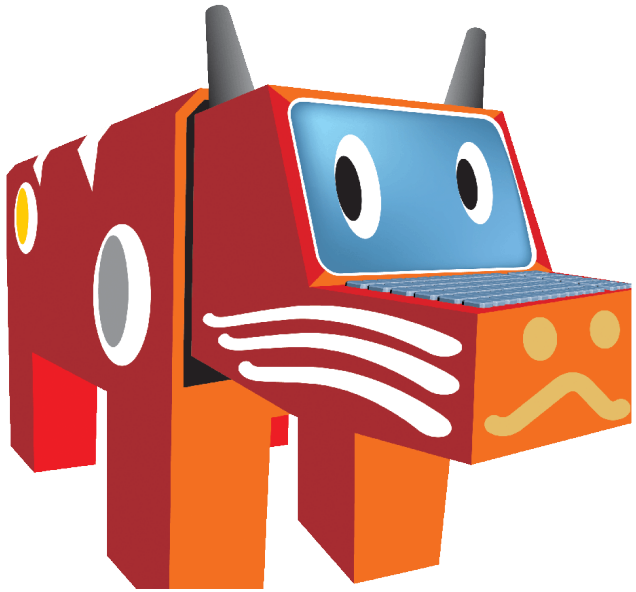
パソコン甲子園2012 本選

プログラミング部門　解説



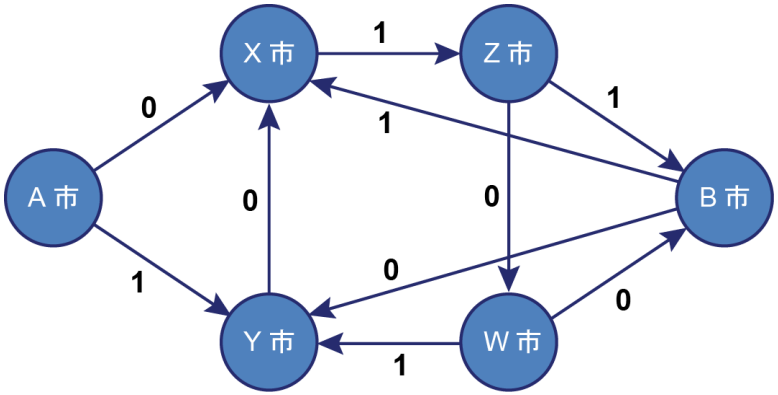
会津大学
THE UNIVERSITY OF AIZU

To Advance Knowledge for Humanity
THE UNIVERSITY OF AIZU
1993

問1　アカ･ベコと40人の盗賊

問題概要

- 都市とそれらを結ぶ道を表す有向グラフについて、与えられる指示が、スタート地点のA市からゴール地点のB市へたどり着く経路かを判定するプログラムを作成せよ.
- 指示は行先を示す0と1からなる文字列で与えられる.



講評

- 簡単なシミュレーションを行うプログラムが作成できるかを問う問題である.

問1　アカ･ベコと40人の盗賊

解法１

- オートマトン.
- 現在位置を保持しながらシミュレーションを行う.
- 分岐処理で次に移動する都市を決定する.

解法２

- 各都市と行先を基に、有向グラフを配列（行列）で表現する.
- グラフ上の探索のシミュレーションを行う.

問1　アカ･ベコと40人の盗賊

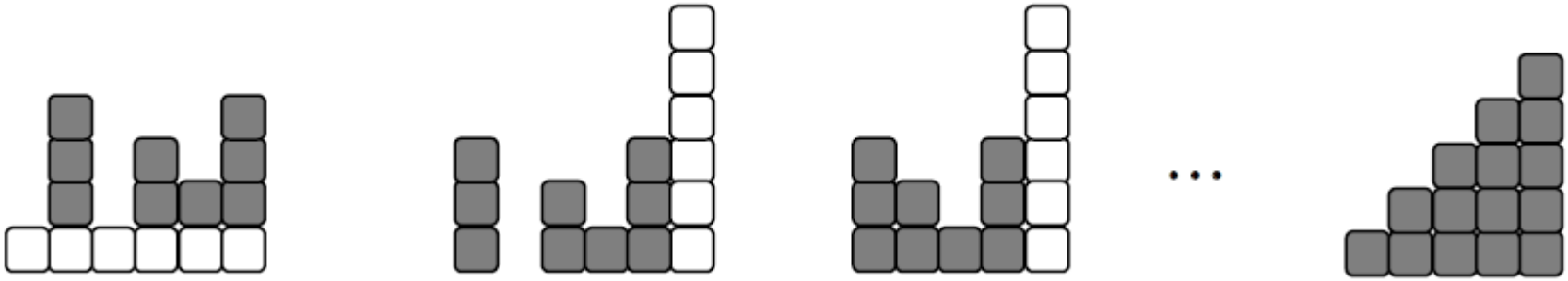
Cによる解答例（解法1）

```
int simulate(char S[]){
  int i, u;
  char cur = 'A';
  for ( i = 0; i < strlen(S); i++ ){
    u = S[i]-'0';
    if ( cur == 'A' ) cur = u ? 'Y' : 'X';
    else if ( cur == 'W' ) cur = u ? 'Y' : 'B';
    else if ( cur == 'X' ) cur = u ? 'Z' : ' ';
    else if ( cur == 'Y' ) cur = u ? ' ' : 'X';
    else if ( cur == 'Z' ) cur = u ? 'B' : 'W';
    else if ( cur == 'B' ) cur = u ? 'X' : 'Y';
    if ( cur == ' ' ) return 0;
  }
  return cur == 'B';
}
```

問2　ブロックの三角形

問題概要

- 与えられたブロックの列に対して「最下部のブロックを右端に積み上げ、隙間を左に詰める」処理を何回繰り返せば、ブロックが階段状に並ぶかを求めるプログラムを作成せよ.



講評

- ブロックの総数の制限から、効率は考慮しなくてよい.
- 簡単なシミュレーションを行えるかを問う問題である.

問2 ブロックの三角形

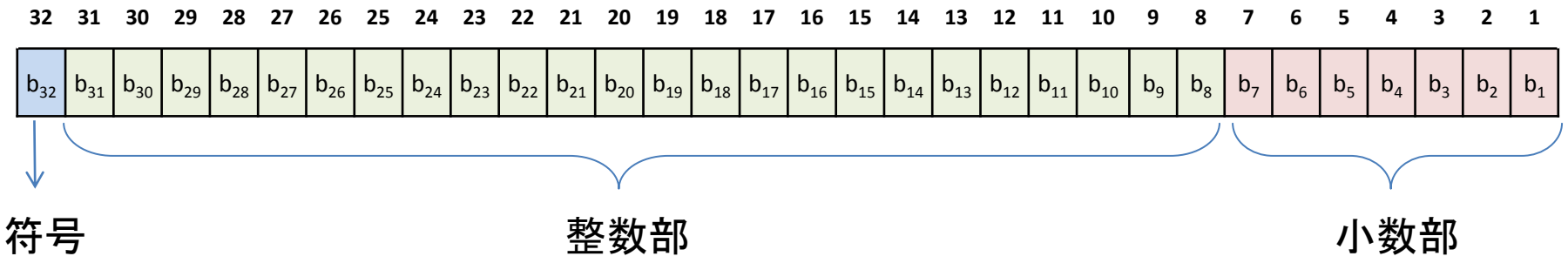
C++による解答例: シミュレーション関数

```
int simulate(vector<int> v, int sum){
  int cnt = 0;
  while(1){
    if ( check(v) ) return cnt;
    int r = v.size();
    for ( int i = 0; i < v.size(); i++ ) v[i]--;
    vector<int> nv;
    for ( int i = 0; i < v.size(); i++ ) {
      if ( v[i] ) nv.push_back(v[i]);
    }
    v = nv;
    v.push_back(r);
    cnt++;
    if ( cnt > LIM ) return -1;
  }
}
```

問3　金剛の型

問題概要

- 実数型（固定小数点型）を10進数に変換するプログラムを作成せよ.
- 2進->10進変換を行う.
- 入力は16進数で与えられる.



問3　金剛の型

解法：整数部の取得

- 入力はlong long (C/C++), long int (Java)で読み込む.
- 上一桁を消して（符号の分）、7ビットシフトダウンすると整数部が得られる.

解法：小数部の取得

- 小数部も整数値として計算することができる.
- 7ビット目から1ビット目の順番で、ビットが立っているところについて、5000000, 2500000, 1250000, 625000, 312500, 156250, 78125（0.0078125に相当）を加算していけば、小数部の値が整数値で得られる.

問3　金剛の型

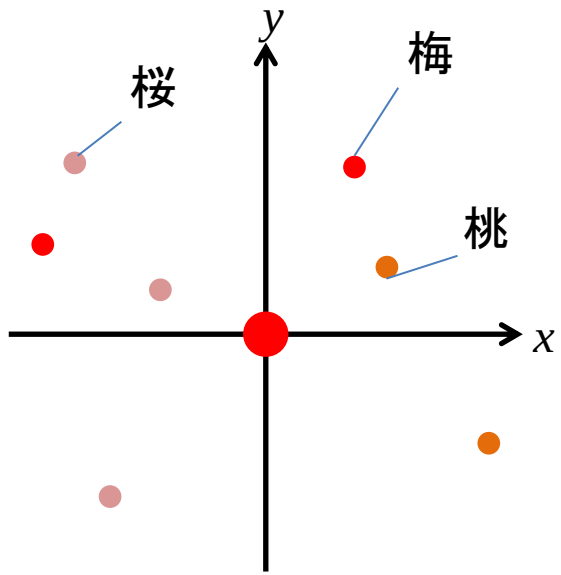
C++による解答例：整数部、小数部の取得

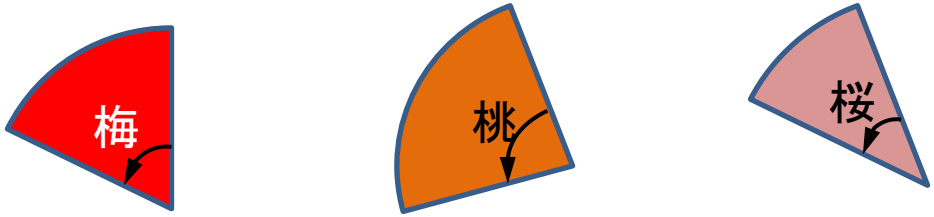
```
uint ExtractIntegerPart(uint nBits) {
     return (nBits & 0x7fffff80) >> 7;
}

ullong ExtractFractionPart(uint nBits) {
    ullong nFracValue = 0;
    ullong nMask = power(10, 7);

     for (int i=6; i>=0; --i) {
         if ( (nBits >> i) & 1 ) nFracValue += (nMask >> (7 - i));
    }

    return nFracValue;
}
```

問題4　風よ、私の梅の香りを届けておくれ

問題概要

- 私の梅が原点にあり、他の梅、桃、桜が各々別々の位置にある.
- 香りが拡がる角度が花の種類ごとに与えられる.





- 風の吹く向きと強さが、数日分与えられる
  - 吹く向きは扇形の中心軸
  - 吹く強さは扇形の半径

問題4　風よ、私の梅の香りを届けておくれ

問題概要

- 複数の物件の位置が与えられるので、「私の梅の香りだけが届く日が最も多い物件」を出力せよ.

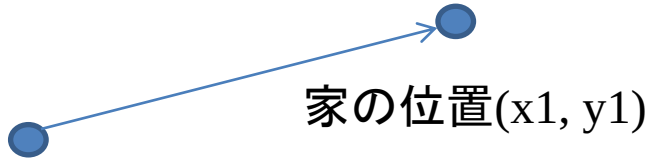
講評

- 問題の本質は、ある点が扇形の内部に位置するかどうかを判定することである.
- 花の数、物件の数が少ないため、効率は考えなくてよい.

問題4　風よ、私の梅の香りを届けておくれ

解法：点が扇型の内部にあるかの判定

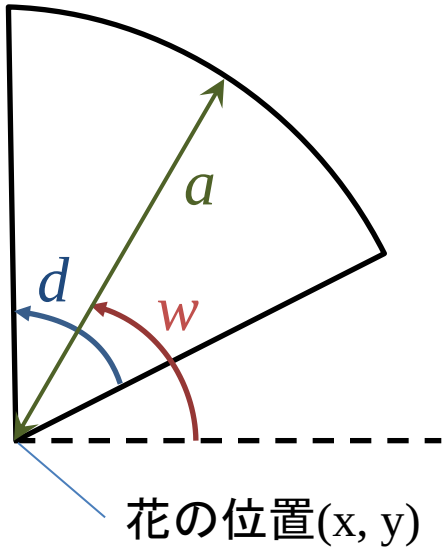
1. 距離内である

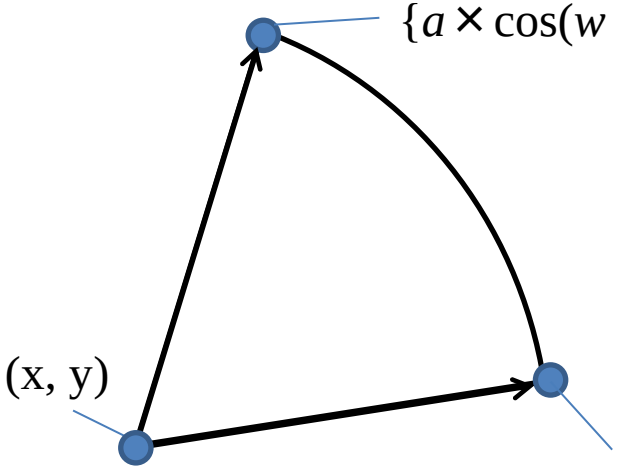


$$距離 = \sqrt{(x1 - x0)^2 + (y1 - y0)^2}$$

距離 > 風の強さなら
この花の香りは、この家に届かない
(距離の2乗と風の強さの2乗の比較でsqrtは避けることができる)

2. 扇の両側面の内である

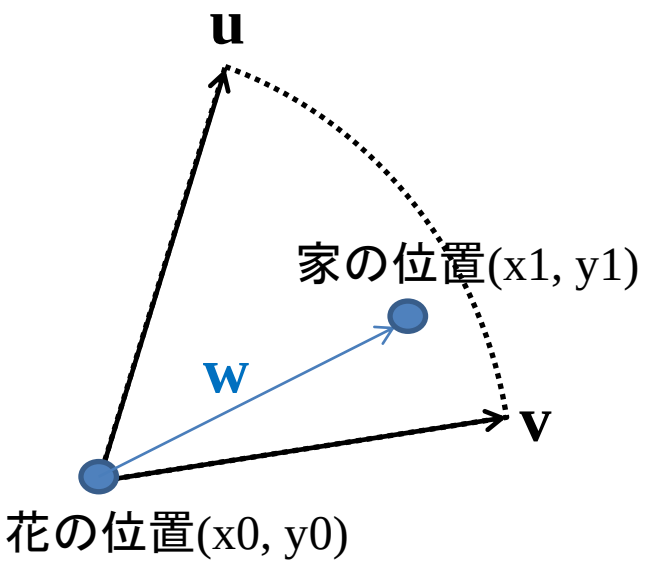




問題4　風よ、私の梅の香りを届けておくれ

解法：点が扇型の内部にあるかの判定

2. 扇の両側面の内である (説明続き)



u→wがCW（時計回り）
　かつ
v→wがCCW（反時計回り）

問題4　風よ、私の梅の香りを届けておくれ

C++による解答例：扇形の内部に点pがあるか

```
bool isInSector(Point c, double a, double d, double r, Point p ){
  double dist = abs(c - p);
  if ( dist > a ) return false;

  Vector v1 = Vector(a*cos((r + d/2)*PI/180), a*sin((r + d/2)*PI/180));
  Vector v2 = Vector(a*cos((r - d/2)*PI/180), a*sin((r - d/2)*PI/180));

  if ( ccw(c, c+v1, p) == CLOCKWISE &&
       ccw(c, c+v2, p) == COUNTER_CLOCKWISE ) {
    return true;
  }
  return false;
}
```

問題4　風よ、私の梅の香りを届けておくれ

C++による解答例：CCW

```
double norm( Vector a ){ return a.x*a.x + a.y*a.y; }
double dot( Vector a, Vector b ){ return a.x*b.x + a.y*b.y; }
double cross( Vector a, Vector b ){ return a.x*b.y - a.y*b.x; }

int ccw( Point p0, Point p1, Point p2 ){
    Vector a = p1 - p0;
    Vector b = p2 - p0;
    if ( cross(a, b) > EPS ) return COUNTER_CLOCKWISE;
    if ( cross(a, b) < -EPS ) return CLOCKWISE;
    if ( dot(a, b) < -EPS ) return ONLINE_BACK;
    if ( norm(a) < norm(b) ) return ONLINE_FRONT;
    return ON_SEGMENT;
}
```

問題5　モジュロ・クエリ

問題概要

- 要素数Nの数列AとQ個のクエリqについて、A[i] mod q[j]の最大値を求めるプログラムを作成せよ.
- N ≦ 300000、Q ≦ 100000、同じクエリは与えられない.
- 数列の要素c、各クエリqの範囲は1≦c, q≦300000.

q[j] = 4

A[i]	4	9	2	13
A[i]%q[j]	0	1	2	1

講評

- N, Qの上限が大きいため、O(NQ)の素朴なアルゴリズムでは時間切れとなる.

問題5　モジュロ・クエリ

解法

- 与えられた数列Aは、その要素の出現の有無を表すサイズがmax(A)+1の配列Tで表現する.
- T[i]をもちいて、iより前でAに出現する要素の中で最大のものを記録した配列Lを生成する.

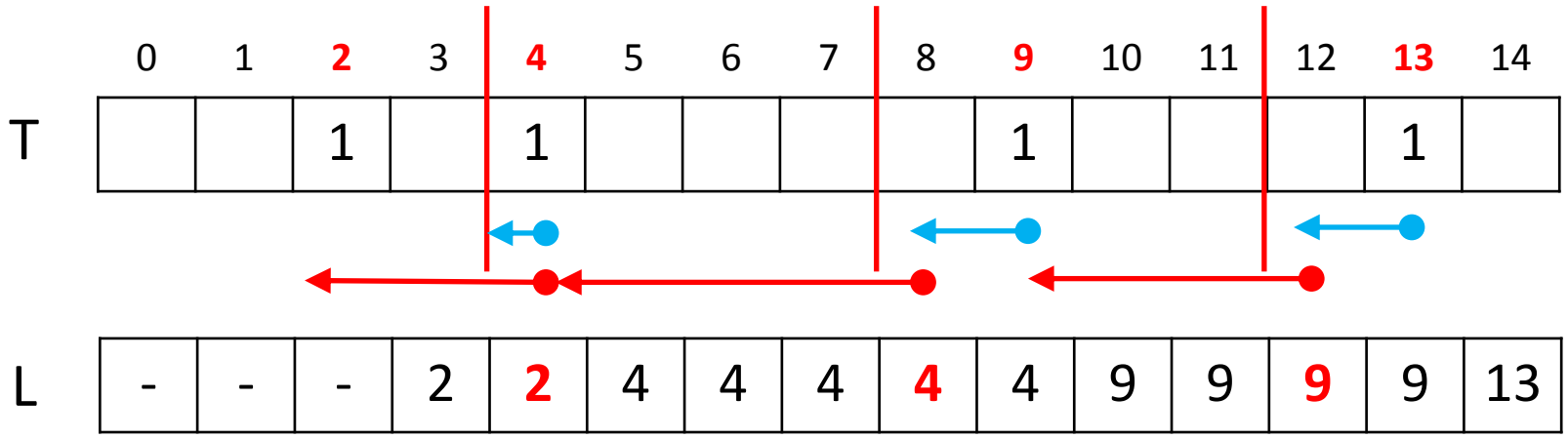
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
T			1		1					1				1	

L	-	-	-	2	2	4	4	4	4	4	9	9	9	9	13

問題5　モジュロ・クエリ

解法

- 配列Tを後ろからたどっていき、(T[i]が1であるi)を探し、i%qの最大値を更新していく.
- 次の候補に移動するときは、現在位置iからi%qを引いた位置に移動(図の青い矢印)し、その位置のLの値(L[i-i%q])へ移動(図の赤い矢印)する.



問題5　モジュロ・クエリ

計算量

- 同じクエリは与えられないことから、O(∑ (max(A)/q[j]))のアルゴリズムとなる.
- 最大ケースでも 4000000以下程度.

問題5　モジュロ・クエリ

C++による解答例：クエリの処理

```
for ( int i = 0; i < Q; i++ ){
  scanf("%d", &q);
  int maxv = 0;
  /* m = max(A) */
  for ( int cur = m; cur; ){
    int p = cur%q;
    maxv = max(maxv, p);
    if ( cur - p < 0 ) break;
    cur = L[cur - p];
  }
  printf("%d¥n", maxv);
}
```

問題6　イヅア国語辞典

問題概要

- イヅア語はN種類の文字からなり、重複しないN文字で1つの単語が表される.
- 与えられる単語が辞書順で何番目かを求めるプログラムを作成せよ. 答えは定数Mで割った余りを出力せよ.
- ただし、与えられる単語は、辞書順で最初の単語について、2つの文字の位置を入れ替える操作を高々R（50以下）回行うことで得られる.

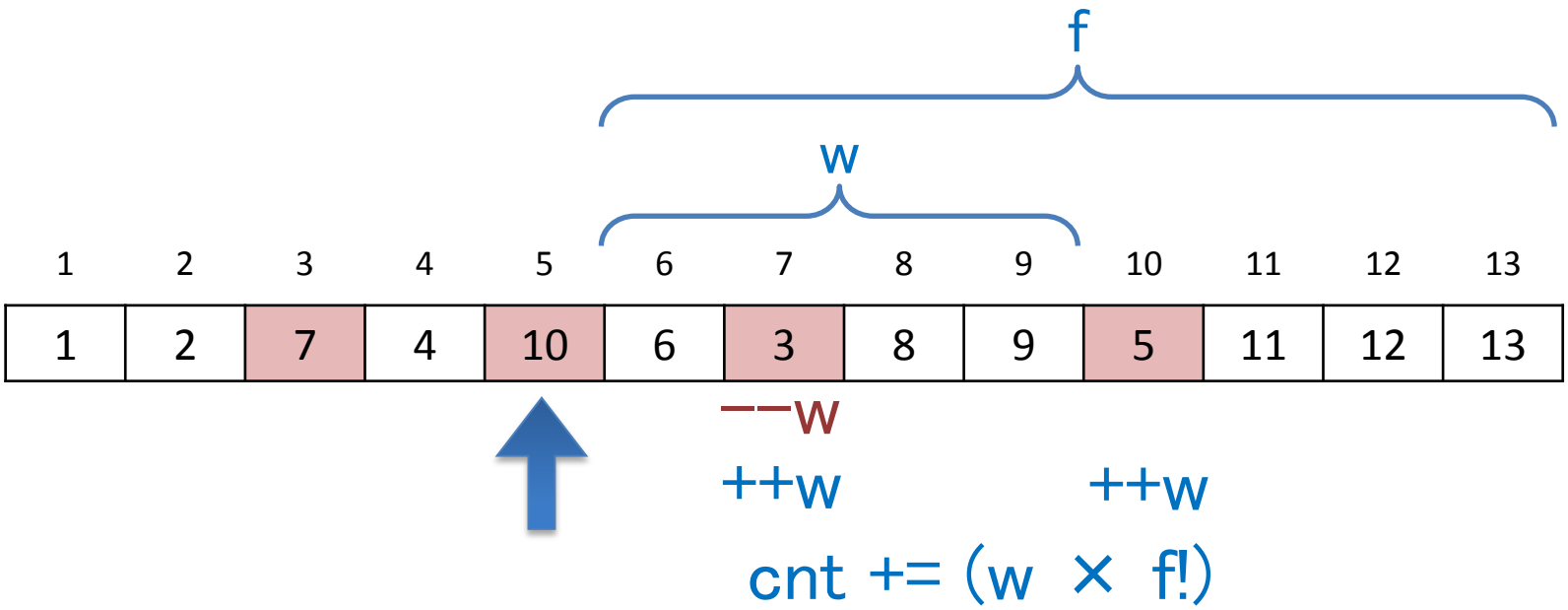
講評

- 高等的なデータ構造を用いるか、あるいは問題の制約をうまく利用し、簡単な計算に落とせるかが問われている.

問題6　イヅア国語辞典

解法（制約を利用し、簡単な計算に落とす）

- 単語の先頭から1文字づつ注目して、それが辞書順で何番目かを求め、総和を求める.
- 最初の単語から高々R回スワップしてできた単語であることを利用しO(NR)のアルゴリズムを実装することができる.



問題6　イヅア国語辞典

C++による解答例：　総和の計算

```
for ( int i = 1; i <= N; i++ ){
    if ( v[i] != i ) R.push_back(make_pair(v[i], i));
}

llong cnt = 0;
llong a = 1, f = 1;
for ( int p = N; p > 0; p-- ){
    llong L = max(v[p]-1, p-1);
    llong W = max(L-p, 0LL);
    for ( int i = 0; i < R.size(); i++ ){
        if ( R[i].second <= p ) continue;
        if ( R[i].second <= L ) W--;
        if ( R[i].first < v[p] ) W++;
    }
    cnt = (cnt + (W*a)%M)%M;
    a = (a*f)%M;
    f = (f+1)%M;
}
cout << cnt << endl;
```

問題7　寂しがり屋のついた嘘

問題概要

- 私と彼女がカードゲームをする.
- それぞれ数字の書かれたN枚のカードを順番に対戦させ、N/2より多く勝利した者を勝ちとする.
- N回勝負ではなく、k回勝負では勝てるかもしれない。
- 2人のカードを入力し、彼女がどんな選択をしようとわたしが勝てるような最小のkを求めるプログラムを作成せよ.

問題7　寂しがり屋のついた嘘

解法：考え方

- 両者とも強い方からk匹選ぶのが最適戦略.
- わたしの最強のk匹で彼女の最強のk匹に勝てるか判定できればよい.

あるkについて、

彼女がどの順番で出してきてもわたしが勝利できる.

対戦するモンスターの組み合わせを任意に決めるとき、わたしの最低勝利数が k/2 より大きい.

問題7　寂しがり屋のついた嘘

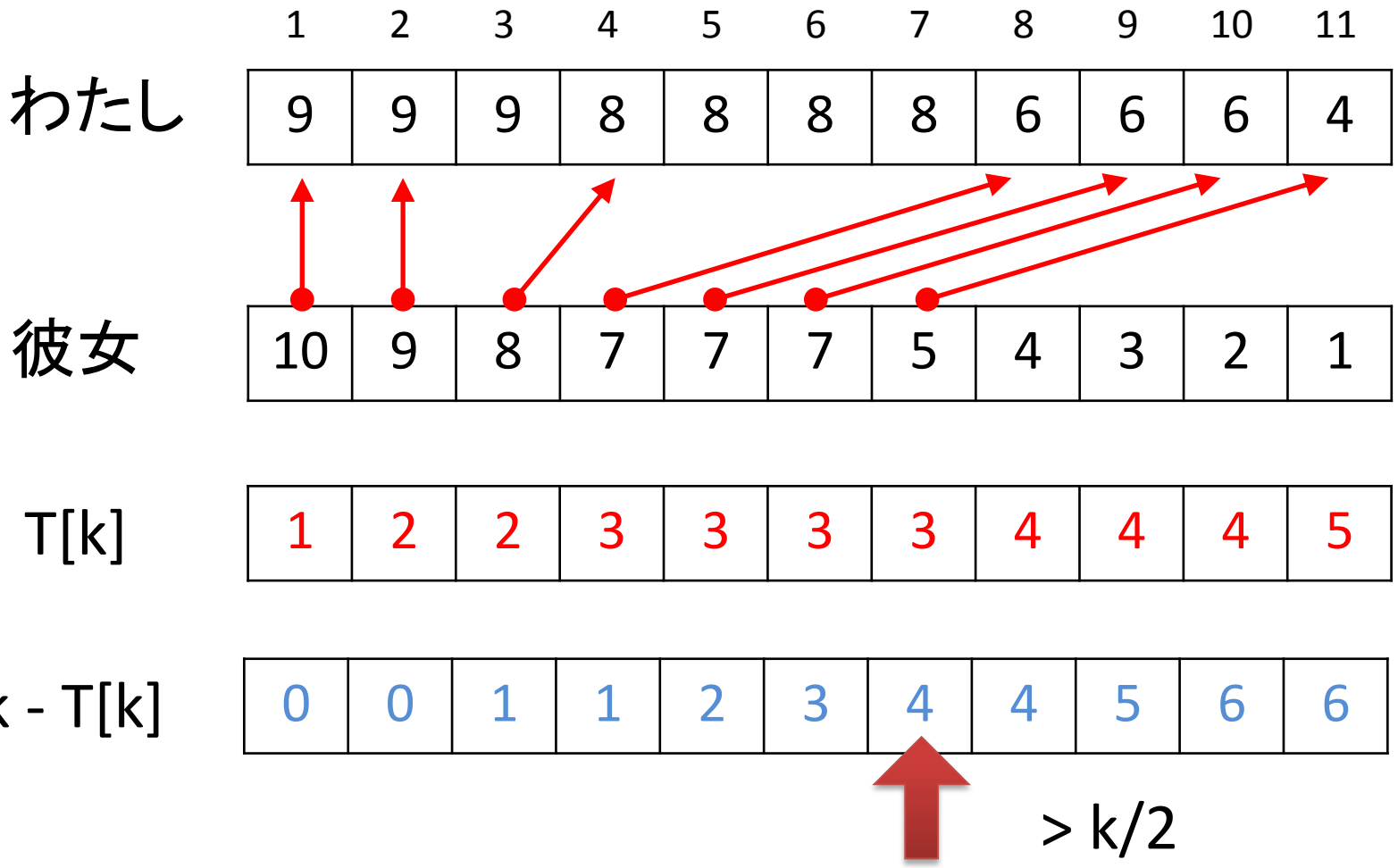
解法：考え方

わたしの最低勝利数　　＝
わたしの最大[敗北＋引き分け]数　＝
彼女の最大[勝利＋引き分け]数

彼女が任意に対戦組み合わせを決められるとして、彼女の最大[勝利＋引き分け]数を求めればよい.

問題7　寂しがり屋のついた嘘

解法：強い順にソートして貪欲法



問題7　寂しがり屋のついた嘘

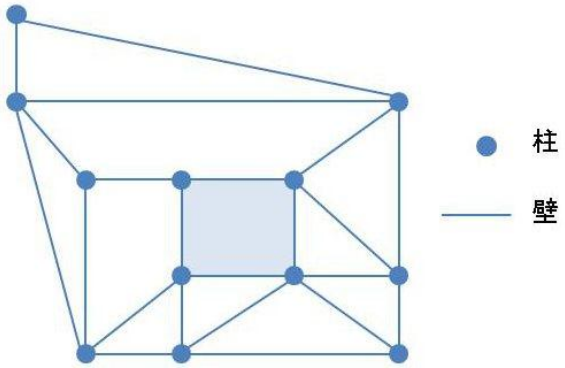
C++による解答例

```
sort(P1+1, P1+n+1, greater<int>());
sort(P2+1, P2+n+1, greater<int>());

for ( int i = 0; i <= n; i++ ) T[i] = 0;

int s = 1, t = 1;
while( s <= n && t <= n ){
  while( P1[t] > P2[s] && t <= n ) t++;
  if ( t <= n && P2[s] >= P1[t] ){
     T[t]++;
     t++; s++;
  }
}

for ( int i = 1; i <= n; i++ ) T[i] += T[i-1];

int k = -1;
for ( int i = 1; i <= n; i++ ){
  if ( (i-T[i]) >= (i+2)/2 ) {
     k = i;
     break;
  }
}

if ( k == -1 || k == n ) cout << "NA" << endl;
else cout << k << endl;
```

問題8　ねこまっしぐら2

問題概要

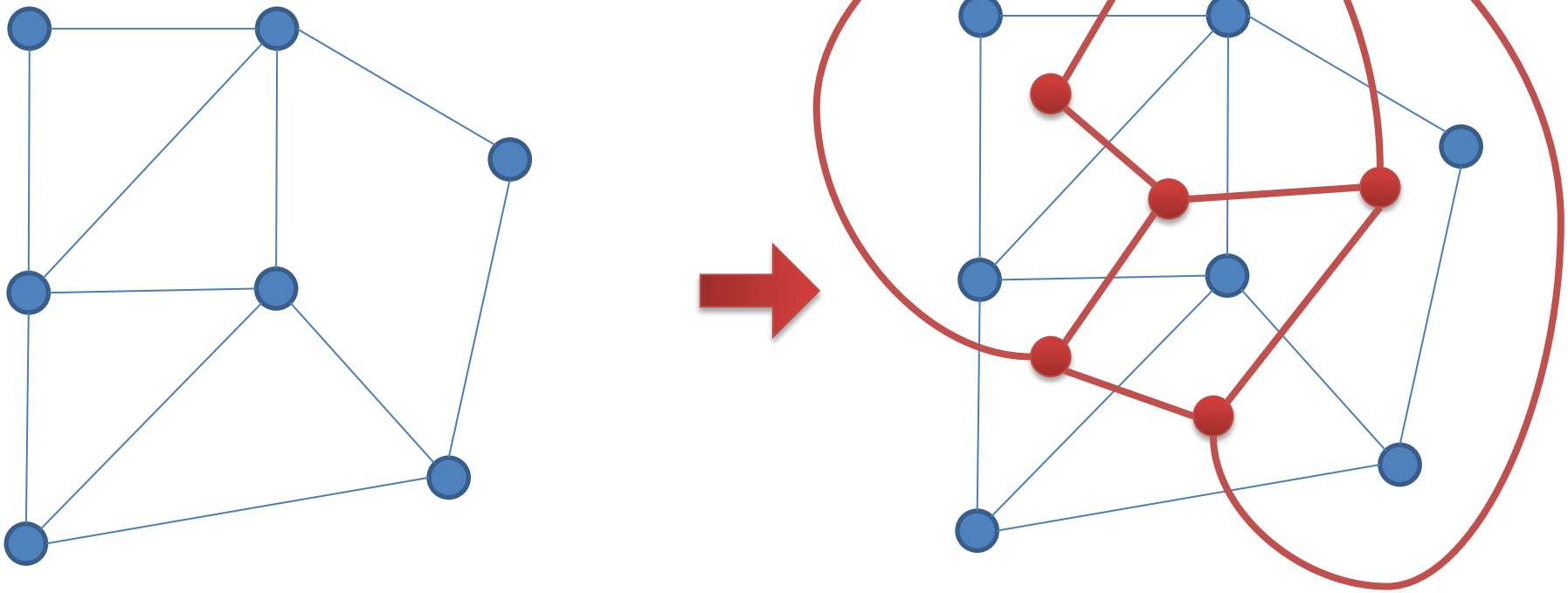
- 多角形で表されたいくつかの部屋で構成されたお屋敷の見取り図が与えられる.
- 見取り図は、お屋敷の柱の位置と、2つの柱を繋ぐ壁で与えられる.
- お屋敷の部屋のどこかに1匹の猫がいる.
- 猫は壁を通り抜けることによってお屋敷の外に出る.
- 猫はどの部屋にいるかわからない.
- 猫が最短の経路を選ぶとして、最大いくつの壁を通り抜けて外へ出てくるかを求めるプログラムを作成せよ.



問題8　ねこまっしぐら2

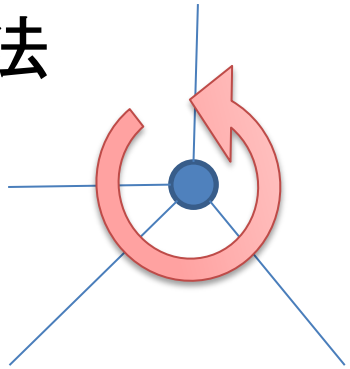
解法

- 部屋と屋敷の外部をノード、部屋と部屋または部屋と外部を仕切る壁をエッジとした双対グラフを作成する.
- 双対グラフの外部のノードから最も遠い部屋を幅優先探索で求める.

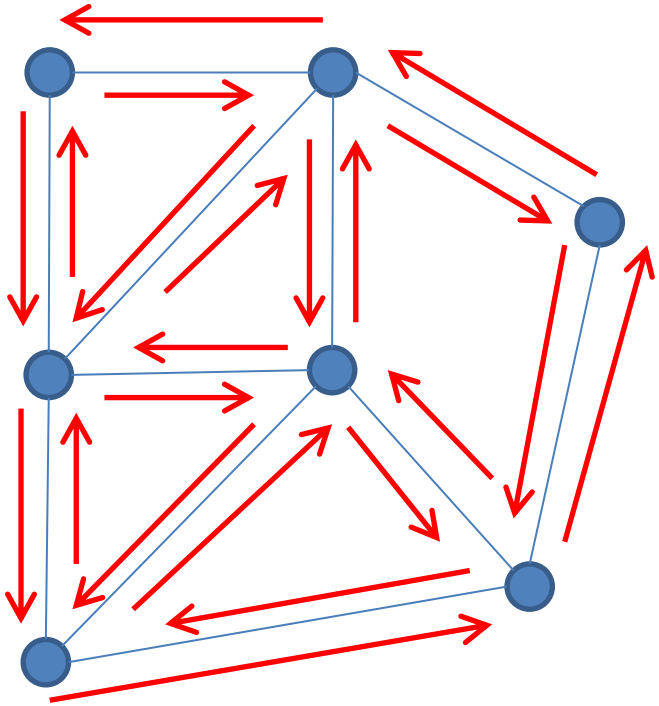


問題8　ねこまっしぐら2

解法



- 各ノードについて、そこから出ているエッジを反時計回りにソートしておく.



- 辺を時計回りに辿って領域を検出する.
- 壁を共有する領域同士を繋げ双対グラフを構築する.

問題９　図画工作

問題概要

- Ｍ個の課題からＮ人の生徒にそれぞれ１つ割り当てる.
- 各課題を完成させるための部品の種類と個数が決まっている.
- いくつかの部品が入った袋がＰ袋あり、各生徒は袋を１つ以下使うことができる.
- その他の部品は購入する必要がある.
- 日と部品の種類によって価格が変動する.
- 最小の費用はいくらか？

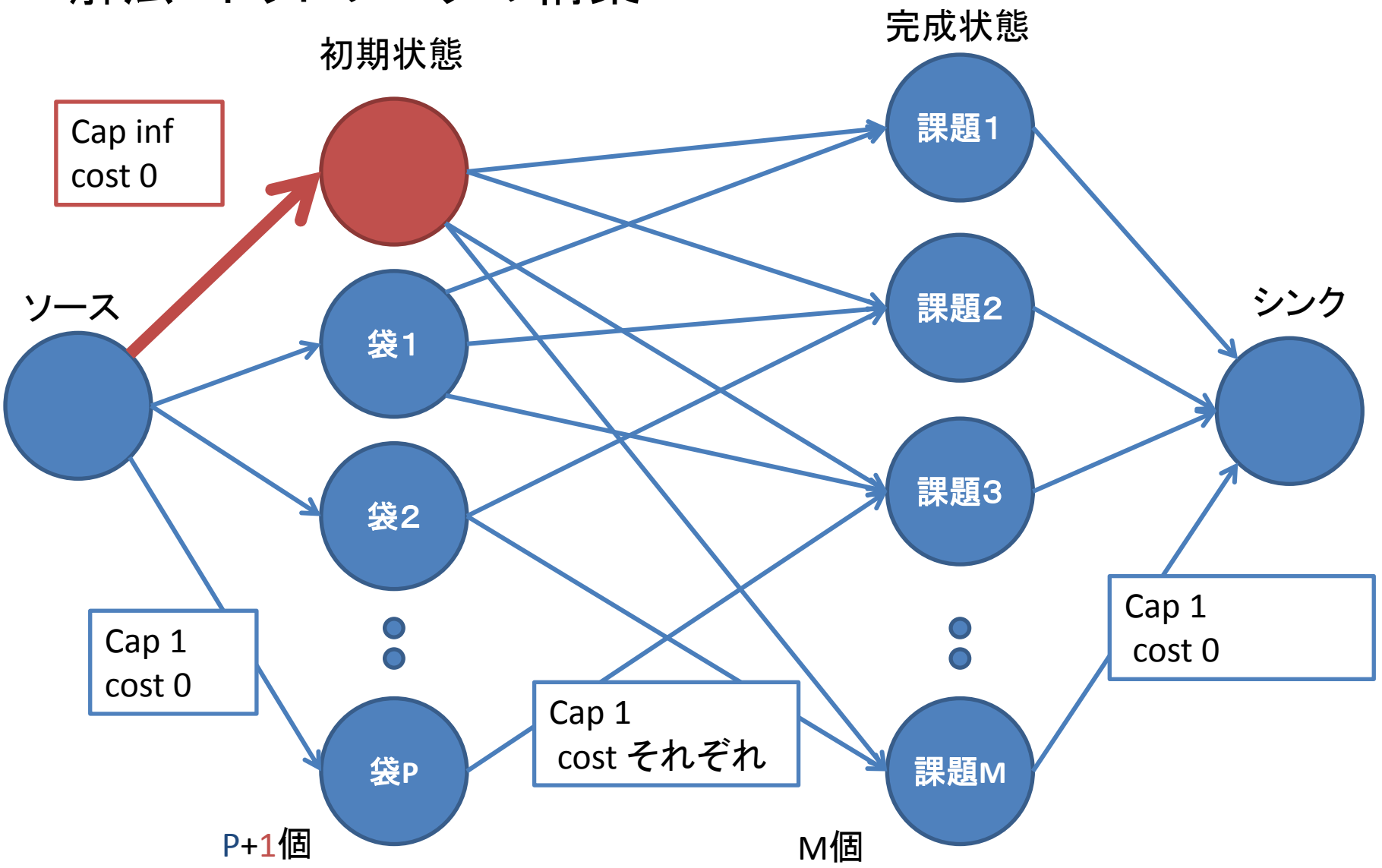
問題９　図画工作

解法

- 作品を作るために必要な最小コストを動的計画法により求め、ネットワークを作成する.
- ネットワークに対して、流量がNの最小費用流を適用する.

問題9　図画工作

解法：ネットワークの構築



問題９　図画工作

疑似コード：動的計画法

dp[i][j][a] := {
　　i:何日目か
　　j:部品購入状況
　　a:i日目に何個部品を買ったか}

for(i : 0 → D) for(j: 0 → 3^k) for(a: 0 → L) {
　dp[i+1][j][0] = min(dp[i+1][j][0], dp[i][j][a]);
　for(b: 0 → K){
　dp[i][j+3^b][a+1] = min(dp[i][j+3^b][a+1] ,
　　dp[i][j][a] + cost[i][b]);
}}

問題9　図画工作

計算量:　最小費用流$O(FElog_2V)$

- $F = N$
- $V = 2 + P + 1 + M$
- $E = P + 1 + (P + 1) \times M + M$
- $N \times P \times M \times log_2(P + M)$
- 約7000万

計算量:　動的計画法$O(D \times 3^K \times L \times K)$

- 約300万

問題１０　鉄道路線

問題概要

- 路線情報を表す重み付き有向グラフと、始点a終点bが与えられる.
- 始点c終点dの組で与えられるクエリにたいして、c->dの最短経路がa->bの最短経路上にあるかどうかを判定せよ.
- 頂点の数は最大100000、辺の数は最大200000
- クエリ数は最大40000.

講評・解法

- 最短経路グラフの辺の最大数は200000になり得るため、単純なダイクストラ法、深さ優先探索では時間切れとなる.

問題10　鉄道路線

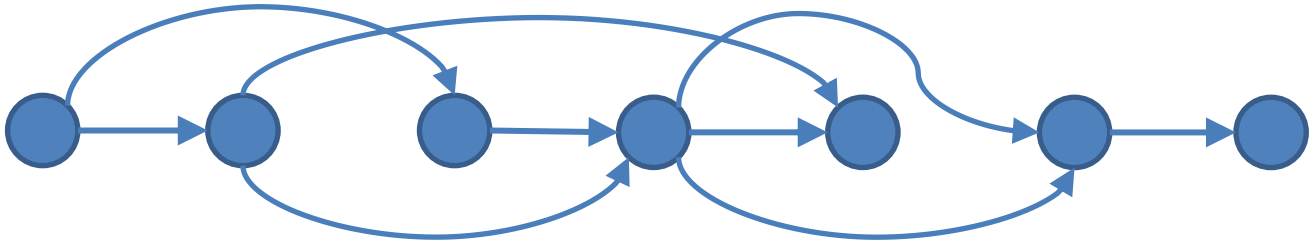
解法

- ダイクストラのアルゴリズムによりa->bにおける最短経路グラフを生成.
- 最短経路グラフにおいて、c->dが存在するかを調べる.
- 最短経路グラフをトポロジカルソート.
- 動的計画法で到達可能性を更新.

問題10　鉄道路線

解法

- ビット演算を用いた動的計画法（ビットDP）でクエリを64個づつ並列処理する.



64個

1						
		1				
	1					
…	…	…	…	…	…	…
…	…	…	…	…	…	…
				1		

問題10　鉄道路線

C++による解答例：ビットDPでクエリを並列処理

```
void valid(int n, int nedge,
              int sources[], int targets[],
              int bit, int qs[], int qt[], bool ans[]){
      static unsigned long long dp[NMAX];

      for ( int i = 0; i < n; i++ ) dp[i] = 0;

      for ( int i = 0; i < bit; i++ ){
          dp[qs[i]] |= ( 1ULL << i);
      }

      for ( int i = 0; i < nedge; i++ ){
          dp[targets[i]] |= dp[sources[i]];
      }

      for ( int i = 0; i < bit; i++ ){
          ans[i] = dp[qt[i]] & ( 1ULL << i);
      }
}
```