - l
  - l > r の場合は，重ならないので，0を返せばよい.
- この関数のx，y軸に対する積で，重なっている面積を求めることができる.



問3 海苔

AIZU
THE UNIVERSITY OF AIZU

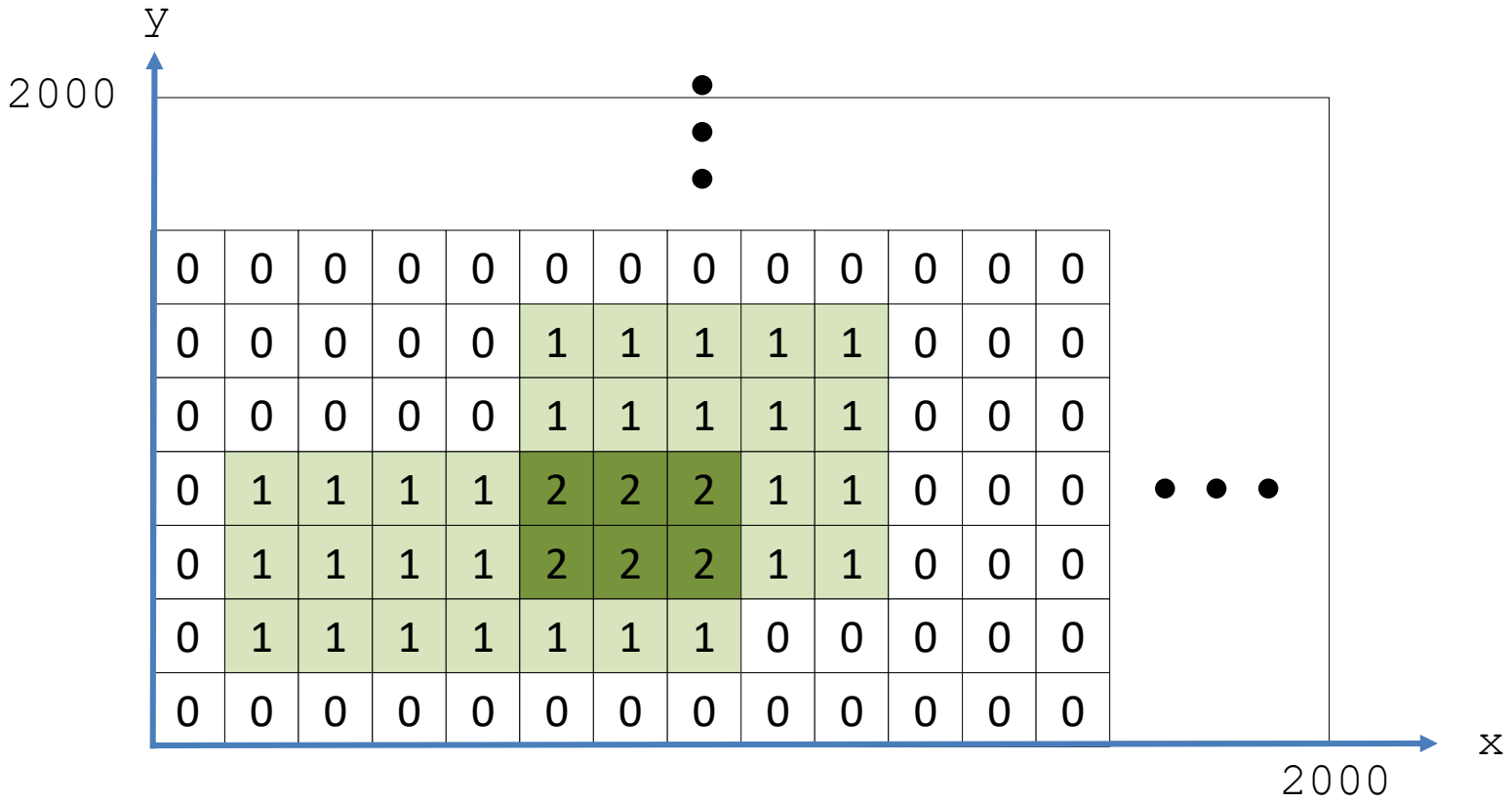
解答例1

```
int overlap(int x11, int x12, int x21, int x22){
  int l = max(x11, x21);
  int r = min(x12, x22);
  return ( l > r ) ? 0 : r - l;
}

main(){
  int x1, y1, w1, h1, x2, y2, w2, h2;
  cin >> x1 >> y1 >> w1 >> h1 >> x2 >> y2 >> w2 >> h2;
  int H = overlap(x1, x1+w1, x2, x2+w2);
  int V = overlap(y1, y1+h1, y2, y2+h2);
  cout <<  w1*h1 + w2*h2 - 2*H*V << endl;
}
```

問3　海苔

解法２

- $0 \leqq x \leqq 2000, 0 \leqq y \leqq 2000$ の領域を１×１のマスに分け、各マスに０を書き込む.
- それぞれの海苔について、海苔が乗っているすべてのマスに１を加える.
- １が書き込まれたマスの個数が求める面積.



問3 海苔

AIZU
THE UNIVERSITY OF AIZU

解法２

```
static const int MAX = 2000;
int G[MAX][MAX];

main(){
  int x1, y1, w1, h1, x2, y2, w2, h2;
  cin >> x1 >> y1 >> w1 >> h1 >> x2 >> y2 >> w2 >> h2;
  for ( int x = x1; x < x1+w1; x++ )
    for ( int y = y1; y < y1+h1; y++ ) G[x][y]++;
  for ( int x = x2; x < x2+w2; x++ )
    for ( int y = y2; y < y2+h2; y++ ) G[x][y]++;
  int ans = 0;
  for ( int x = 0; x < MAX; x++ )
    for ( int y = 0; y < MAX; y++ ) if (G[x][y] == 1 ) ans++;

  cout << ans << endl;
}
```

問4　へびの脱皮

AIZU
THE UNIVERSITY OF AIZU

問題概要

- へびをoとxからなる文字列で表す.
- １回脱皮すると、oが２つ連続する箇所がooxooに変化する.
  - oとoの間にoxoが挿入される
- へびを表す文字列と、整数Nが与えられるので、N回脱皮した後のへびの長さを求めよ.
- 1≦N≦50, 1≦ へびの長さ ≦100.

問4　へびの脱皮

AIZU
THE UNIVERSITY OF AIZU

考察

- へびの長さは、指数的に増える.
- 愚直に文字列を生成するシミュレーションではメモリ・計算量ともに厳しい.
- 文字列を生成する必要はない.

問4　へびの脱皮

AIZU
THE UNIVERSITY OF AIZU

解法

- oが連続する箇所の数は２倍になっていく.
- oxoの個数の累積数は初項１、等比２の等比数列の和になり，2^N-1になる.
- 挿入されるoxoは３文字.

N	ヘビの状態	○○の数	○x○の数	累積
0	○○	1	0	0
1	○○x○○	2	1	1
2	○○x○○x○○x○○	4	2	3
3	○○x○○x○○x○○x○○x○○x○○x○○	8	4	7

2^N-1

問4　へびの脱皮

AIZU
THE UNIVERSITY OF AIZU

解法

- へビに含まれるooの数をA，へビの長さをSとすると，N回脱皮した後のへびの長さは

$$A \times 3 \times (2^N-1) + S$$

解答例

```
main(){
  int L, N;
  string S;

  cin >> L >> N >> S;
  int oo = 0;
  for ( int i = 1; i < S.size(); i++ )
    if ( S[i-1] == 'o' && S[i] == 'o')  oo++;
  cout << oo*3*((1LL<<N)-1) + S.size() << endl;
}
```

問5　デジットK

AIZU
THE UNIVERSITY OF AIZU

問題概要

- 1から9までの数字を使ったN個の数字が並んでいる.
- ここからK個の数字を消去し，残ったN-K個の数字を左から順番に繋げてできる数の最大値を求めよ.
- 1≦N≦200,000
- 0≦K<N

1414231 (K=3)　→　1414231　→　4423

1199 (K=1)　→　1199　→　199

問5　デジットK

AIZU
THE UNIVERSITY OF AIZU

考察

- 1から9までのN個の数からなる数列からP(＝N-K)個選ぶ.
- 与えられた数列の中の数をkey、インデックスをidxとする.
- (key,-idx)を辞書式順序で大きい順に、選択可能なものを選んでいく.
- 選択可能なものは：
  - その時点で選択されていない
  - 最後に選択された数のidxよりも後にあるもの
  - 最終的にP桁の数が作れると保証できるidxのもの

問5 デジットK

解法１

- 例えば、N=7, P=4 の keys = {7 9 1 6 9 7 9}を考える

7 9 1 6 9 7 9	先頭からN-P個をPriorityQueue (PQ)に挿入
7 9 1 6 9 7 9	残りの先頭をPQに挿入
~~7~~ 9 1 6 9 7 9	PQの先頭を選択・削除しそれより前の要素も削除
~~7~~ 9 1 6 9 7 9	残りの先頭をPQに挿入
~~7~~ 9 ~~1 6~~ 9 7 9	PQの先頭を選択・削除しそれより前の要素も削除
~~7~~ 9 ~~1 6~~ 9 7 9	残りの先頭をPQに挿入
~~7~~ 9 ~~1 6~~ 9 7 9	PQの先頭を選択・削除しそれより前の要素も削除
~~7~~ 9 ~~1 6~~ 9 7 9	残りの先頭をPQに挿入
~~7~~ 9 ~~1 6~~ 9 7 9	PQの先頭を選択・削除しそれより前の要素も削除

問5 デジットK

AIZU
THE UNIVERSITY OF AIZU

解法 1

- O(N log N).

問5 デジットK

AIZU
THE UNIVERSITY OF AIZU

解答例 1

```
int N, K, P, A[N_MAX];
priority_queue<pair<int, int> > PQ;

main(){
  cin >> N >> K;
  P = N - K; // select P
  for ( int i = 0; i < N; i++ ) cin >> A[i];
  int last = N - P;
  int used = -1;
  for ( int i = 0; i < last; i++ ) PQ.push(make_pair(A[i], -i));
  for ( int p = 0; p < P; p++ ){
    PQ.push(make_pair(A[last], -last));
    last++;
    while( (-1)*PQ.top().second <= used ) PQ.pop();
    cout << PQ.top().first;
    used = PQ.top().second*(-1);
    PQ.pop();
  }
  cout << endl;
}
```

問5 デジットK

AIZU
THE UNIVERSITY OF AIZU

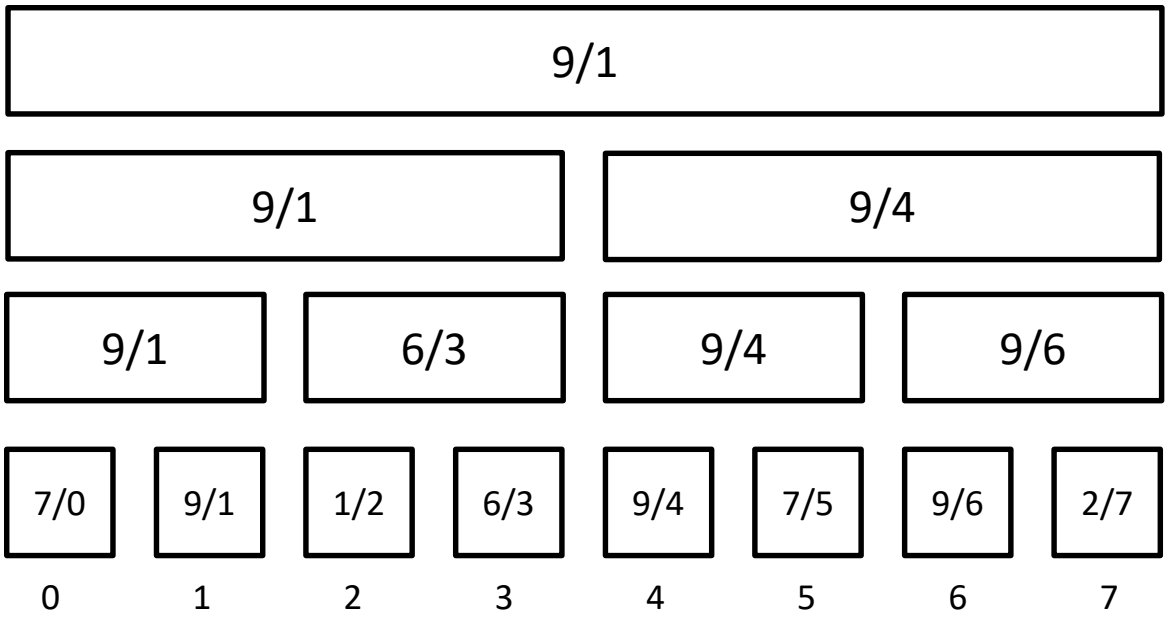
解法２

- セグメントツリー　（区間最大値）
  - 区間[l, r]の最大値を出力,最大値が同じ場合はインデックスの昇順
  - l: 最後に選択した要素の次
  - r: N-Pからインクリメント
- O(N log N).

問5 デジットK

解法２

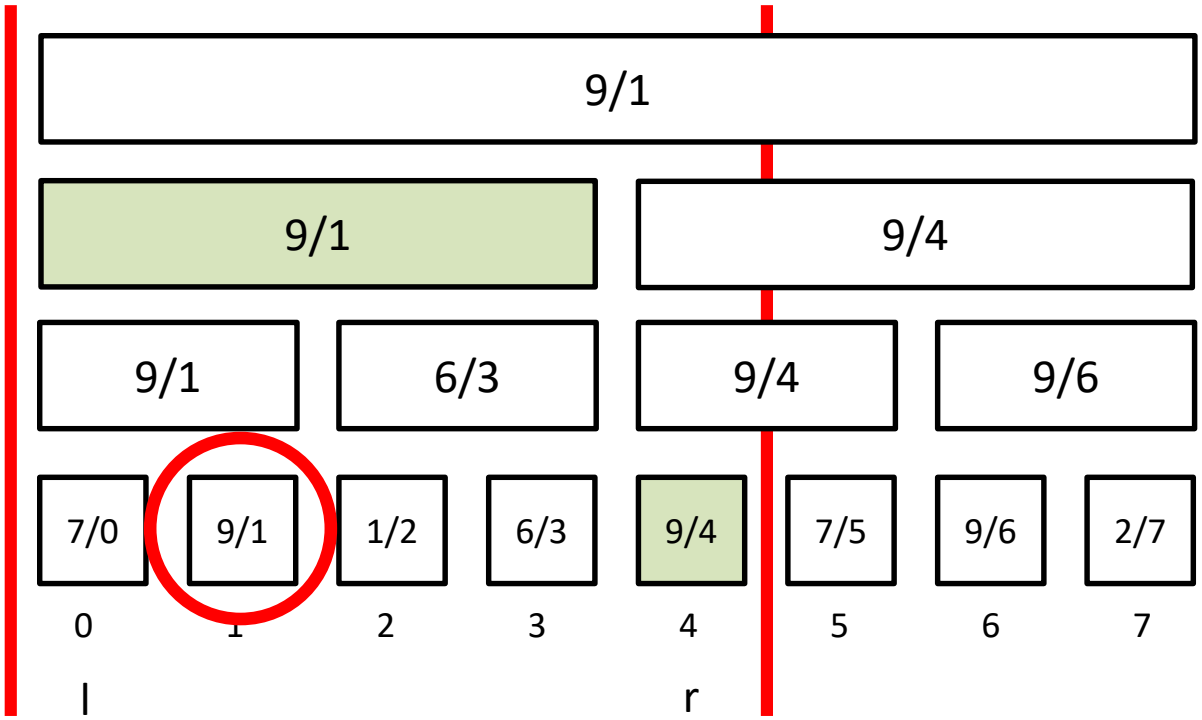
- 例えば、N=8, P=4 の keys = {7 9 1 6 9 7 9 2}を考える



問5 デジットK

解法２

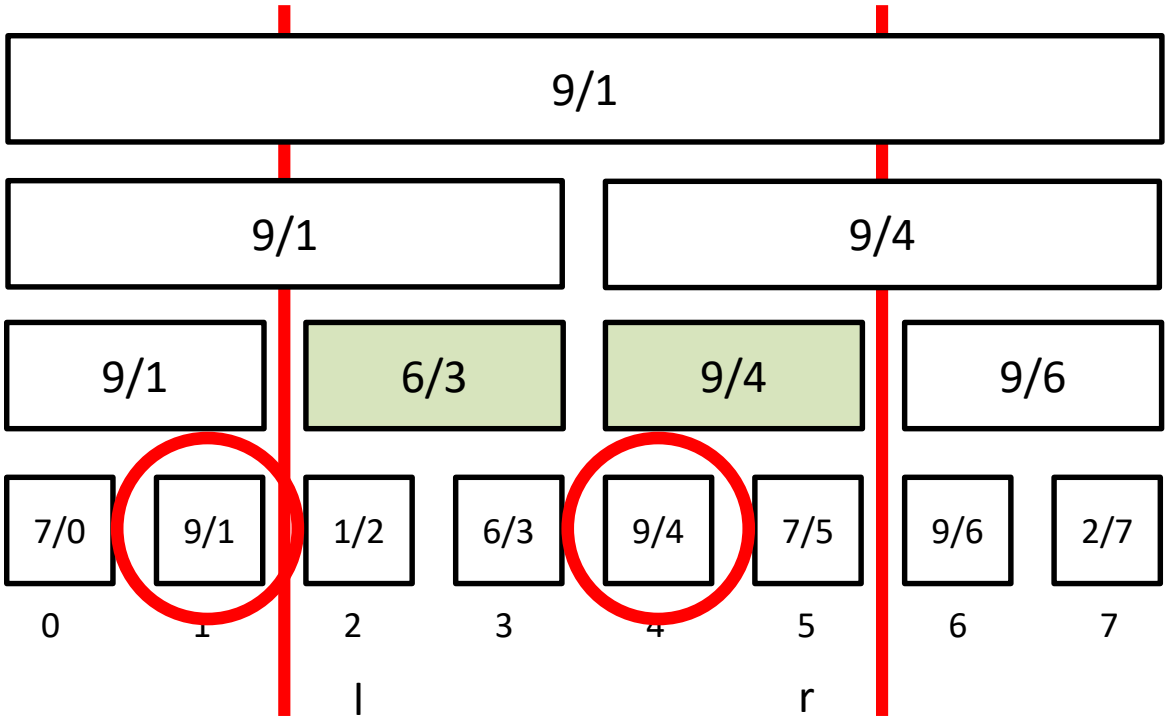
- 例えば、N=8, P=4 の keys = {7 9 1 6 9 7 9 2}を考える



問5 デジットK

解法２

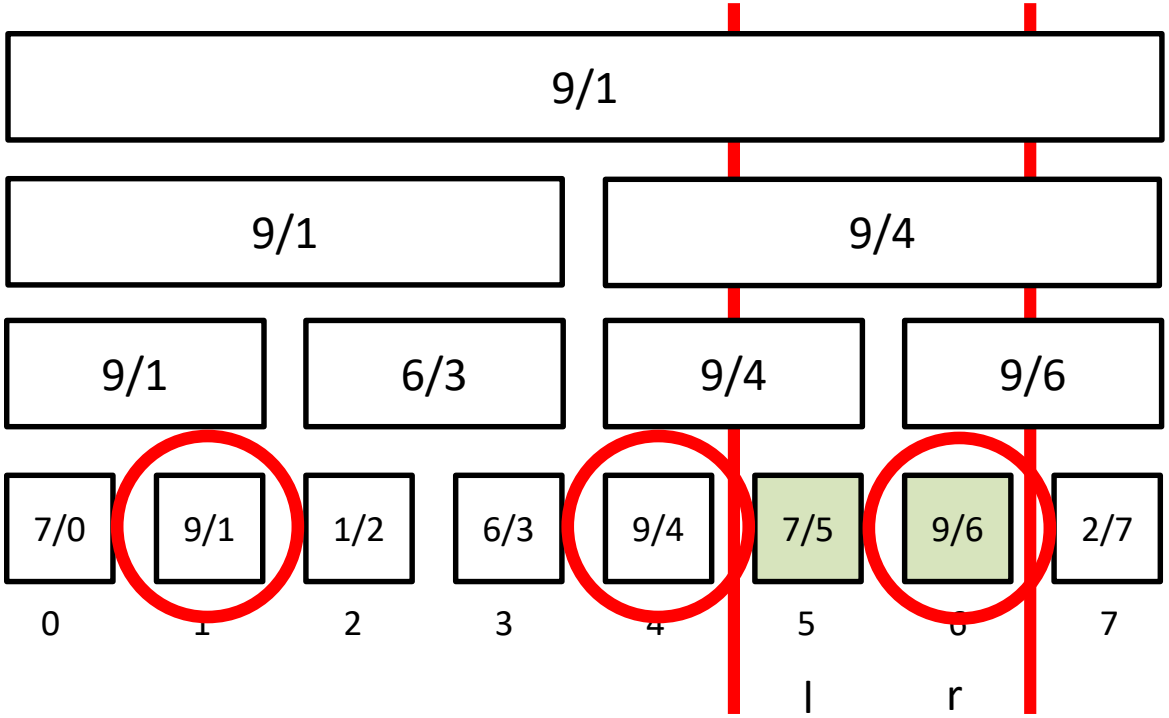
- 例えば、N=8, P=4 の keys = {7 9 1 6 9 7 9 2}を考える



問5 デジットK

解法２

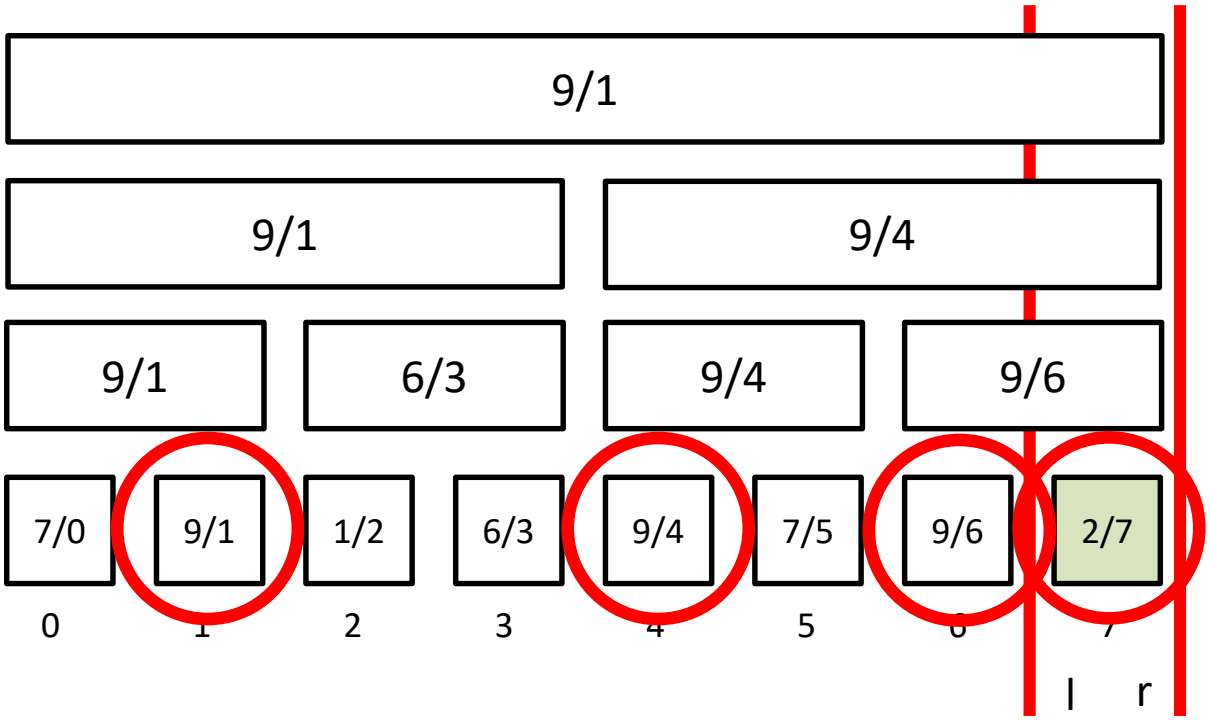
- 例えば、N=8, P=4 の keys = {7 9 1 6 9 7 9 2}を考える



問5 デジットK

解法２

- 例えば、N=8, P=4 の keys = {7 9 1 6 9 7 9 2}を考える



問6　2つの多角形

AIZU
THE UNIVERSITY OF AIZU

問題概要

- 長さがa_iのN本の棒が与えられる.
- N本すべてを使って、２つの多角形を作りたい
  - 面積が０になるものはNG
- ２つの多角形を、それらの周の長さの差が最小になるように作ったときの長さの差を求めよ．作成できない場合は-1を出力せよ.
- $6 \leqq N \leqq 40$，$1 \leqq a_i \leqq 100$.

問6　2つの多角形

AIZU
THE UNIVERSITY OF AIZU

考察

- 貪欲は難しそう？
- 棒の数、長さともに制約が小さい.
- それでも全探索は厳しそう.

- 基本方針：棒を適当に選んで、それらの長さの和がある整数Kになるようにできるかどうかを動的計画法で求める.

問6　2つの多角形

AIZU
THE UNIVERSITY OF AIZU

解法

- 1つ目の多角形をP1、2つ目の多角形をP2とする.
- P1の最大長の棒をbase1、P2の最大長の棒をbase2とする.
- base1は、全ての棒の中で最大のものに固定する.
- base2を残りの棒で決め打ちする.
- P2はbase2とbase2以下の長さの2本以上の棒でつくることになる.
- base2以下の棒で、整数Kが作れるかどうかを動的計画法で求める.
- $O(N^3L)$　（ただしLはbase1の長さ）

問6 2つの多角形

AIZU
THE UNIVERSITY OF AIZU

解答例

```
sort(L, L+N); reverse(L, L+N);
base1 = 0; // const
R1 = 0;
R2 = SUM - L[base1];
ans = INF;
for ( base2 = 1; base2 <= N-3; base2++ ){
  R2 -= L[base2];
  check();
  R1 += L[base2];
}
cout << (ans==INF?-1:ans) << endl;
```

問6　2つの多角形

AIZU
THE UNIVERSITY OF AIZU

解答例（つづき）

```
// dp[v]: L[base2+1] から最後までの要素を使ってvを作れる？
void check(){
  for ( int j = 0; j <= V_MAX; j++ ) dp[j][0] = dp[j][1] = false;
  dp[0][0] = true;
  for ( int i = base2+1; i < N; i++ ){
    int v = L[i];
    for ( int j = 0; j <= V_MAX; j++ )
      if ( dp[j][0] && j + v <= V_MAX ) dp[j+v][1] = true;
    for ( int j = 0; j <= V_MAX; j++ ){
      dp[j][0] |= dp[j][1];
      dp[j][1] = false;
    }
  }
  for ( int j = 0; j <= V_MAX; j++ ){
    if ( !dp[j][0] ) continue;  // 整数K
    int rem1 = R1 + (R2 - j);
    int rem2 = j;
    if ( L[base1] < rem1 && L[base2] < rem2 ){
      int d1 = L[base1] + rem1;
      int d2 = L[base2] + rem2;
      ans = min(ans, max(d1, d2) - min(d1, d2));
    }
  }
}
```

問７　コンピュータシステムの不具合

問題概要

- 各命令が1以上K以下の整数で表される長さNの命令列を考える.
- 命令列のk番目の命令をX_kで表したとき

$$X_i + X_{j-1} = X_j + X_{i-1}$$

を満たすとき、命令列に不具合が存在する

- N, K, M が与えられるので、不具合が起こる命令列の数をMで割った余りを求めよ.
- 3≦N≦100,000、 1≦K≦10、100,000,007≦M≦1,000,000,007.

問７　コンピュータシステムの不具合

考察

- $X_i + X_{j-1} = X_j + X_{i-1} \rightarrow X_i - X_{i-1} = X_j - X_{j-1}$
- 階差が等しいものがある組み合わせの数
- 命令列は全部でK^N通り
- 階差の種類数は2K-1個
- N-1が2K-1を超えると、同じ階差が必ず存在する
- N≦2Kのとき、階差が重複しない命令列を作ることができる
- 階差が重複しない命令列の数を求めて全体から引く

K^N – 階差が重複しない命令列の数

問７　コンピュータシステムの不具合

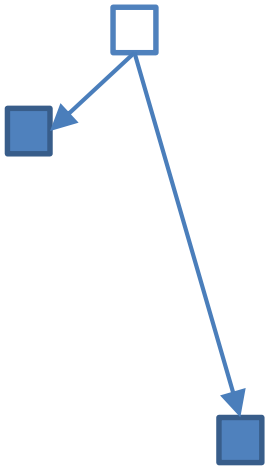
解法

- 階差の集合をbitで持つ動的計画法
- dp[s][n] = 階差集合sで命令列の最後の命令がnのときの、階差が重複しない命令列の数とする
- 階差の集合・最後の命令を基に、次の状態へ遷移する

K = 3 の例:

階差の集合					最後に選んだ命令		
2	1	0	-1	-2	1	2	3
...							
0	0	0	1	1			
0	0	1	0	0			
0	0	1	0	1			
0	0	1	1	0			
0	0	1	1	1			
0	1	0	0	0			
0	1	0	0	1			
0	1	0	1	0			
0	1	0	1	1			
0	1	1	0	0			
0	1	1	0	1			
...							

(2K − 1)個



問７　コンピュータシステムの不具合

解法

- 命令の数は状態として持つ必要がない.
- bitsetのcountによって、命令数が分かるので、countがN-1である dp[bit][k]を合計する
- $O(2^{(2K-1)}K^2)$

問７　コンピュータシステムの不具合

```
ll N, K;
MInt ALL;
MInt dp[1<<(2*K_MAX-1)][K_MAX+1];
int SHIFT;

main(){
  cin >> N >> K >> MOD;
  SHIFT = K-1;
  ALL = MInt(1); // K^N = all patterns
  for ( int i = 0; i < N; i++ ) ALL = ALL*K;
  for( int s = 1; s <= K; s++ ){
    for ( int t = 1; t <= K; t++ ){
      int bit = (t-s) + SHIFT;
      dp[1<<(bit)][t] = 1;
    }
  }
```

問７　コンピュータシステムの不具合


```
  for ( ll s = 0; s < (1<<(2*K-1)); s++ ){
    for (int l = 1; l <= K; l++ ){
      for ( int n = 1; n <= K; n++ ){
        int bit = (n - l) + SHIFT;
        if ( s & (1<<bit) ) continue; // already used
        ll ns = (s|(1<<bit));
        dp[ns][n] = dp[ns][n] + dp[s][l];
      }
    }
  }
  MInt OK = 0;
  for ( ll s = 0; s < (1<<(2*K-1)); s++ ){
    bitset<2*K_MAX> bs( s );
    if ( bs.count() == N-1 ){
      for ( int k = 1; k <= K; k++ ){
         OK = OK + dp[s][k];
      }
    }
  }
  MInt ans = ALL - OK;
  cout << ans.x << endl;
}
```

問8　決まりごとの多いジム

AIZU
THE UNIVERSITY OF AIZU

問題概要

- 1からNまでの番号が付いたN台のトレーニング機器がある。
- トレーニング機器を１回利用するには、その機器専用のチケットが１枚必要。
- トレーニング機器を１回利用したときの消費カロリーが、機器ごとに決まっている。
- 機器の利用回数にルールがある：
  - 「a番の機器はb番の機器よりもc回以上多く使ってはならない」
- 持っているチケットを使って、ルールで許される範囲でなるべく多くのカロリーを消費せよ。
- 1≦N≦100,000、　0≦ルールの数≦100,000

問8　決まりごとの多いジム

AIZU
THE UNIVERSITY OF AIZU

考察

- 例題

#	t	e
1	5	1
2	6	2
3	2	3
4	7	1

a	b	c
1	2	4
2	1	3
1	3	2
3	2	3
3	4	2

問8　決まりごとの多いジム

AIZU
THE UNIVERSITY OF AIZU

考察

- 例題

#	t	e
1	**5**	1
2	**6**	2
3	**2**	3
4	**7**	1

a	b	c
1	2	4
2	1	3
1	3	2
3	2	3
3	4	2

各機器を使える最大の回数をm_iと書く

ルールを何も考えなければ当然すべてのチケットを使うべき

$\{m_1\ m_2\ m_3\ m_4\} = \{t_1\ t_2\ t_3\ t_4\} = \{$**5　6　2　7**$\}$

問8　決まりごとの多いジム

AIZU
THE UNIVERSITY OF AIZU

考察

- 例題

#	t	e
1	5	1
2	6	2
3	2	3
4	7	1

a	b	c
1	**2**	**4**
2	1	3
1	3	2
3	2	3
3	4	2

各機器を使える最大の回数をm_iと書く

ルールを何も考えなければ当然すべてのチケットを使うべき

$\{m_1\ m_2\ m_3\ m_4\} = \{t_1\ t_2\ t_3\ t_4\} = \{5\ \ 6\ \ 2\ \ 7\}$

ルール1：**1 2 4** $\Rightarrow\ m_1 < m_2 + 4$

m_iは整数なので $m_1 \leqq m_2 + 3$ と書ける

問8　決まりごとの多いジム

AIZU
THE UNIVERSITY OF AIZU

考察

- 例題

#	t	e
1	5	1
2	6	2
3	2	3
4	7	1

a	b	c
1	2	4
2	**1**	**3**
1	**3**	**2**
3	**2**	**3**
3	**4**	**2**

各機器を使える最大の回数をm_iと書く

ルールを何も考えなければ当然すべてのチケットを使うべき

$\{m_1\ m_2\ m_3\ m_4\} = \{t_1\ t_2\ t_3\ t_4\} = \{5\ \ 6\ \ 2\ \ 7\}$

ルール1：**1 2 4** $\Rightarrow\ m_1 < m_2 + 4$

m_iは整数なので $m_1 \leqq m_2 + 3$ と書ける

ルール2：**2 1 3** $\Rightarrow\ m_2 \leqq m_1 + 2$

ルール3：**1 3 2** $\Rightarrow\ m_1 \leqq m_3 + 1$

ルール4：**3 2 3** $\Rightarrow\ m_3 \leqq m_2 + 2$

ルール4：**3 4 2** $\Rightarrow\ m_3 \leqq m_4 + 1$

問8　決まりごとの多いジム

考察

- 例題

#	t	e
1	5	1
2	6	2
3	2	3
4	7	1

a	b	c
1	2	4
2	**1**	**3**
1	**3**	**2**
3	**2**	**3**
3	**4**	**2**

各機器を使える最大の回数をm_iと書く

すべてまとめると

$m_1 \leqq 5$
$m_2 \leqq 6$
$m_3 \leqq 2$
$m_4 \leqq 7$

} 持っているチケットの枚数自体も制約とみなせる

$m_1 \leqq m_2 + 3$
$m_2 \leqq m_1 + 2$
$m_1 \leqq m_3 + 1$
$m_3 \leqq m_2 + 2$
$m_3 \leqq m_4 + 1$

という不等式制約の下

$$\text{maximize}\left(\sum_{i=1}^{N} m_i e_i\right)$$

を解く(e_iは機器iの消費カロリー)

問8　決まりごとの多いジム

考察

- 例題

#	t	e
1	5	1
2	6	2
3	2	3
4	7	1

a	b	c
1	2	4
2	**1**	**3**
1	**3**	**2**
3	**2**	**3**
3	**4**	**2**

各機器を使える最大の回数をm_iと書く

すべてまとめると

$$m_1 \leqq 5$$
$$m_2 \leqq 6$$
$$m_3 \leqq 2$$
$$m_4 \leqq 7$$
$$m_1 \leqq m_2 + 3$$
$$m_2 \leqq m_1 + 2$$
$$m_1 \leqq m_3 + 1$$
$$m_3 \leqq m_2 + 2$$
$$m_3 \leqq m_4 + 1$$

という不等式制約の下

$$\text{maximize}\left(\sum_{i=1}^{N} m_i e_i\right)$$

を解く(e_iは機器iの消費カロリー)

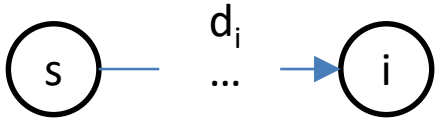
線形最適化問題だが、このような2変数間の制約の場合、グラフにできる.

問8　決まりごとの多いジム

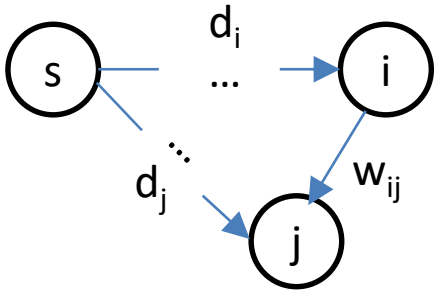
AIZU
THE UNIVERSITY OF AIZU

考察

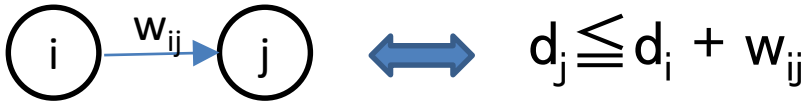
- 始点がsのグラフで、節点iへの最短距離がd_iだとする
  （複数の節点を経由することもあることに注意）.



- 節点jへの最短距離がd_jで、iとjを直接結ぶ辺の距離がw_{ij}だとする.



- d_jはd_i+w_{ij}以下である. このようなiからjへの有向辺は不等式で表せる.



問8　決まりごとの多いジム

AIZU
THE UNIVERSITY OF AIZU

解法

始点sを作り、同じ形にすれば以下のような有向辺が張れる

$m_1 \leqq S + 5$

$m_2 \leqq S + 6$

$m_3 \leqq S + 2$

$m_4 \leqq S + 7$

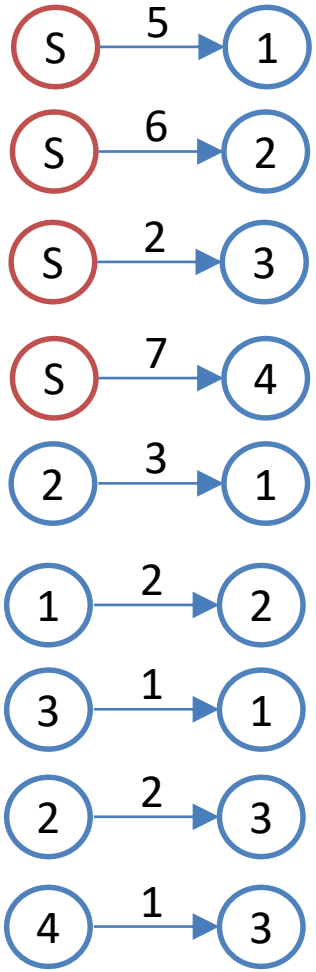
$m_1 \leqq m_2 + 3$

$m_2 \leqq m_1 + 2$

$m_1 \leqq m_3 + 1$

$m_3 \leqq m_2 + 2$

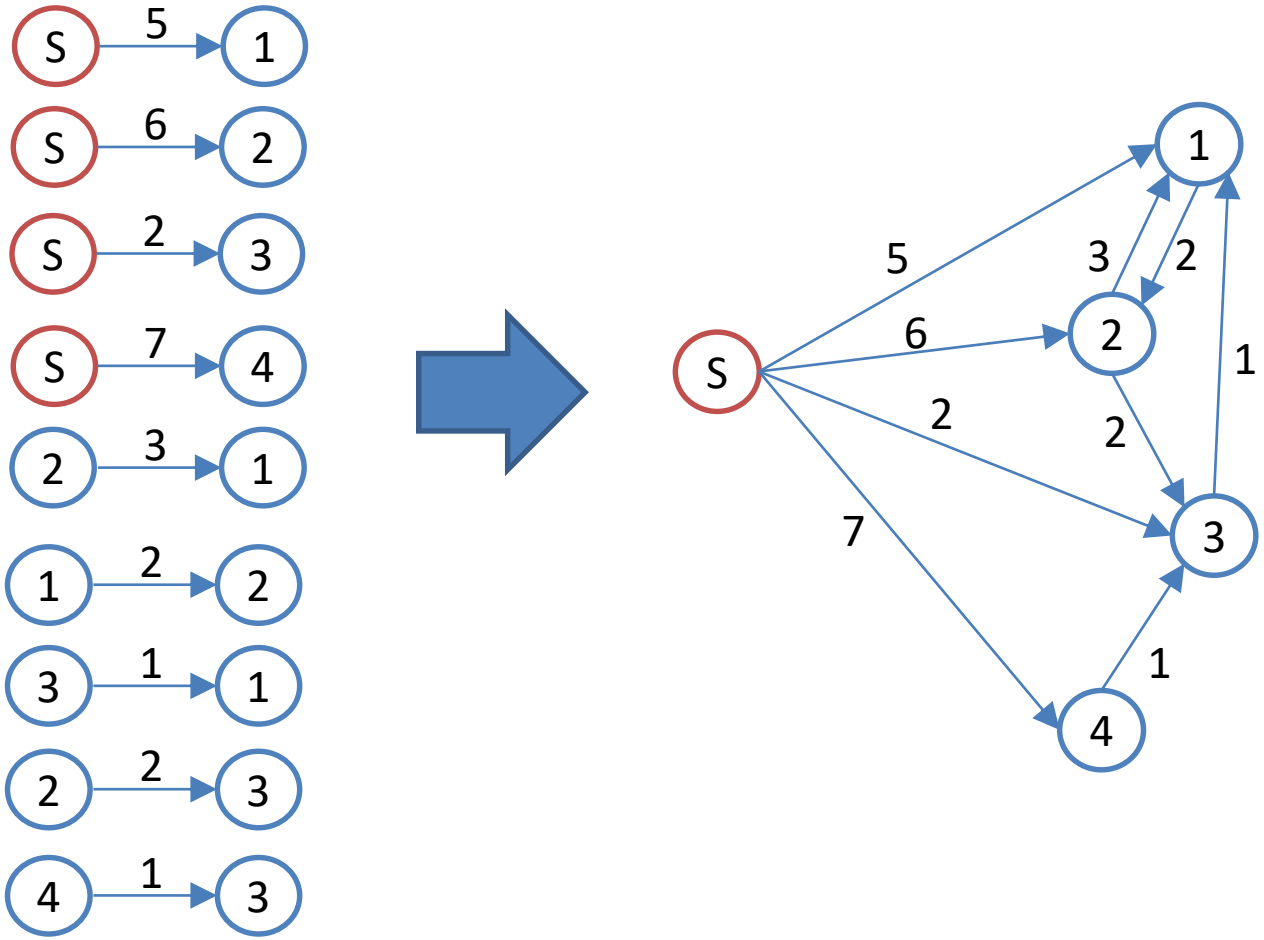
$m_3 \leqq m_4 + 1$



問8　決まりごとの多いジム

解法

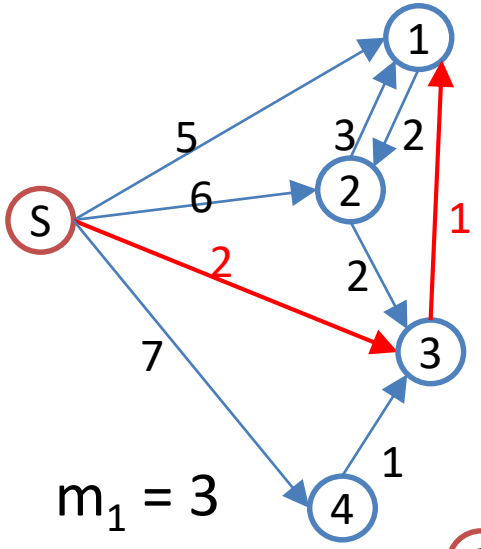
- すべての辺をまとめてグラフを作る
  （ルールがない場合でも、少なくともSとすべてのノードが直接つながる）



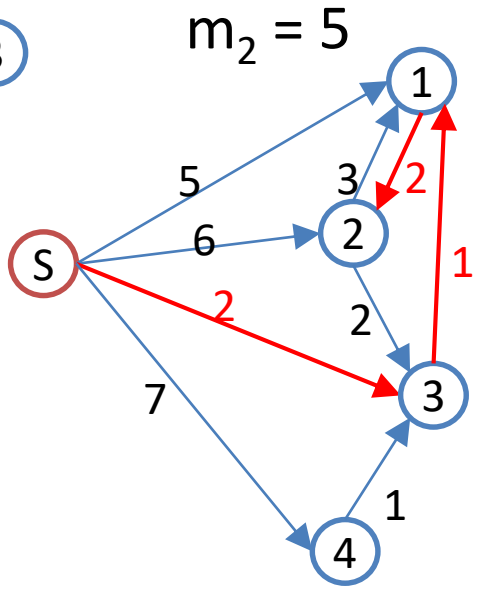
問8　決まりごとの多いジム

解法

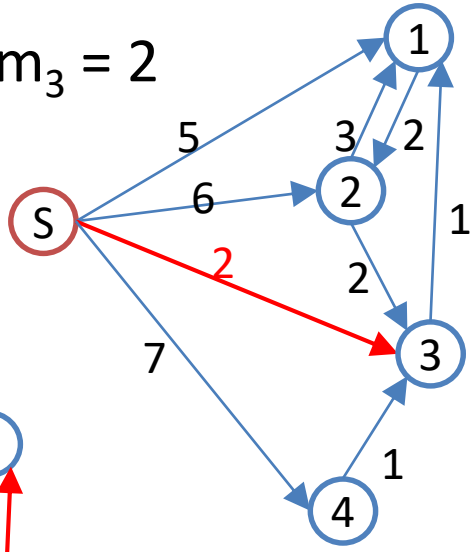
- Dijkstraで最短経路を計算したときの最短距離が求めたいm_iになる.
- O((N+R)logN).



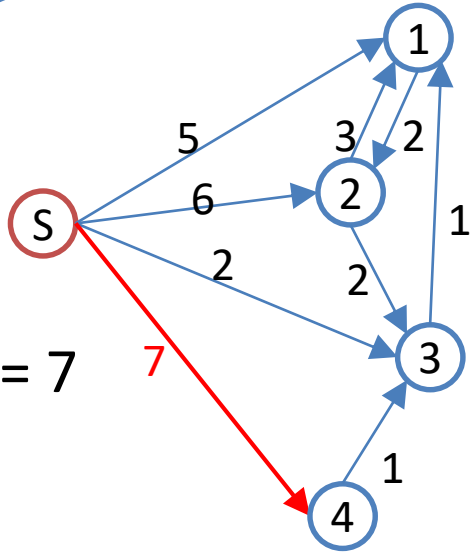
$m_1 = 3$



$m_2 = 5$



$m_3 = 2$



$m_4 = 7$

問8　決まりごとの多いジム

AIZU
THE UNIVERSITY OF AIZU

解答例

```
for ( int i = 1; i < n; i++ ){
   cin >> t[i] >> e[i];
   adj[0].push_back(make_pair(i, t[i]));
 }
 for ( int i = 0; i < m; i++ ){
   cin >> a >> b >> c;
   adj[b].push_back(make_pair(a, c-1));
 }

 dijkstra(0);//ダイクストラ法O(ElogV)を始点Sから

 ll ans = 0;
 for ( int i = 1; i < n; i++ ){
   ans += d[i]*e[i];
 }
 cout << ans << endl;
```

問9　山へ帰そう

AIZU
THE UNIVERSITY OF AIZU

問題概要

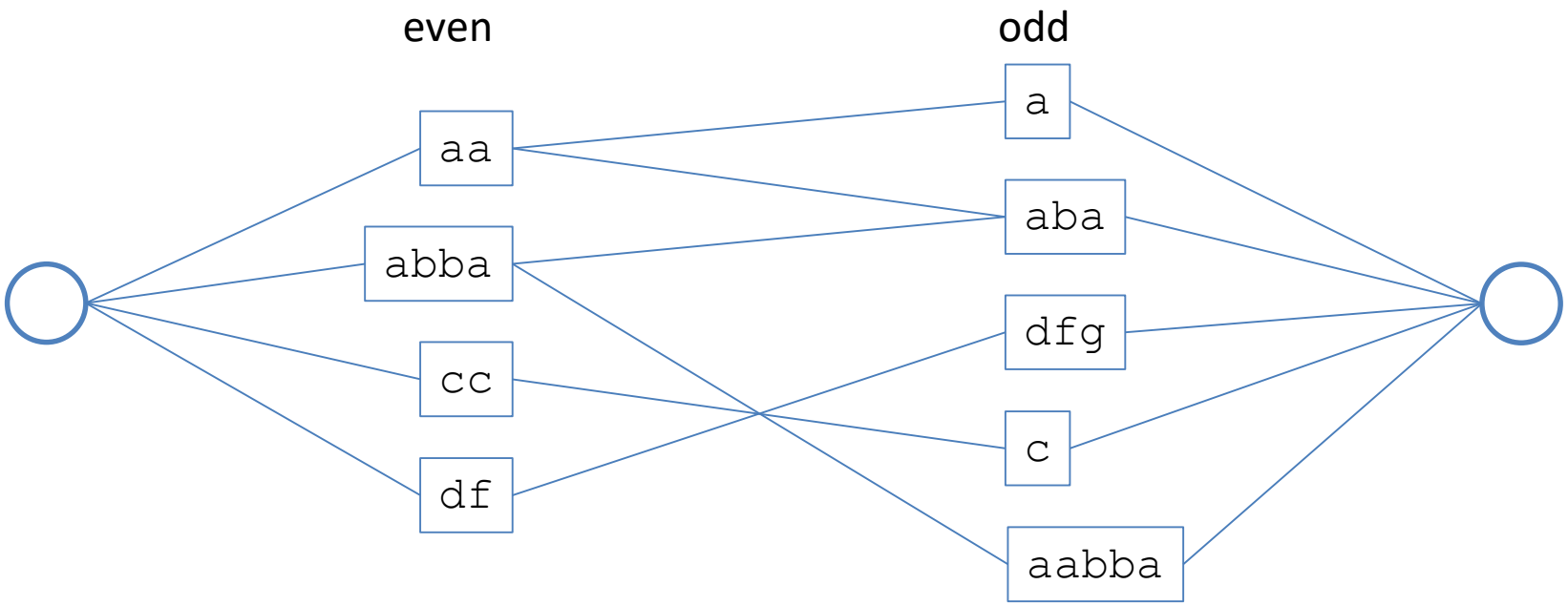
- N匹の動物がいる.
- それぞれの動物には固有の名前がついている。
- ２体の動物について、一方の動物の名前のどこかの位置に１文字を挿入すると、もう一方の動物の名前と一致する場合、これらをペアにすることができる。
- 作ることのできるペアの数の最大値を求めよ。
- 2≦N≦100,000、 1≦名前の長さ≦10

問9　山へ帰そう

AIZU
THE UNIVERSITY OF AIZU

考察

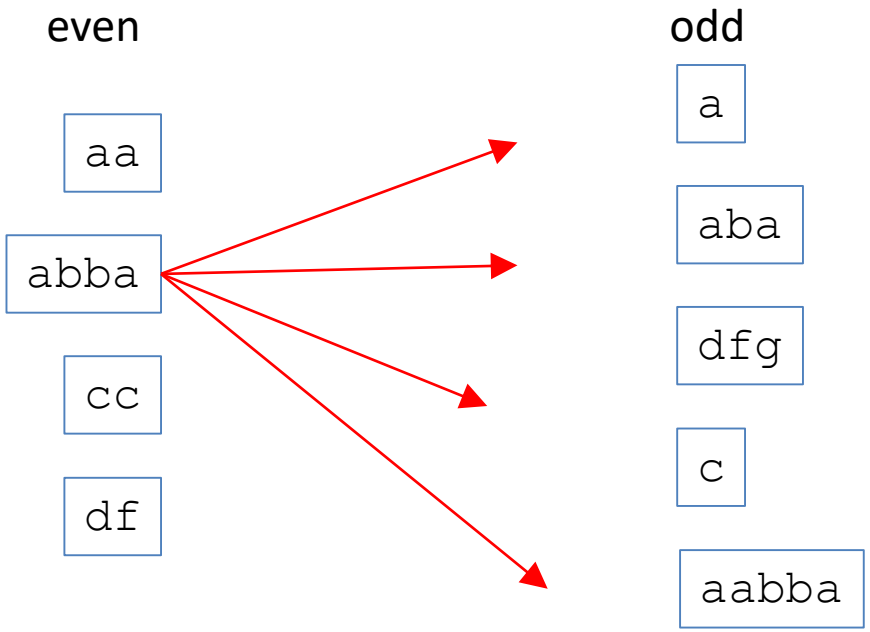
- 文字列はその長さの偶奇で分類でき、二部マッチングに帰着できる。



問9　山へ帰そう

考察

- 片方の文字列の適当な位置に1文字挿入すると、もう片方の文字列と一致する。
- 片方の文字列の適当な位置を1文字削除すると、もう片方の文字列と一致する。



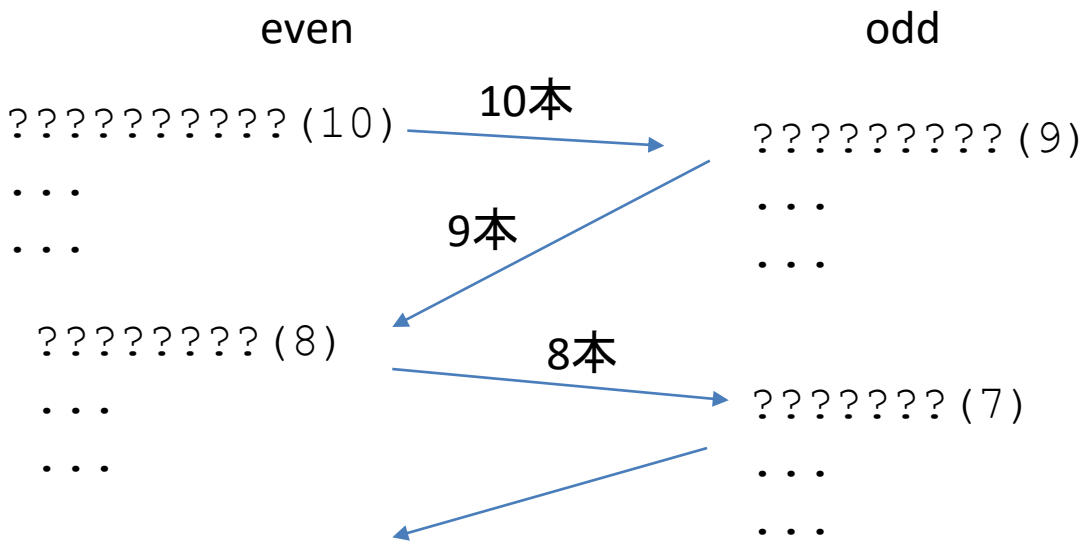
二分探索/map/set ...

※両側から

問9　山へ帰そう

考察

- Ford-FulkersonでO(|S| N^2)
- DinicでO(|S| N $\sqrt{N}$)
- 2文字以上の文字列から１文字削除したものと同じ文字列が存在するときだけ、それらの間に辺を張ればよく、最も長い文字列が１０文字であるから、辺の本数は高々10Nとなる、辺の数はO(|S|N)



問10　科学物質アルファ

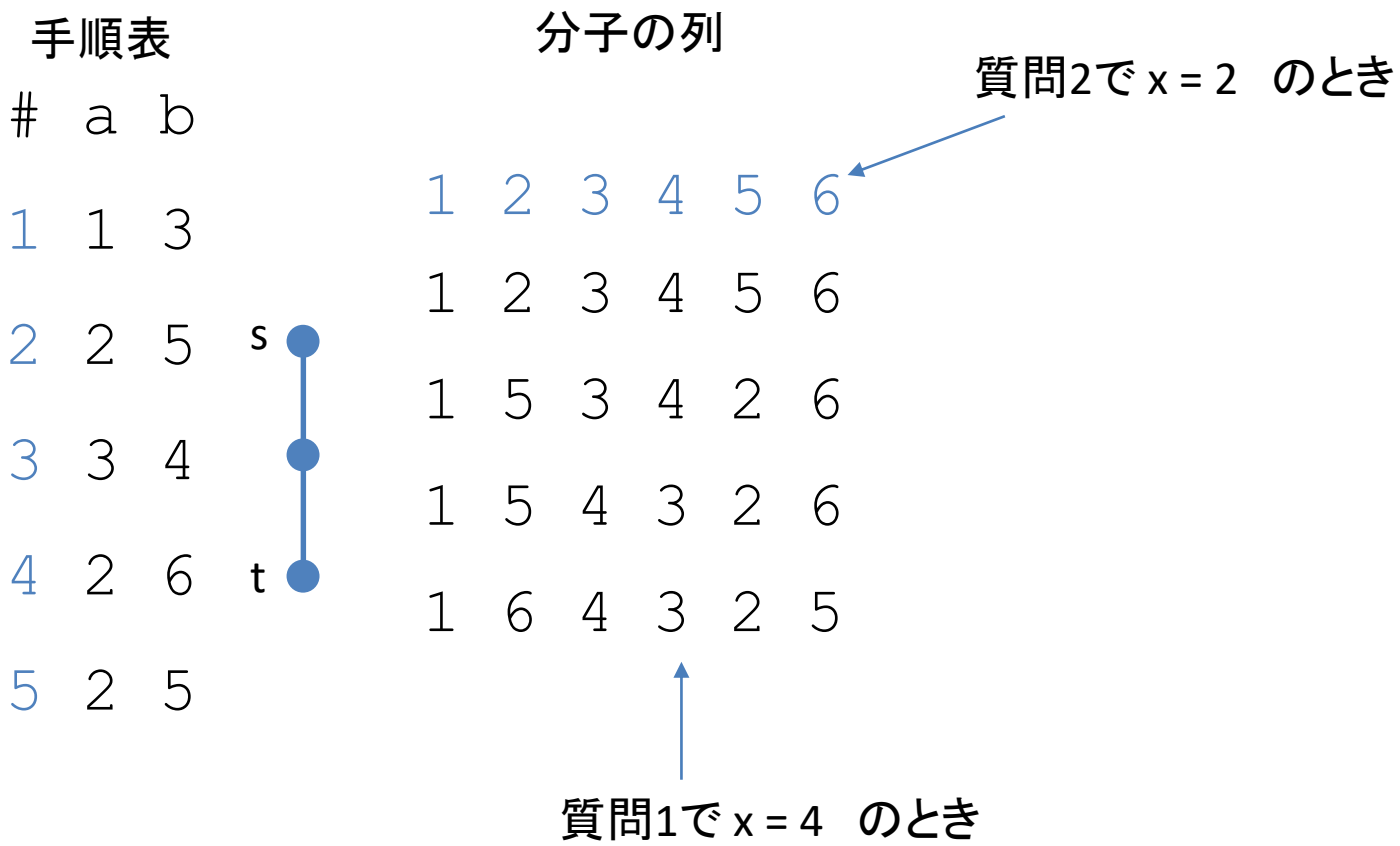
AIZU
THE UNIVERSITY OF AIZU

問題概要

- 1番からN番の分子が並んだ数列A = {1, 2, ... N} の要素をスワップするシミュレーションを行う。
- スワップは決まった手順表の始点から終点を選んだスワップのリストで行う.
- 固定の手順表が１つ与えられる。シミュレーションを何度か行ったとき、各シミュレーションでの分子の位置について以下の質問に答えよ。
  - (1) 最初にi 番目に位置していた分子が終了後にどの位置に来ているか。
  - (2) 終了後に左端からi番目に位置している分子は、最初は何番目に位置していたか。
- ただし、各シミュレーションは、初期状態（1番からN番の分子が左端から右端へ直線状にならんだ状態）から始めるものとする。

問10　科学物質アルファ

問題概要



問10　科学物質アルファ

AIZU
THE UNIVERSITY OF AIZU

考察

分子の列

	1	2	3	4	5	6
P	1	2	3	4	5	6

`P[i] = i` として初期化

分子の位置

	1	2	3	4	5	6
R	1	2	3	4	5	6

`R[P[i]] = i` なる配列

例:

	1	2	3	4	5	6
P	3	6	5	4	1	2

	1	2	3	4	5	6
R	5	6	1	4	3	2

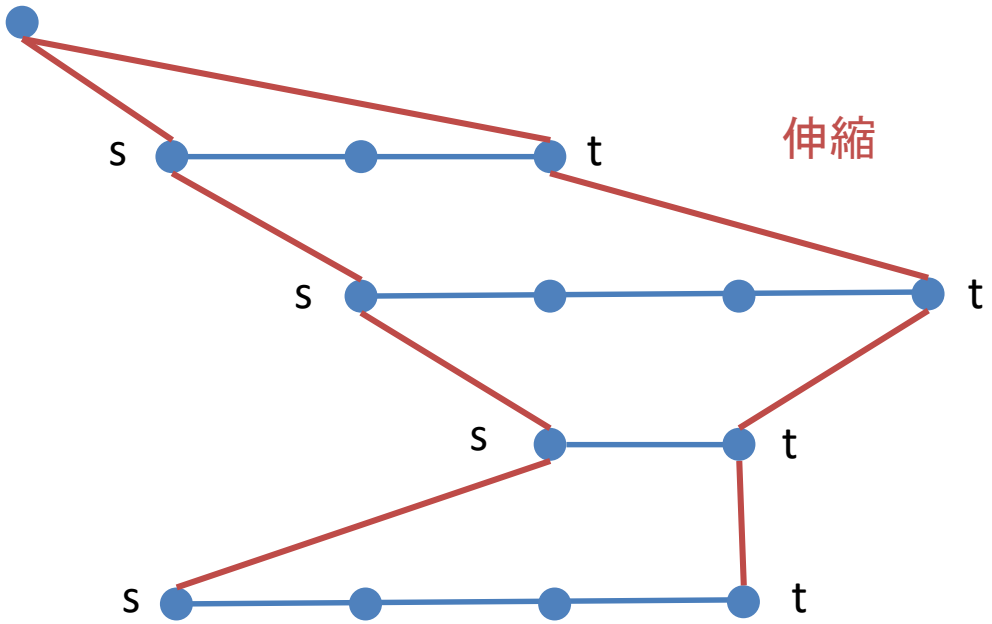
問10　科学物質アルファ

AIZU
THE UNIVERSITY OF AIZU

考察

手順表

	1	2	3	4	5
a	1	2	3	2	2
b	3	5	4	6	5



1	2	3	4	5	6
?	?	?	?	?	?
?	?	?	?	?	?
?	?	?	?	?	?
?	?	?	?	?	?

問10　科学物質アルファ

AIZU
THE UNIVERSITY OF AIZU

考察

伸縮

末尾に追加

swap(R[P[a]], R[P[b]])

末尾を削除

swap(P[a], P[b])

先頭に追加

swap(P[R[a]], P[R[b]])

先頭を削除

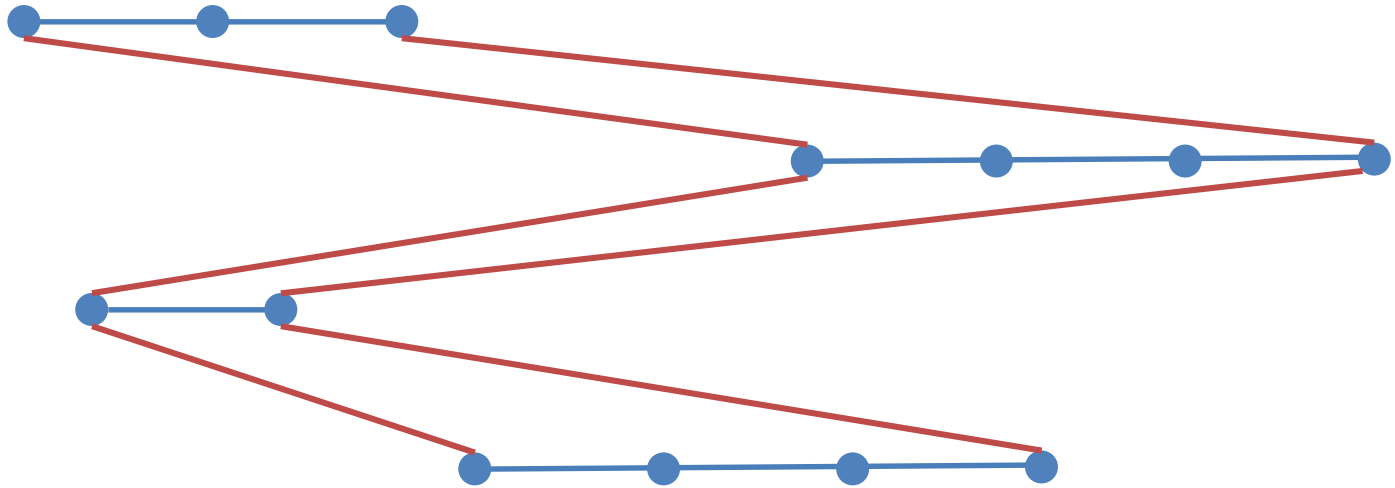
swap(R[a], R[b])

問10　科学物質アルファ

AIZU
THE UNIVERSITY OF AIZU

考察

- PとRを使って、1回の伸縮はO(1), 伸縮の幅はO(K)
- O(QK) → TLE

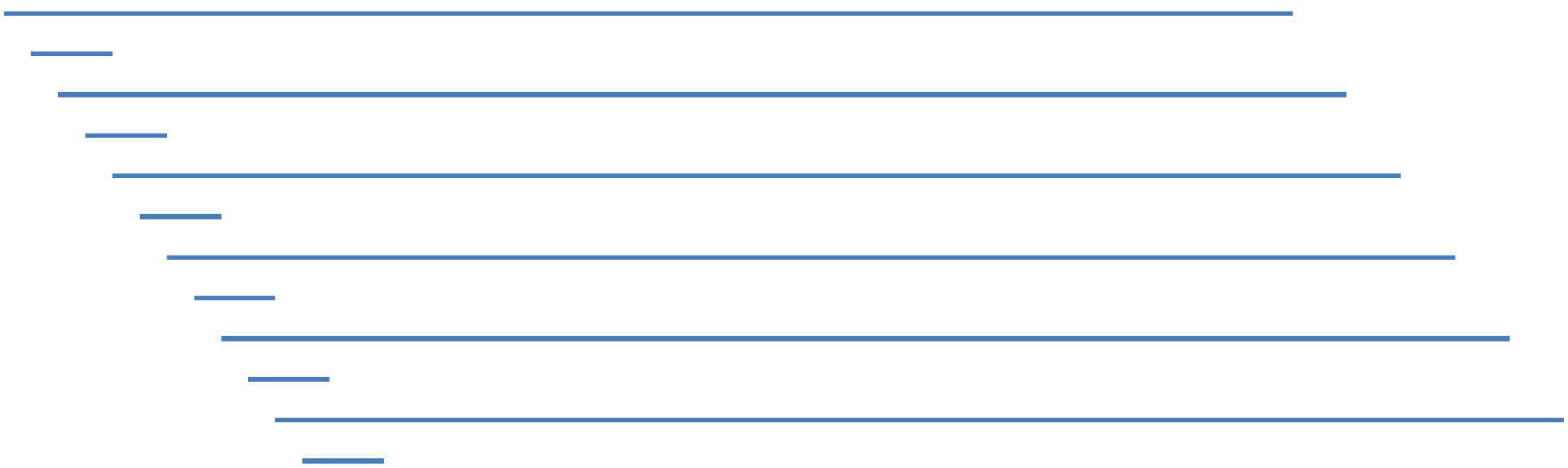


問10　科学物質アルファ

AIZU
THE UNIVERSITY OF AIZU

考察

- クエリを先読みして、s, t を基準にソートは？
- コーナーケースが存在する
- O(QK) → TLE

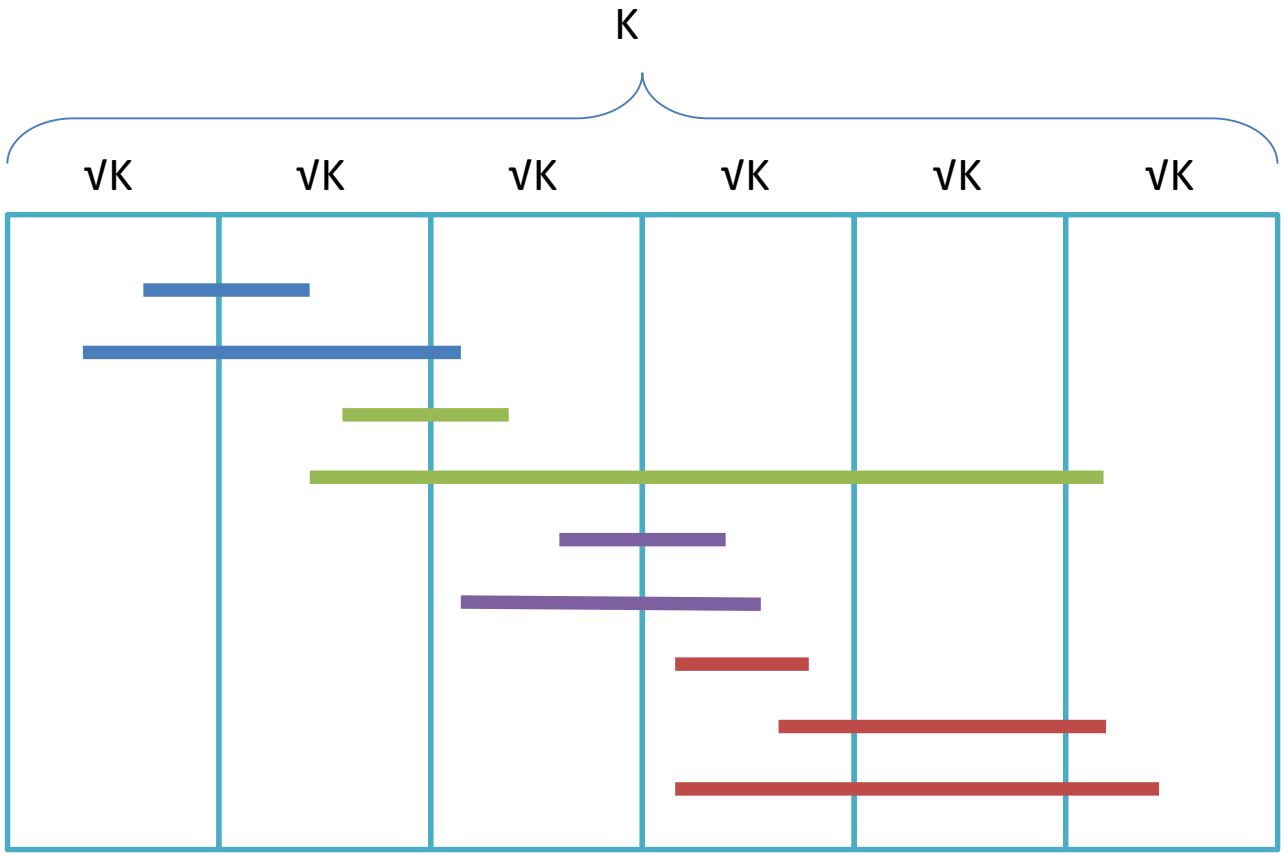


問10　科学物質アルファ

AIZU
THE UNIVERSITY OF AIZU

解法

- クエリを先読みして、平方分割+ソート
- 左端は区画のバケットに入れる．区画が同じ場合は右端の座標でソート.

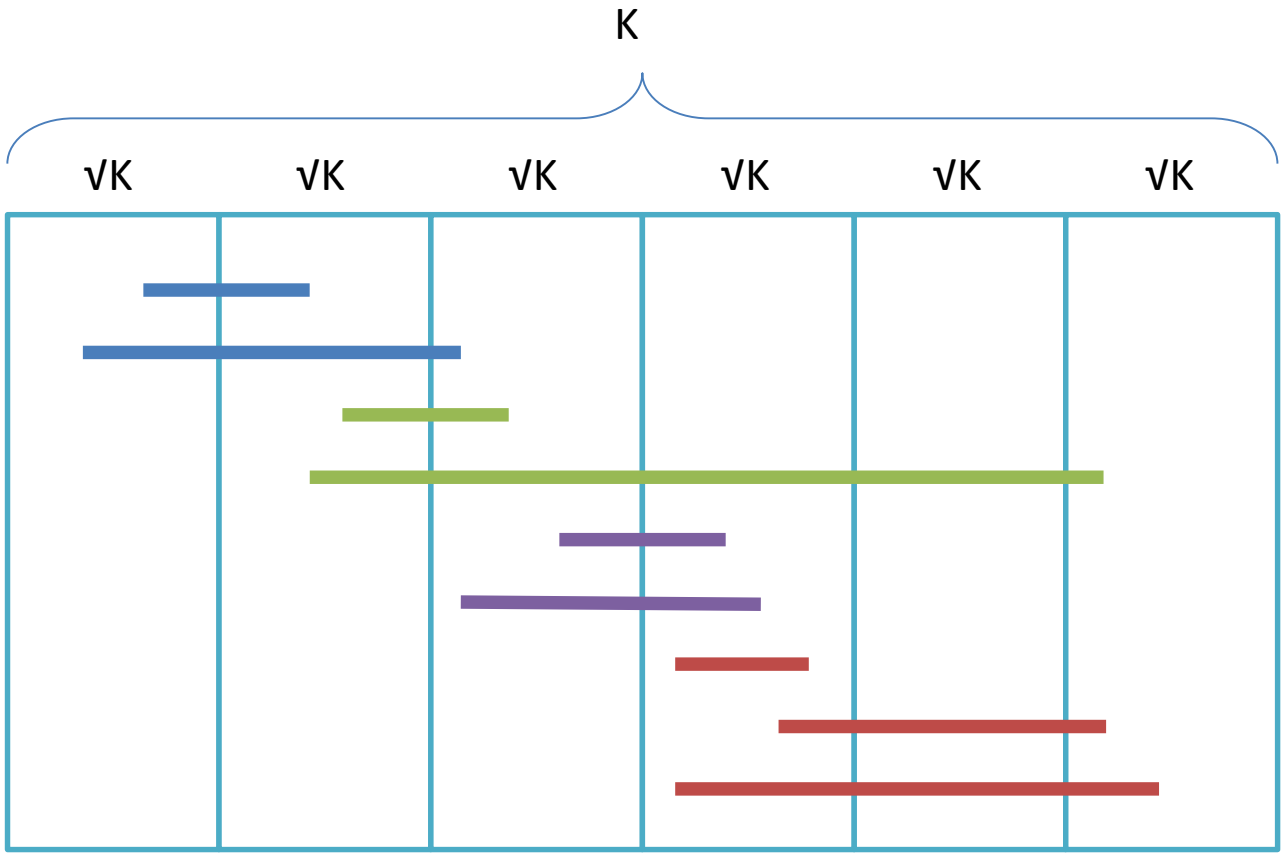


問10　科学物質アルファ

AIZU
THE UNIVERSITY OF AIZU

解法

- 左端：１つのクエリあたり最大√K回移動するので、O(Q√K)
- 右端：それぞれの区画ごとに最大K回移動するので、O(K√K)



問10　科学物質アルファ

AIZU
THE UNIVERSITY OF AIZU

解答例

```
class Query{
public:
  int id, q, l, r, x;
  Query(){}
  Query(int id, int q, int l, int r, int x): id(id), q(q), l(l), r(r), x(x){}
};
int width;
bool lrCmp(const Query &q1, const Query &q2) {
  if ( q1.l / width != q2.l / width ) return q1.l < q2.l;
  return q1.r < q2.r;
}

void forward(int i){
  swap(P[R[a[i]]], P[R[b[i]]]);
  swap(R[a[i]], R[b[i]]);
}
void backward(int i){
  swap(R[P[a[i]]], R[P[b[i]]]);
  swap(P[a[i]], P[b[i]]);
}
```

問10　科学物質アルファ

AIZU
THE UNIVERSITY OF AIZU

解答例（つづき）

```
void solve(){
  for ( int i = 0; i < N; i++ ) P[i] = R[i] = i;

  sort(Q.begin(), Q.end(), lrCmp);

  int nl = 0, nr = 0;
  for ( int i = 0; i < NQ; i++ ){
    while(nl > Q[i].l) forward(--nl);  // add forward
    while(nr < Q[i].r) backward(nr++); // add backward
    while(nl < Q[i].l) forward(nl++);  // delete forward
    while(nr > Q[i].r) backward(--nr); // delete backward
    if ( Q[i].q == 1 ){
      ans[Q[i].id] = P[Q[i].x]+1;
    } else {
      ans[Q[i].id] = R[Q[i].x]+1;
    }
  }
}
```

問11　三角形の個数の和

AIZU
THE UNIVERSITY OF AIZU

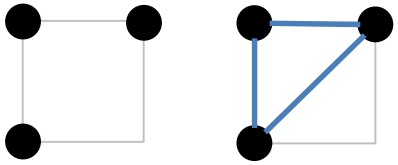
問題概要

- 座標平面上の原点Oを左下、座標(W,H)にある点を右上とする長方形の領域に含まれる、x座標とy座標がともに整数である点を３個以上含む点の集まりをD1、D2、…、DNとする。
- D1、D2、…、DNのそれぞれに含まれる点を頂点とする三角形の個数をt1、t2、…、tNとするとき、三角形の個数の和t1+t2+…+tNを求めよ
- 1≦W, H≦1,000

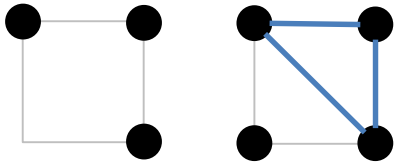
問11　三角形の個数の和

AIZU
THE UNIVERSITY OF AIZU

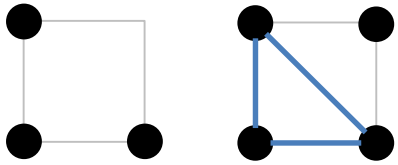
考察

D1 

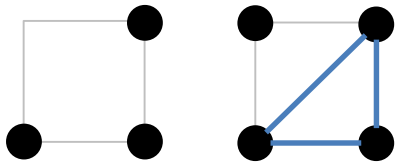
t1 = 1

D2 

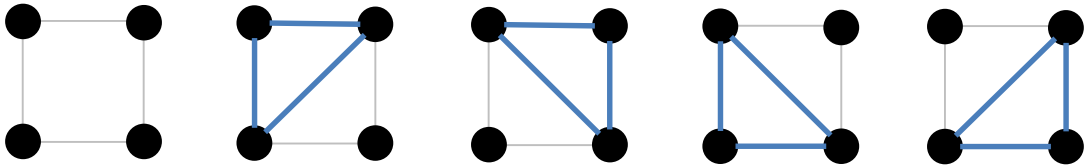
t2 = 1

D3 

t3 = 1

D4 

t4 = 1

D5 

t5 = 4　　Σti = 8

問11　三角形の個数の和

AIZU
THE UNIVERSITY OF AIZU

考察

- ※W ＝ W+1, H ＝ H+1
- W×Hの点集合Aから、任意の要素数３の部分集合Pを選び、答えに数えられる回数を求める.
- Pが三角形をなすとき、Pを含むAの部分集合の数だけ数える.

  ＝　A＼Pの部分集合の数

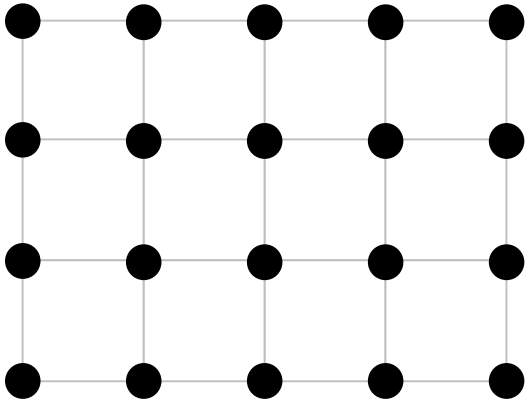
  ＝　$2^{A \setminus P} = 2^{(HW-3)}$
- 三角形をなすようなAの要素数3の部分集合の選び方の場合の数を求め、$2^{(HW-3)}$ をかけることで答えを求める。
- 三角形をなさないようなAの要素数3の部分集合の選び方の場合の数を求める。
- これを要素数3の部分集合を選ぶ場合の数を全体から引くことで、三角形をなすような要素数3の部分集合の選び方の場合の数を求める。

問11　三角形の個数の和

AIZU
THE UNIVERSITY OF AIZU

考察

- 異なる三点が三角形をなさないことと、三点を含む直線がただ一つ存在することは同値。
- 傾きが0である場合の数：
  - H(W(W-1)(W-2)/3!)
- 傾きが垂直である場合の数：
  - W(H(H-1)(H-2)/3!)

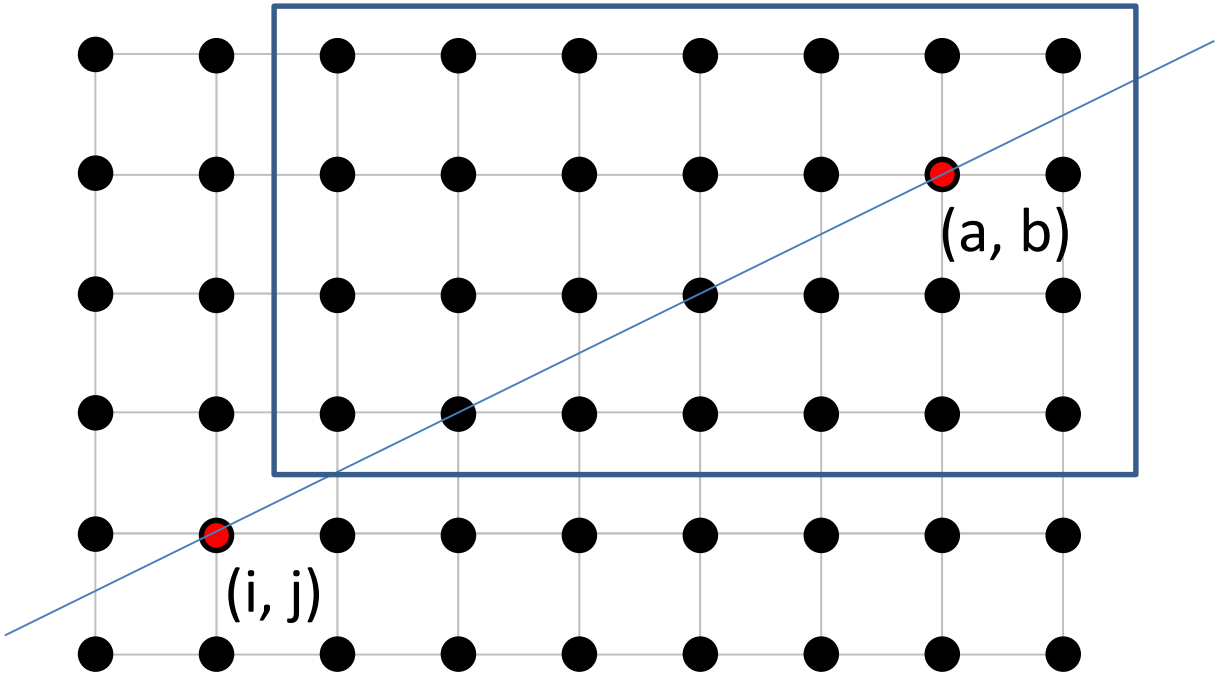


問11　三角形の個数の和

AIZU
THE UNIVERSITY OF AIZU

考察

- 三点を含む直線の傾きが正であるとき：
  - Aの要素(i,j)を左下の点として固定したとき、残りの二点を選ぶ場合の数を$C_{i,j}$
  - 右上の点を (a,b) in A,(i<a, j<b) に固定すると、残り一点の選び方は gcd(a-i,b-j) – 1



問11　三角形の個数の和

AIZU
THE UNIVERSITY OF AIZU

考察

- $C_{i,j}$ = Σ_{(a,b) in A,i<a,j<b} (gcd(a-i,b-j) - 1).
- gcd(x,y)-1 , (1≦ x ≦ H,1 ≦ y ≦ W) の値をあらかじめ求め、累積和をとったテーブルにすることでO(1)で求まる。
- 左下の点の候補としてAの全ての要素を試すことで、求めたい値が得られる。

- 対称性から、傾きが負である場合の数は正である場合の数と等しい。

- O(HW log(min(H,W)))

問11　三角形の個数の和

AIZU
THE UNIVERSITY OF AIZU

解答例

```
void initGCD(){
  for (ll i = 1; i <= W; i++ )
    for (ll j = 1; j <= H; j++ ) G[i][j] = gcd(i, j) - 1;
  for ( ll j = 1; j <= H; j++ )
    for ( ll i = 1; i <= W; i++ ) AG[i][j] += AG[i-1][j] + G[i][j];
  for ( ll i = 1; i <= W; i++ )
    for ( ll j = 1; j <= H; j++ ) AG[i][j] += AG[i][j-1];
}
```

問11　三角形の個数の和

AIZU
THE UNIVERSITY OF AIZU

解答例（つづき）

```
MInt BASE = 1;
MInt ALL3 = N*(N-1)*(N-2)/6;
for ( int i = 0; i < N-3; i++ ) BASE = BASE*2;
MInt VER = WW*HH*(HH-1)*(HH-2)/6;
MInt HOR = HH*WW*(WW-1)*(WW-2)/6;
MInt DIAG = 0;
for ( int i = 0; i < W; i++ ){
  for ( int j = 0; j < H; j++ ){
    DIAG = DIAG + AG[W-i][H-j];
  }
}
MInt S = DIAG*2 + VER + HOR;
MInt ans = (ALL3 - S)*BASE;
```

問１２　惑星ヤナイヅの資源

AIZU
THE UNIVERSITY OF AIZU

問題概要

- 幼虫がc_i匹いる群生の並び $\{c_1, c_2, \ldots c_N\}$ について以下の操作を行う
  - 指定した範囲のすべての群生に、d匹の幼虫を追加する.
  - 指定した範囲のすべての群生で、d匹の幼虫を成虫にう化させ、う化した幼虫の数の合計を求める.
- $1 \leqq N \leqq 100,000$
- 操作の数$M \leqq 100,000$
- $1 \leqq d \leqq 1,000$
- 初期状態の幼虫の数$c_i \leqq 100,000$

問１２　惑星ヤナイヅの資源

AIZU
THE UNIVERSITY OF AIZU

考察

- 数列{c_1, c_2, … c_N} について以下のクエリに答える
  - 区間[l, r]に対して add(d)
  - 区間[l, r]に対して add(-d)した場合、合計でいくら減ったか？　ただし、結果が負の値になってはいけない

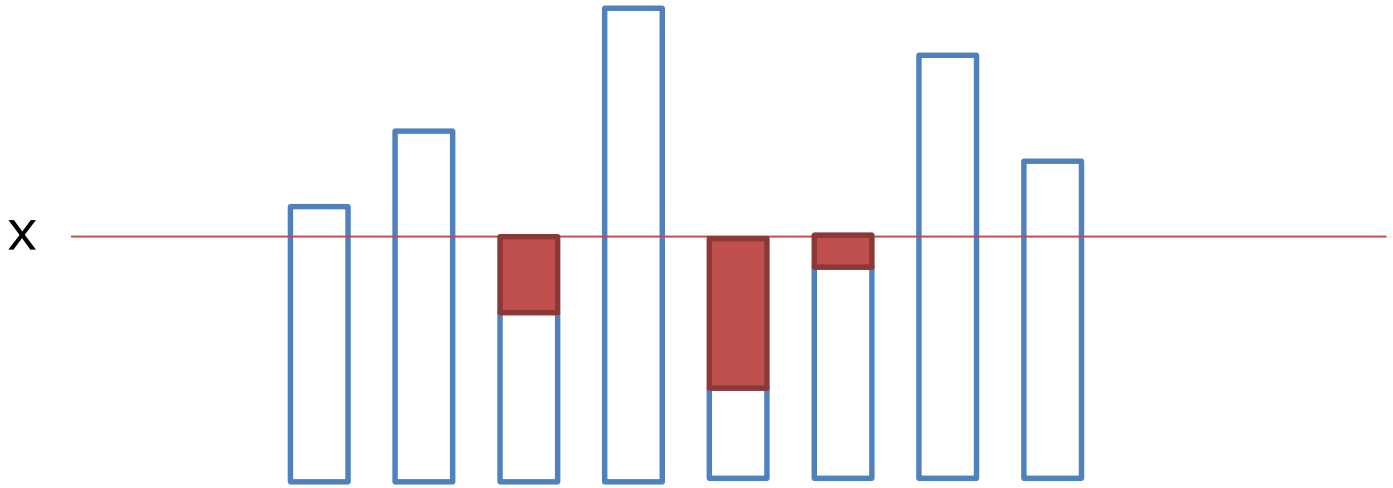
- 数列{c_1, c_2, … c_N} について以下のクエリに答える
  - 区間[l, r]に対して add(d)
  - 区間[l, r]に対して add(-d)を行い、chmax(c_i, 0)を行う
    - このときの差分がう化した幼虫の数

問１２　惑星ヤナイヅの資源

AIZU
THE UNIVERSITY OF AIZU

考察

- chmax(c_i, x)を考える
  - 指定された区間[l, r]について、max(c_i, x)をとる
  - 通常の遅延評価付きセグメントツリーでは難しい



問１２　惑星ヤナイヅの資源

AIZU
THE UNIVERSITY OF AIZU

考察

- 以下の状態を持たせた遅延評価付きセグメントツリーを実装する
  - minVal[k]：　区間k内の最小値
  - minCnt[k]：　区間k内の最小値の個数
  - minVal2nd[k]：　区間k内の（厳密に）2番目の最小値
  - sumVal[k]：　区間kの合計値
  - addLazy[k]：　区間kに一様に足された値（遅延評価用）

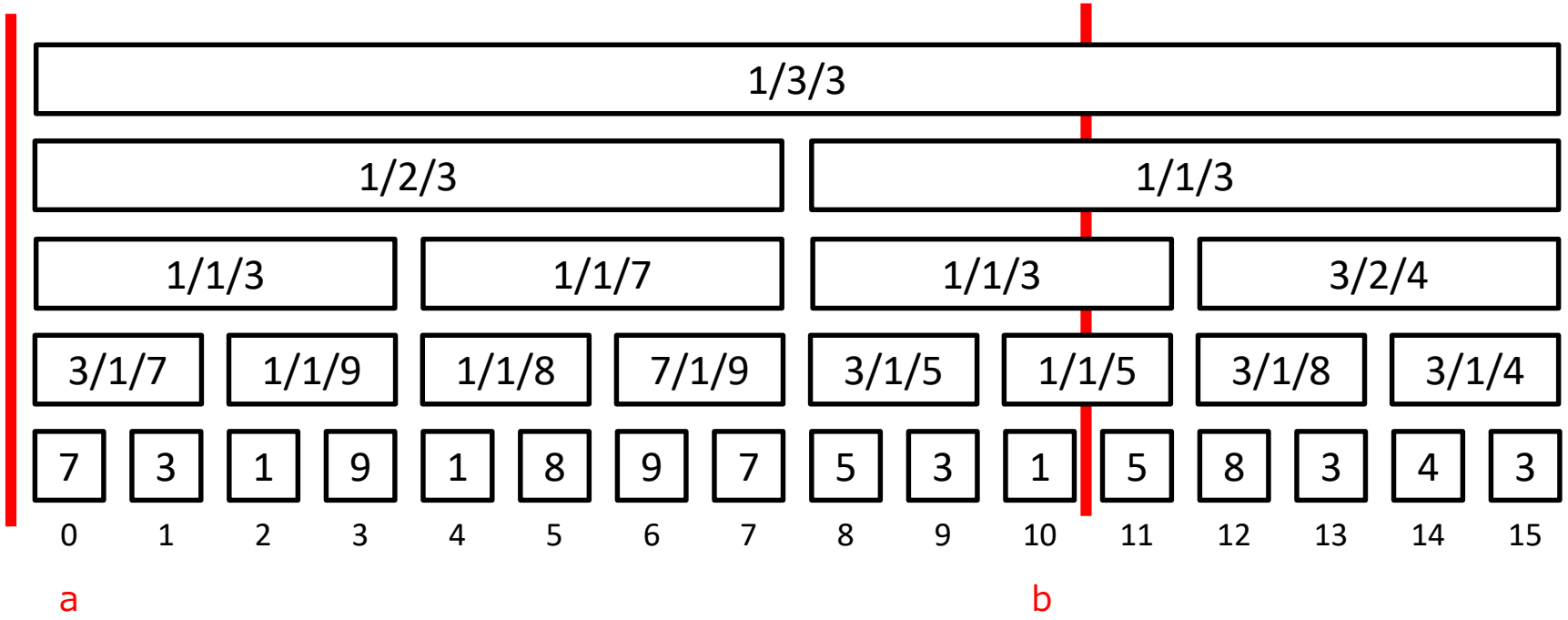
問１２　惑星ヤナイヅの資源

AIZU
THE UNIVERSITY OF AIZU

```
// chmax
void updateMax(ll x, int a, int b, int k, int l, int r) {
    if(b <= l || r <= a || x <= minVal[k]) {
      return;
    }
    if(a <= l && r <= b && x < minVal2nd[k]) {
      updateMinVal(k, x);
      return;
    }
    push(k);
    updateMax(x, a, b, 2*k+1, l, (l+r)/2);
    updateMax(x, a, b, 2*k+2, (l+r)/2, r);
    update(k);
 }

void updateMinVal(int k, ll x) {
    sumVal[k] += (x - minVal[k]) * minCnt[k];
    minVal[k] = x;
}
```

問１２　惑星ヤナイヅの資源

考察

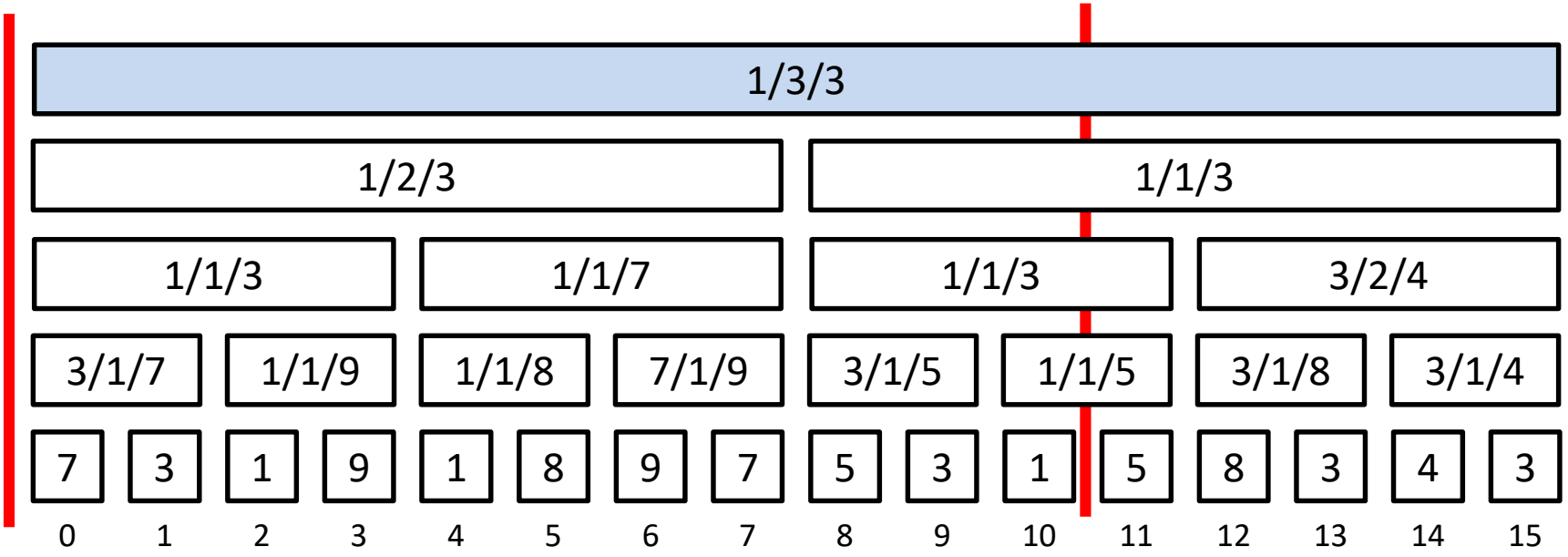


minVal/minCnt/minVal2nd

問１２　惑星ヤナイヅの資源

AIZU
THE UNIVERSITY OF AIZU

考察

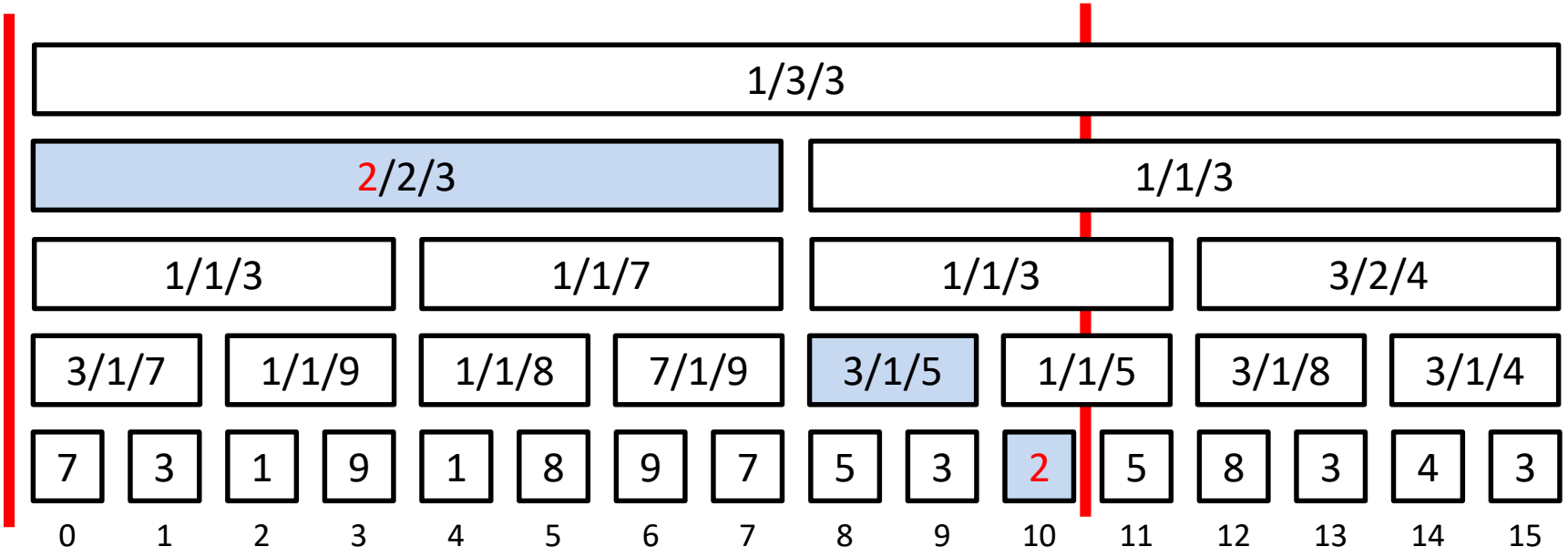


[0, 10]に対してchmax(c_i, 0)

問１２　惑星ヤナイヅの資源

AIZU
THE UNIVERSITY OF AIZU

考察



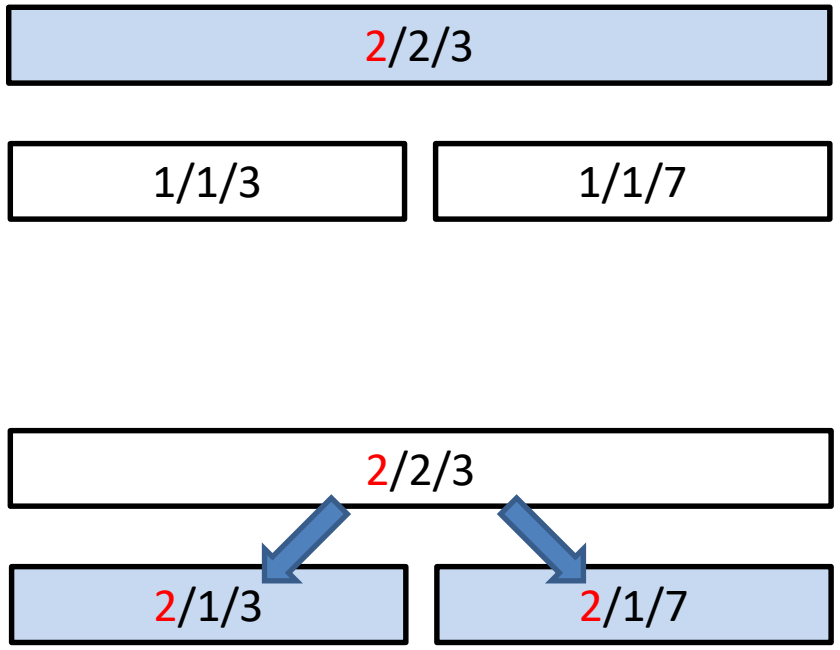
[0, 10]に対してchmax(c_i, 2)

minVal[k] < x < minVal2nd[k]

問１２　惑星ヤナイヅの資源

AIZU
THE UNIVERSITY OF AIZU

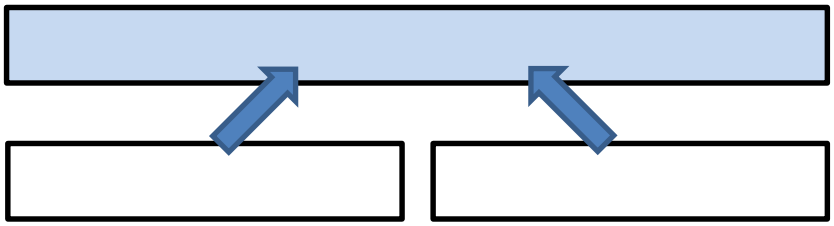
```
void push(int k) {
    ...

    if(addLazy[k] != 0) {
      addAll(2*k+1, addLazy[k]);
      addAll(2*k+2, addLazy[k]);
      addLazy[k] = 0;
    }
    if(minVal[2*k+1] < minVal[k]) {
      updateMinVal(2*k+1, minVal[k]);
    }
    if(minVal[2*k+2] < minVal[k]) {
      updateMinVal(2*k+2, minVal[k]);
    }
  }
```



問１２　惑星ヤナイヅの資源

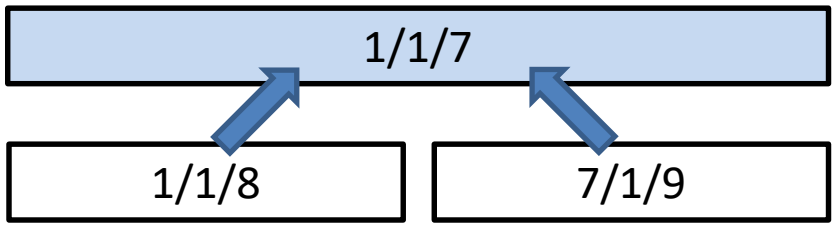
AIZU
THE UNIVERSITY OF AIZU

```
void update(int k) {
    sumVal[k] = sumVal[2*k+1] + sumVal[2*k+2];

    if(minVal[2*k+1] < minVal[2*k+2]) {
      minVal[k] = minVal[2*k+1];
      minCnt[k] = minCnt[2*k+1];
      minVal2nd[k] = min(minVal2nd[2*k+1], minVal[2*k+2]);
    } else if(minVal[2*k+1] > minVal[2*k+2]) {
      minVal[k] = minVal[2*k+2];
      minCnt[k] = minCnt[2*k+2];
      minVal2nd[k] = min(minVal[2*k+1], minVal2nd[2*k+2]);
    } else {
      minVal[k] = minVal[2*k+1];
      minCnt[k] = minCnt[2*k+1] + minCnt[2*k+2];
      minVal2nd[k] = min(minVal2nd[2*k+1], minVal2nd[2*k+2]);
    }
}
```



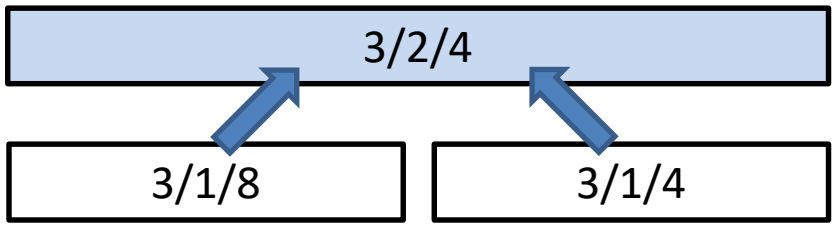
問１２　惑星ヤナイヅの資源

```
...

    if(minVal[2*k+1] < minVal[2*k+2]) {
      minVal[k] = minVal[2*k+1];
      minCnt[k] = minCnt[2*k+1];
      minVal2nd[k] = min(minVal2nd[2*k+1], minVal[2*k+2]);
    } ...
```



問１２　惑星ヤナイヅの資源

```
...
} else { // minVal[2*k+1] == minVal[2*k+2]
      minVal[k] = minVal[2*k+1];
      minCnt[k] = minCnt[2*k+1] + minCnt[2*k+2];
      minVal2nd[k] = min(minVal2nd[2*k+1], minVal2nd[2*k+2]);
}...
```



問１２　惑星ヤナイヅの資源

AIZU
THE UNIVERSITY OF AIZU

計算量

- 全体でO(N log N + M $\log^2$ N)
- 参考：Codeforces, Segment Tree Beats