University of Aizu Reference

```
//
// Aho-Corasick で複数パターンの検索
// O(n + m)
//
// verified@ UVa10679
//
#include <iostream>
#include <string>
#include <queue>
#include <cstdio>
using namespace std;

const int C_LIM = 257; // char で表現出来るサイズ
const int W_LEN = 1001; // キーワード長
const int W_NUM = 1001; // キーワード数

struct Node
{
  Node *next[C_LIM];
  vector<int> out;
  vector<int> accept;

  Node() : out(), accept()
  {
    fill(next, next+C_LIM, (Node*)0);
  }

  void free()
  {
    for (int i=0; next[i]; ++i) {
      next[i]->free();
      delete next[i];
      next[i] = 0;
    }
  }
};

Node *root;

/**
 * p : オートマトンを作る為の文字列たち
 * pn : p の個数
 *
 * @remark
 *  'A'<= c <='z' (c:入力)
 */
void buildTree(char p[W_NUM][W_LEN], const int pn)
{
  // keyword tree & out function
  // O(|P1|+...+|Pn|) |Pi| : pattern length
  root = new Node();
  for (int i=0; i < pn; ++i) {
    Node *t = root;
    for (int j=0; p[i][j]; ++j) {
      const char c = p[i][j];
      if (t->next[c] == 0) {
        t->next[c] = new Node();
      }
      t = t->next[c];
    }
    t->accept.push_back(i);
  }

  // failture function
  queue<Node*> que;
  for (int c='A'; c <= 'z'; ++c) {
    if (root->next[c]) {
      root->next[c]->next[0] = root;
      que.push(root->next[c]);
    }
    else {
      root->next[c] = root;
    }
  }
  while (!que.empty()) {
    Node *t = que.front();
    que.pop();
    for (int c='A'; c <= 'z'; ++c) {
      if (t->next[c]) {
        que.push(t->next[c]);
        Node *r = t->next[0];
        while (r->next[c]==0) r = r->next[0];
        t->next[c]->next[0] = r->next[c];
      }
    }
  }
}

/**
 * base : 探す元になる文字列
```

```
 * root : 生成されたオートマトンのルート
 * result : 何回出現したかを記録
 *
 * @remark
 *   これを呼ぶ前に buildTree を呼んでおく。
 *   result の初期化を忘れない。
 */
void checkMatch(const char *base, Node *root, int *result)
{
  Node *v = root;
  const int n = strlen(base);
  for (int i=0; i < n; ++i) {
    const char c = base[i];
    while (!v->next[c]) v = v->next[0];
    v = v->next[c];
    for (int j=0; j < v->accept.size(); ++j) {
      ++result[v->accept[j]];
    }
  }
}
// 多角形の面積を求める
// pt の条件：
//   (半) 時計回り
//   pt[0] == pt[pt.size()-1]
// test@10065
double areaOfPolygon(const vector<P> &pt)
{
  double s = 0.0;
  for (int i=0; i < pt.size()-1; i++) {
    s += real(pt[i])*imag(pt[i+1]) - real(pt[i+1])*imag(pt[i]);
  }
  return abs(s / 2);
}// 橋を列挙する
// test@796

const int N = 1001;
bool adj[N][N];
int n;

int dfs(int st, int now, int visitTime[], int c, vector<pair<int,int> > &bridges)
{
  int ret = visitTime[now] = ++c;

  for (int i=0; i < n; ++i) {
    //if (!adj[now][i] && !adj[i][now]) continue;
    if (!adj[now][i]) continue;
    if (visitTime[i] != 0) {
      ret = min(ret, visitTime[i]);
      continue;
    }
    bool t1 = adj[now][i];
    bool t2 = adj[i][now];
    adj[now][i] = adj[i][now] = false;
    int up = dfs(st, i, visitTime, c, bridges);
    if (visitTime[i] <= up) {
      bridges.push_back(make_pair(min(now, i), max(now, i)));
    }
    ret = min(ret, up);
    adj[i][now] = t1;
    adj[now][i] = t2;
  }
  return ret;
}

void bridge(vector<pair<int,int> > &bridges)
{
  int visitTime[N] = { 0 };
  for (int i=0; i < n; ++i) {
    if (visitTime[i] != 0) continue;
    dfs(i, i, visitTime, 0, bridges);
  }
  sort(bridges.begin(), bridges.end());
}

// Gift-Wrapping で凸包を求める
// test@10065
void convexHull(const vector<P> &pt, vector<P> &result)
{
  int st = 0;
  for (int i=0; i < pt.size(); ++i) {
    if (pt[st].imag() >= pt[i].imag()) {
      if (pt[st].imag()==pt[i].imag() && pt[st].real() <= pt[i].real()) continue;
      st = i;
    }
  }
  int now = st;
  P prev = P(pt[st].real()-1, pt[st].imag());
  do {
    result.push_back(pt[now]);
    int min_index = -1;
    double min_arg = PI * 10;
    for (int i=0; i < pt.size(); ++i) {
```

```
      if (i==now || pt[i]==prev) continue;
      P p = (pt[i]-pt[now]) / (pt[now] - prev);
      double angle = arg(p);

      if (angle+EPS >= 0 && angle+EPS < min_arg) {
        min_arg = angle;
        min_index = i;
      }
      else if (EQ(angle, min_arg)) {
        // 同じ角度なら距離が遠い方を選ぶ
        if (abs(pt[min_index]-pt[now]) < abs(pt[i]-pt[now])) {
          min_index = i;
        }
      }
    }
    prev = pt[now];
    now = min_index;
  } while (now!=st);
}

class DisjointSet {
private:
  map<int, int> rank, parent;
public:
  DisjointSet() : rank(), parent() {}

  bool isExist(int x)
  {
    return (rank.find(x)!=rank.end());
  }

  void makeSet(int x)
  {
    parent[x] = x;
    rank[x] = 0;
  }

  void marge(int x, int y)
  {
    link(findSet(x), findSet(y));
  }

  int findSet(int x)
  {
    if (x!=parent[x]) {
      parent[x] = findSet(parent[x]);
    }
    return parent[x];
  }

  bool isSameSet(int x, int y)
  {
    if (x==y) return true;
    if (!isExist(x) || !isExist(y)) return false;
    return (findSet(x)==findSet(y));
  }

  void refreshParent()
  {
    map<int, int>::iterator itr = parent.begin();
    while (itr != parent.end()) {
      findSet(itr->first);
      ++itr;
    }
  }

  int maxCount()
  {
    refreshParent();

    map<int, int> c;
    map<int, int>::iterator itr = parent.begin();
    int result = 0;
    while (itr != parent.end()) {
      ++c[itr->second];
      result = max(c[itr->second], result);
      ++itr;
    }
    return result;
  }

  int groupCount()
  {
    int c = 0;
    map<int, int>::iterator itr = parent.begin();
    while (itr!=parent.end()) {
      if (itr->first==itr->second) ++c;
      ++itr;
    }
    return c;
  }
```

```
private:
  void link(int x, int y)
  {
    if (rank[x] >  rank[y]) {
      parent[y] = x;
    }
    else {
      parent[x] = y;
      if (rank[x]==rank[y]) rank[y]++;
    }
  }
};


// 射影を求める
P projection(const Line &l, const P &p)
{
  double t = dot(p-l.p1, l.p1-l.p2) / norm(l.p1-l.p2);
  return l.p1 + t*(l.p1-l.p2);
}

// 点 a と点 b との距離
double distPP(const P &a, const P &b)
{
  return abs(b-a);
}

// 直線と点の距離
// test@10709
double distLP(const Line &l, const P &p)
{
  return abs(p - projection(l, p));
}

// 線分と点の距離
// test@10709
double distSP(const Segment &s, const P &p)
{
  const P r = projection(s, p);
  if (intersectSP(s, r)) return abs(r - p);
  return (min(abs(s.p1-p), abs(s.p2-p)));
}

// 線分と線分の距離
// test@10709
double distSS(const Segment &s, const Segment &t)
{
  if (intersectSS(s, t)) return 0;
  return min(min(distSP(s, t.p1), distSP(s, t.p2)),
    min(distSP(t, s.p1), distSP(t, s.p2)));
}

// 直線と線分
// test@10709
double distLS(const Line &l, const Segment &s)
{
  if (intersectLS(l, s)) return 0;
  return min(distLP(l, s.p1), distLP(l, s.p2));
}

// 直線と直線の距離
// test@10709
double distLL(const Line &a, const Line &b)
{
  return (intersectLL(a, b) ? 0 : distLP(a, b.p1));
}
#include <iostream>
#include <string>
#include <algorithm>
using namespace std;

int main(void)
{
  string s1, s2;
  cin >> s1 >> s2;

  int d[100][100] = { 0 };

  for (int i=0; i <= s1.length(); i++) d[i][0] = i;
  for (int i=0; i <= s2.length(); i++) d[0][i] = i;

  for (int i=1; i <= s1.length(); i++) {
    for (int j=1; j <= s2.length(); j++) {
      d[i][j] = min( d[i-1][j]+1, d[i][j-1]+1 ); // 文字の挿入, 文字の削除
      d[i][j] = min( d[i-1][j-1]+(s1[i-1]!=s2[j-1]), d[i][j] ); // 文字の置換
      cout << d[i][j] << " ";
    }
    cout << endl;
  }

  cout << d[s1.length()][s2.length()] << endl;
```

```
  return 0;
}
//
// MaximumFlow を EdmondsKarp で求める
// O(E^2 * V)
//
// test@uva820
//
#include <iostream>
#include <vector>
#include <algorithm>
#include <utility>
#include <queue>
using namespace std;

const int UNCONNECTED = 0;

const int N=101;
int adj[N][N];
int n;

int flow[N][N]; // u,v 間のフロー
int cap[N][N]; // u,v 間の許容量

typedef pair<int,int> Edge;
typedef vector<Edge> Edges;

// BFS で増加可能経路を求める
void getGf(int s, int t, Edges &gf, int &minFlow)
{
  gf.clear();

  int parent[N];
  fill(parent, parent+N, -1);
  queue<int> que;
  que.push(s);

  while (que.empty()==false) {
    int now = que.front();
    que.pop();
    if (now==t) break;
    for (int v=0; v < n; ++v) {
      if (cap[now][v]-flow[now][v]<=0 || parent[v]!=-1) continue;
      parent[v] = now;
      que.push(v);
    }
  }
  if (parent[t]==-1) return;

  minFlow = INT_MAX;
  int p = t;
  while (p!=s) {
    int c = cap[parent[p]][p]-flow[parent[p]][p];
    minFlow = min(minFlow, c);
    gf.push_back(Edge(parent[p], p));
    p = parent[p];
  }
}

// s から t までの最大フローを求める
int edmondsKarp(const int s, const int t)
{
  fill(&flow[0][0], &flow[n-1][n], 0);
  copy(&adj[0][0], &adj[n-1][n], &cap[0][0]);

  int cost = 0;
  int minFlow = 0;
  Edges gf;
  while (getGf(s, t, gf, minFlow), gf.empty()==false)
  {
    for (int i=0; i < gf.size(); ++i) {
      int u = gf[i].first;
      int v = gf[i].second;
      flow[u][v] += minFlow;
      flow[v][u] -= minFlow;
    }
    cost += minFlow;
  }
  return cost;
}
// m >= n
int gcd(int m, int n)
{
  if (n==0) return m;
  return gcd(n, m % n);
}
static const double PI = acos(0.0)*2.0;

#define EQ(a,b) (abs((a)-(b)) < EPS)

typedef std::complex<double> P;
```

```
typedef P V;

bool lessP(const P &a, const P &b)
{
  if (a.real() != b.real()) return a.real() < b.real();
  return a.imag() < b.imag();
}

struct Line
{
  P p1, p2;
  Line(const P &a, const P &b)
    : p1(a), p2(b)
  {}
};

typedef Line Segment;


// 法線ベクトル
V normalV(const V &a)
{
  return a * P(0, 1);
  //return a * P(0, -1);
}

// 単位法線ベクトル
V unitNormalV(const V &a)
{
  return (a * P(0, 1)) / abs(a);
  //return (a * P(0, -1)) / abs(a);
}

// 内積
double dot(const V &a, const V &b)
{
  return (a.real() * b.real() + a.imag() * b.imag());
}

// 外積
double cross(const V &a, const V &b)
{
  return (a.real() * b.imag() - a.imag() * b.real());
}


// 与えられた2つのベクトルが作る角度(Rad)を返す
double getAngle(const V &a, const V &b)
{
  return acos(dot(a, b) / (abs(a)*abs(b)));
}

// test@いろいろ
double getAngle(Segment &s1, Segment &s2)
{
  P p = (s1.dst-s1.src) / (s2.dst-s2.src);
  return arg(p);
}

// test@いろいろ
double getAngle(P &a, P &b, P &c)
{
  P p1 = b-a;
  P p2 = c-b;
  return arg(p1/p2);
}

// 2直線の直交判定 => dot(a, b) = 0
bool isOrth(const V &a, const V &b)
{
  return EQ(dot(a, b), 0.0);
}

// 2直線の平行判定 => cross(a, b) = 0
bool isParallel(const V &a, const V &b)
{
  return EQ(cross(a, b), 0.0);
}

// a,bを通る直線とc,dを通る直線が同じか調べる
// test@uva378
bool isSameLine(const Line &a, const Line &b)
{
  return abs(cross(a.p2-a.p1, b.p1-a.p1)) < EPS;
}

// 2直線の平行判定
// test@uva378
bool isParallel(const Line &a, const Line &b)
{
  return EQ(cross(a.p2-a.p1, b.p2-b.p1), 0.0);
}
```

```
// a -> b -> c と進む時の
// return +1 : 時計周り
//         0 : 直線上
//        -1 : 半時計周り
//        +2 : c--a--b
//        -2 : a--b--c
// test@いろいろ
int ccw(P a, P b, P c)
{
  b-=a, c-=a;
  if (cross(c, b) > 0) return +1;
  if (cross(c, b) < 0) return -1;
  if (dot(b,c) < 0) return +2;
  if (norm(b) < norm(c)) return -2;
  return 0;
}

// 多角形内に点が含まれるか
// return +1 : 多角形内
//         0 : 線上
//        -1 : 多角形外
// test@uva191
int contains(vector<P> &v, const P &p)
{
  bool in = false;
  for (int i=0; i < v.size(); ++i) {
    P a = v[i]-p;
    P b = v[(i+1)%v.size()] - p;
    if (a.imag() > b.imag()) swap(a, b);
    if (a.imag() <= 0 && 0 < b.imag()) {
      if (cross(a,b) < 0) in = !in;
    }
    if (cross(a,b) == 0 && dot(a,b) <= 0) return 0;
  }
  return (in ? 1 : -1);
}


// 円の中に入っているか (円周上を含む)
bool isInCircle(const P &p, const P &cirPos, double r)
{
  return (r*r >= norm(cirPos-p));
}

// 三角形の中に入っているか (線上を含まない)
bool isInTriangle(const Point &p, const Point &a, const Point &b, const Point &c)
{
  Point g(0, 0);
  g.x = (a.x + b.x + c.x) / 3;
  g.y = (a.y + b.y + c.y) / 3;

  Point interP;
  if (intersectS(g, p, a, b, interP)) return false;
  if (intersectS(g, p, b, c, interP)) return false;
  if (intersectS(g, p, c, a, interP)) return false;

  return true;
}

// 内接円の半径
// a,b,c = 各辺の長さ
double incircleR(double a, double b, double c)
{
  double s = (a+b+c)/2;
  return sqrt(s*(s-a)*(s-b)*(s-c));
}

// 重心を求める
// test@uva10002
P getG(const vector<P> &v, double area)
{
  double sx=0.0, sy=0.0;
  for (int i=0; i < v.size()-1; ++i) {
    sx += ((v[i].real()+v[i+1].real())*cross(v[i],v[i+1]));
    sy += ((v[i].imag()+v[i+1].imag())*cross(v[i],v[i+1]));
  }
  return P(sx/(6.0*area), sy/(6.0*area));
}
// 線分 s と t の交差判定
// test@10709
bool intersectSS(const Segment &s, const Segment &t)
{
  return (
    ccw(s.p1,s.p2,t.p1)*ccw(s.p1,s.p2,t.p2) <= 0 &&
    ccw(t.p1,t.p2,s.p1)*ccw(t.p1,t.p2,s.p2) <= 0
  );
}

// 直線同士の当たり判定
// test@10709
```

```
bool intersectLL(const Line &a, const Line &b)
{
  return (abs(cross(a.p2-a.p1, b.p2-b.p1)) > EPS // non-parallel
    || abs(cross(a.p2-a.p1, b.p1-a.p1)) < EPS); // same line
}

// 直線と線分の当たり判定
// test@10709
bool intersectLS(const Line &l, const Segment &s)
{
  return (cross(l.p2-l.p1, s.p1-l.p1) *  // s.p1 と s.p2 のいる位置が逆かどうか
    cross (l.p2-l.p1, s.p2-l.p2) < EPS);
}

// 線分 s 上に p があるか (点上を含む)
// test@10709
bool intersectSP(const Segment &s, const P &p)
{
  double d = abs((s.p1-p)+(s.p2-p));
  double t = abs(s.p2-s.p1);
  return d-t<EPS;
}

// 点 c が直線 a,b 上にあるかないか
int intersectLP(const Line &l, const P &p)
{
  return EQ(cross(l.p2-l.p1, p-l.p1), 0.0);
}


/*
// 交差する線の交差位置を調べる
// test@uva378
P crossPointL(const Line &a, const Line &b)
{
  double ca = cross(a.p2 - a.p1, b.p2 - b.p1);
  double cb = cross(a.p2 - a.p1, a.p2 - b.p1);
  if (abs(ca) < EPS && abs(cb) < EPS) return b.p1; // same line
  //if (abs(ca) < EPS) assert(false);
  return b.p1 + cb/ca * (b.p2-b.p1);
}
*/

// 直線同士の交差位置
// test@uva438
P crossPointL(const Line &l1, const Line &l2)
{
  P a = l1.p2-l1.p1;
  P b = l2.p2-l2.p1;
  return (l1.p1+a*cross(b, l2.p1-l1.p1)/cross(b,a));
}

// 線分の交差位置
// 未テスト
P crossPointS(const Segment &s1, const Segment &s2)
{
  P b = s2.p2-s2.p1;
  double d1 = abs(cross(b, s1.p1-s2.p1));
  double d2 = abs(cross(b, s1.p2-s2.p1));
  double t = d1 / (d1+d2);
  return s1.p1 + (s1.p2-s1.p1)*t;
}
// n : 人数
// k : スキップ距離
// m : 開始位置
int joseph(int n, int k, int m)
{
  int j = 0, d;
  for (int i=2; i <= n; ++i) {
    d = (k-1) % i;
    j=(j+d)%(i-1);
    if (j>=d) ++j;
  }
  return (j-(k-m)%n+n)%n + 1;
}
// test@10130
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

const int ITEM_N = 1001;
const int W_LIM = 31;

const int  DIAGONAL = 1;
const int  LEFT = 0;

struct Item
{
  int value, weight;
};
```

```
int cost[ITEM_N+1][W_LIM+1];
int from[ITEM_N+1][W_LIM+1];

// limit : 入れ物の大きさ
int knapsack(const int limit, vector<Item> &items, vector<int> &path)
{
  fill(cost[0], cost[0]+W_LIM, 0);
  fill(from[0], from[0]+W_LIM, DIAGONAL);

  int n = items.size();
  for (int i = 1; i <= n; i++) cost[i][0] = 0;

  for (int i=1; i <= n; i++) {
    for (int w=1; w <= limit; w++) {
      if (items[i-1].weight <= w) {
        int nw = w - items[i-1].weight;
        if (items[i-1].value + cost[i-1][nw] > cost[i-1][w]) {
          cost[i][w] = items[i-1].value + cost[i-1][nw];
          from[i][w] = DIAGONAL;
        } else {
          cost[i][w] = cost[i-1][w];
          from[i][w] = LEFT;
        }
      } else {
        cost[i][w] = cost[i-1][w];
        from[i][w] = LEFT;
      }
    }
  }

  // path を求める
  path.clear();
  int w = limit;
  for (int i = n; i >=1; i--){
    if (from[i][w] == DIAGONAL){
      path.push_back(i);
      w -= items[i-1].weight;
    }
  }
  reverse(path.begin(), path.end());

  return cost[n][limit];
}
// lcs
// test@uva10405

static const int MAX = 1001;
int t[MAX][MAX] = { 0 };

int lcs(const string &s1, const string &s2)
{
  for (int i=1; i <= s1.length(); i++) {
    for (int j=1; j <= s2.length(); j++) {
      t[i][j] = max(t[i-1][j], t[i][j-1]);
      if (s1[i-1]==s2[j-1]) {
        t[i][j] = max(t[i-1][j-1]+1, t[i][j]);
      }
    }
  }
  return t[s1.length()][s2.length()];
}
const int N = 1000;

// test@uva497
void lis(const vector<int> &num, vector<int> &result)
{
  int t[N], parent[N];
  for (int i=0; i < num.size(); ++i) {
    t[i] = 1;
    parent[i] = -1;
    for (int j=0; j < i; ++j) {
      if (num[i] > num[j] && t[i] < t[j]+1)  {
        t[i] = t[j]+1;
        parent[i] = j;
      }
    }
  }

  int maxi = 0;
  for (int i=0; i < num.size(); ++i) {
    if (t[maxi] < t[i]) maxi = i;
  }

  int p = maxi;
  while (p!=-1) {
    result.push_back(num[p]);
    p = parent[p];
  }
  reverse(result.begin(), result.end());
}
// 最小全域木を求める
// (DisjointSet を使用)
```

```
// test@uva10034

struct Edge
{
  int src, dst;
  double cost;
  Edge(int s, int d, double c=0)
    : src(s), dst(d), cost(c)
  {}
  bool operator<(const Edge &e) const {
    return cost < e.cost;
  }
};

typedef vector<Edge> Edges;

// result : 最小全域木
// return : エッジの総コスト
double kruskal(Edges edge, Edges result, const int n)
{
  double cost = 0.0;
  DisjointSet ds;
  for (int i=0; i < n; ++i) ds.makeSet(i);
  sort(edge.begin(), edge.end());

  for (int i=0; i < edge.size(); ++i) {
    if (ds.findSet(edge[i].src) != ds.findSet(edge[i].dst)) {
      result.push_back(edge[i]);
      cost += edge[i].cost;
      ds.marge(edge[i].src, edge[i].dst);
    }
  }
  return cost;
}
// test@uva10600

const int N = 101;
int n;
int adj[N][N];


// result : 最小全域木
// if (木が作れなかった) return -1
// else エッジの総コスト
int prim(Edges &result, const int n)
{
  int cost = 0;

  bool visit[N] = { 0 };
  priority_queue<Edge, vector<Edge>, greater<Edge> > que;

  visit[0] = true;
  for (int i=1; i < n; ++i) {
    if (adj[0][i]==0) continue;
    que.push(Edge(0, i, adj[0][i]));
  }

  while (que.empty()==false) {
    Edge now = que.top();
    que.pop();

    if (visit[now.dst]) continue;
    visit[now.dst] = true;
    result.push_back(now);
    cost += now.cost;

    for (int i=0; i < n; i++) {
      if (adj[now.dst][i]==0 || visit[i]) continue;
      que.push(Edge(now.dst, i, adj[now.dst][i]));
    }
  }

  for (int i=0; i < n; ++i) {
    if (visit[i]==false) return -1; // 木が作れていない
  }
  return cost;
}

// 2番目に安いMSTのコストを求める
int secondBestMST(Edges &mst, const int n)
{
  int secondBest = INT_MAX;
  for (int i=0; i < mst.size(); ++i) {
    int t = adj[mst[i].src][mst[i].dst];
    adj[mst[i].src][mst[i].dst] = adj[mst[i].dst][mst[i].src] = 0;

    Edges result;
    int cost = prim(result, n);

    adj[mst[i].src][mst[i].dst] = adj[mst[i].dst][mst[i].src] = t;

    if (cost==-1) continue; // 木が作れなかった
```

```
    secondBest = min(cost, secondBest);
    adj[mst[i].src][mst[i].dst] = adj[mst[i].dst][mst[i].src] = t;
  }
  return secondBest;
}


const int N = 21;
long long c[N][N];

void makePascal()
{
  fill(&c[0][0], &c[N-1][N], 0);
  for (int i=1; i < N; ++i) {
    c[i][1] = c[i][i] = 1;
    for (int j=2; j < i; ++j) {
      c[i][j] = c[i-1][j-1]+c[i-1][j];
    }
  }
}
// 次元の maximum-sum を求める
#include <iostream>

using namespace std;

int main( void )
{
  while (true) {
    int N;
    cin >> N;
    if (N==0) break;

    int a[5000];
    for(int i = 0; i < N; i++) cin >> a[i];

    int t = 0;
    int s = a[0]-1;
    for (int i = 0; i < N; i++) {
      t = t + a[i];
      if (t > s) s = t;
      if (t < 0) t = 0;
    }
    cout << s << endl;
  }

  return 0;
}// 2次元の maximum-sum を求める
// test@uva いくつか

#include <iostream>
#include <algorithm>

using namespace std;

int main(void)
{
  int N;
  cin >> N;

  int a[100][100];
  for(int i = 0; i < N; i++)
    for(int j = 0; j < N; j++)
      cin >> a[i][j];

  int S = a[0][0]-1;
  for(int z = 0; z < N; z++) {
    int pr[100] = { 0 };
    for(int x = z; x < N; x++) {
      int t = 0;
      int s = a[x][0]-1;
      for (int i = 0; i < N; i++) {
        pr[i] = pr[i] + a[x][i];
        t = t + pr[i];
        if (t > s) s = t;
        if (t < 0) t = 0;
      }
      if (s > S) S = s;
    }
  }

  cout << S << endl;

  return 0;
}
// n 個の山 (t1,t2,t3...tn) からなる Nim ゲームの勝ち負けを調べる
// test@uva10165

#include <iostream>
using namespace std;

int main()
{
```

```
  int n;
  while (cin>>n,n) {
    int x = 0;
    for (int i=0; i < n; ++i) {
      int t;
      cin >> t;
      x ^= t;
    }
    if (x) cout << "Yes" << endl;
    else cout << "No" << endl;
  }
  return 0;
}
// 安定結婚問題
// test@uva.live-archive3837

const int N = 28;

int prefer[N][N]; // 男性の結婚したい順 ([0]={0},[][0]=0)
int fiancee[N]; // 女性から見た結婚相手 ([]=0)
int rank[N][N]; // 女性の結婚したい度 ([][0]=N+1)
bool use[N];

// O(N^2)
void stableMarriage(int n)
{
  fill(fiancee, fiancee+N, 0);
  int next[N] = { 0 }; // 何番目に結婚したい相手と結婚したか

  for (int m=1; m < N; ++m) {
    if (use[m]==false) continue;
    for (int s=m; s!=0;) {
      ++next[s];
      int w = prefer[s][next[s]];
      if (rank[w][s] < rank[w][fiancee[w]]) {
        swap(fiancee[w], s);
      }
    }
  }
}

// 強連結成分分解
// test@uva247

const int N = 101;
bool adj[N][N];
int n;

void dfs(const int iNode, vector<int>& order, vector<bool>& visited)
{
  if (!visited[iNode]) {
    visited[iNode] = true;
    for(int i = 0; i < n; i++) {
      if (adj[iNode][i]) {
        dfs(i, order, visited);
      }
    }
    order.push_back(iNode);
  }
}


// components : 強連結成分
void scc(vector<vector<int> > &components)
{
  vector<bool> visited(n, false);
  vector<int> order;
  for(int i = 0; i < n; i++) {
    dfs(i, order, visited);
  }
  reverse(order.begin(), order.end());

  // グラフを反転
  for (int i=0; i < n; ++i) {
    for (int j=i+1; j < n; ++j) {
      swap(adj[i][j], adj[j][i]);
    }
  }

  visited.assign(n, false);
  for(int i = 0; i < n; i++) {
    if (visited[order[i]]) continue;
    components.push_back(vector<int>());
    dfs(order[i], components.back(), visited);
  }

  // 元に戻す
  for (int i=0; i < n; ++i) {
    for (int j=i+1; j < n; ++j) {
      swap(adj[i][j], adj[j][i]);
    }
  }
```

```
}



/*
// リスト Ver
// isn't tested

typedef vector<vector<Edge> > Graph;

void dfs(const int iNode, Graph &g, vector<int>& order, vector<bool>& visited)
{
  if (!visited[iNode]) {
    visited[iNode] = true;
    const Edges &edges = g[iNode];
    for(int i = 0; i < edges.size(); i++) {
      dfs(edges[i].dest, g, order, visited);
    }
    order.push_back(iNode);
  }
}


/// 強連結成分分解
void scc(Graph &g, vector<vector<int> > &components)
{
  const int n = g.size();

  vector<bool> visited(n, false);
  vector<int> order;
  for(int i = 0; i < n; i++) {
    dfs(i, g, order, visited);
  }
  reverse(order.begin(), order.end());


  // グラフを反転
  Graph rGraph(n);
  for (int i=0; i < n; ++i) {
    const Edges &edges = g[i];
    for (int j=0; j < edges.size(); ++j) {
      Edge e(edges[j].dest, edges[j].src);
      rGraph[edges[j].dest].push_back(e);
    }
  }

  visited.assign(n, false);
  for(int i = 0; i < n; i++) {
    if (visited[order[i]]) continue;
    components.push_back(vector<int>());
    dfs(order[i], rGraph, components.back(), visited);
  }

}
*///************************************************************
// DFS 版 (path は逆順)
// test@10350
//************************************************************

const int N = 101;
bool adj[N][N];
int n;

void topologicalSort(int now, vector<int> &path, bool used[N])
{
  used[now] = true;
  for (int i=0; i < n; ++i) {
    if (used[i]) continue;
    if (adj[now][i]) topologicalSort(i, path, used);
  }
  path.push_back(now);
}

//************************************************************
// 後ろを削っていく版
// test@10350
//************************************************************

const int N = 101;
bool adj[N][N];
int n;

void topologicalSort(vector<int> &path)
{
  queue<int> que;
  for (int i=0; i < n; ++i) {
    bool f = true;
    for (int j=0; j < n; ++j) {
      if (adj[j][i]) {
        f = false;
        break;
      }
```

```
    }
    if (f) que.push(i);
  }

  bool used[N] = { 0 };

  while (que.empty()==false) {
    int now = que.front();
    que.pop();

    if (used[now]) continue;
    used[now] = true;
    path.push_back(now);

    for (int i=0; i < n; ++i) adj[now][i] = false;
    for (int i=0; i < n; ++i) {
      bool f = true;
      for (int j=0; j < n; ++j) {
        if (adj[j][i]) {
          f = false;
          break;
        }
      }
      if (f) que.push(i);
    }
  }
}

#include <iostream>
#include <cstdio>

#define REP(i,e) for(int i=0;i<(int)(e);i++)

using namespace std;

//const int L=500;
//const int B=10000;
//const int P=4;

template <int L,int B=10000,int P=4>
class BI{
  int d[L];
public:
  BI(){}
  BI(const int &n){
    REP(i,L) d[i]=0;
    d[0]=n;
    for(int i=0;d[i];i++)
      d[i+1]+=d[i]/B, d[i]%=B;
  }

  BI (string s){
    REP(i,L) d[i]=0;
    string t;
    int i;
    for(i=0;s.size();s.erase(max(0,(int)s.length()-P),P),i++)
      d[i]=atoi(s.substr(max(0,(int)s.length()-P),P).c_str());
  }
  BI operator+(const BI &a) const {
    BI r=a;
    REP(i,L)   r.d[i]+=d[i];
    REP(i,L-1) r.d[i+1]+=r.d[i]/B, r.d[i]%=B;
    return r;
  }

  BI operator-(const BI &a) const {
    BI r=*this;
    REP(i,L)   r.d[i]-=a.d[i];
    REP(i,L-1) if (r.d[i]<0) r.d[i]+=B,r.d[i+1]--;
    return r;
  }

  BI operator*(const int a) const {
    BI r=*this;
    REP(i,L) r.d[i]*=a;
    REP(i,L-1) r.d[i+1]+=r.d[i]/B, r.d[i]%=B;
    return r;
  }

  BI operator*(const BI &a) const {
    BI r(0);
    REP(i,L) r=r+((*this)*a.d[i]).lshift(i);
    return r;
  }

  BI operator=(const BI &a){
    REP(i,L) d[i]=a.d[i];
    return *this;
  }
  BI& lshift(const int p){
    for(int i=L-1;i-p>=0;i--) d[i]=d[i-p];
    REP(i,p) d[i]=0;
```

```
    return *this;
  }

  bool operator==(const BI &a) const {
    REP(i,L) if(d[i]!=a.d[i]) return false;
    return true;
  }

  BI operator/(const int a) const {
    BI r=*this;
    for(int i=L-1;i>0;i--)
      r.d[i-1]+=r.d[i]%a*B,r.d[i]/=a;
    r.d[0]/=a;
    return r;
  }
  string str(){
    string s;
    char buf[16];
    REP(i,L){
      sprintf(buf,"%0*d",P,d[i]);
      s=string(buf)+s;
    }
    while(s[0]=='0') s.erase(0,1);
    return (s.size() ? s : "0");
  }
};


main(){
  string a,b;
  BI n();
  while(cin >> a >> b)
    cout << ((BI(a)-BI(b))+BI(b)).str() << endl;
}
class DisjointSet {
  vector<int> v;
  int s; // number of disjoint sets in v
public:
  DisjointSet(int n) {v.resize(n,-1),s=n;}
  int findset(int n) {
    return (v[n]<0 ? n : (v[n]=findset(v[n])));
  }
  void unite(int a,int b){
    int sa=findset(a),sb=findset(b);
    if (sa!=sb){
      if (v[sa]>v[sb]) swap(sa,sb);
      v[sa]+=v[sb];
      v[sb]=sa; s--;
    }
  }
  int size() const { return s; }
  vector<int> sizes() {
    vector<int> n;
    REP(i,v.size()) if (v[i]<0) n.push_back(-v[i]);
    return n;
  }
};

/*
  Extended Euclidean Algorithm
    For given x and y, calculates A and B s.t. Ax + By = gcd(x,y).

    Arguments    : int           : x, y
    Return value : pair<int,int> : A, gcd(x,y)
    Notes        : B can be calculated by (gcd(x,y)-A*x)/y.
                   (BE CAREFUL ABOUT OVERFLOW!)
*/
pair<int,int> euclidEx(int a,int b,int m2=1,int m1=0){
  return (a%b ? euclidEx(b,a%b,m1,m2-m1*(a/b)) : make_pair(m1,b));
}

//  Needless to explain, right?
int gcd(int a,int b){
  return (b?gcd(b,a%b):a);
}

/*
    Multinomial Coefficient  (v[0], v[1], ... , v[n]) !

    Calculates coefficient of a[0]^v[0] * a[1]^v[1] * ... * a[n]^v[n]
    in terms of expansion of (a[0] + a[1] + ... + a[n])^m
    where m is v[0] + v[1] + ... + v[n].

    Multinomial Coefficient is calculated by following formula:
                (v[0], v[1], ... , v[n]) ! = (v[0] + v[1] + ... + v[n]) ! / (v[0] ! * v[1] ! * ... * v[n]!)

    Also, satisfies following:
                (v[0], v[1], ... , v[n]) ! = (v[0], v[1]) ! * (v[0] + v[1], v[2], ... , v[n]) !

    This implementation uses latter formula.

    Inspector : UVa 911 , Multinomial Coefficients
```

```
*/
typedef pair<int,int> P;
// Calculates (a+b)!/(a!b!), replace it if there's not risk of overflow.
int Multinomial_calc(int a,int b){
  if (a<b) swap(a,b);

  static map<P,int> m;
  if (!m.count(P(a,b))) {
    vector<int> u,l;
    for(int i=a+1;i<=a+b;i++) u.push_back(i);
    for(int i=1;i<=b;i++) l.push_back(i);
    for(int i=0;i<u.size();i++){
      for(int j=0;j<l.size();j++){
        int g=gcd(u[i],l[j]);
        u[i]/=g, l[j]/=g;
      }
    }
    m[P(a,b)]=accumulate(u.begin(),u.end(),1,multiplies<int>());
  }
  return m[P(a,b)];
}

int Multinomial(vector<int> &v){
  if (v.size()<2) return 1;
  int n=Multinomial_calc(v[v.size()-1],v[v.size()-2]);
  v[v.size()-2]+=v[v.size()-1];
  v.pop_back();
  return Multinomial(v)*n;
}
#include <iostream>
#include <cmath>

#define REP(i,e) for(int i=0;i<(int)(e);i++)
#define FOR(i,b,e) for(int i=(b);i<(int)(e);i++)

using namespace std;

const int N=10;
const double eps=1e-9;

bool pivswap(double m[N][N+1],int b,int n){
  int p=b;
  FOR(i,b,n) if(m[p][b]<m[i][b]) p=i;
  if(fabs(m[p][b])<eps) return false;
  REP(j,n+1) swap(m[p][j],m[b][j]);
  return true;
}

bool gauss(double m[N][N+1],int n){
  REP(i,n){
    if(!pivswap(m,i,n)) return false;
    FOR(j,i+1,n+1) m[i][j]/=m[i][i]; m[i][i]=1;
    FOR(j,i+1,n){ double v=m[j][i]; FOR(k,i,n+1) m[j][k]-=m[i][k]*v; }
  }
  for(int j=n-1;j;j--) REP(i,j) m[i][n]-=m[i][j]*m[j][n],m[i][j]=0;
  return true;
}


main(){
  int n=2;
  double matrix[N][N+1];

  while(cin >> n){
    REP(i,n) REP(j,n+1) if(!(cin >> matrix[i][j])) return 0;
    if(!gauss(matrix,n)) cout << "not solvable" << endl;
    REP(i,n) printf("%.3lf%c",
                    (fabs(matrix[i][n])>eps ?
                     matrix[i][n] : 0.0),
                    (i+1==n ? '\n' : ' '));
  }
}
#include <complex>
#include <vector>

const double eps=1e-9;
const double inf=1e256;
const double pi=3.1415926535897932;

typedef complex<double> pt;
struct seg{ pt p,d; seg(pt a,pt b) : p(a), d(b-a){}};
typedef seg line;

typedef vector<pt> gon;

namespace std{
  bool operator<(const pt &a,const pt b){
    return a.real()<b.real() || (a.real()==b.real() && a.imag()<b.imag());
  }
};

inline bool eq(double a,double b){ return fabs(a-b)<eps; }
```

```
inline bool pteq(pt a,pt b){ return eq(real(a),real(b))&&eq(imag(a),imag(b));}

inline double dot  (pt a,pt b){ return real(conj(a)*b); }
inline double cross(pt a,pt b){ return imag(conj(a)*b); }

//                 +1
// +2 --- (l.p) ----0---> (l.p+l.d) --- -2
//                 -1            (Same as Mr. Maebara's)
inline int ccw(line l,pt p){
  p-=l.p;
  double c=cross(l.d,p);
  if(!eq(c,0)) return (c<0 ? -1 : 1);
  if(dot(l.d,p)<-eps) return 2;
  if(norm(l.d)+eps<norm(p)) return -2;
  return 0;
}

// smaller angle of (a->b->c), in -pi to pi, negative when clockwise
inline double angle(pt a,pt b,pt c){ return arg((a-b)/(c-b)); }

const pt NO=pt(inf,inf);
pt crossPoint(line a,line b){
  double n=cross(b.p-a.p,b.d), d=cross(a.d,b.d);
  return (eq(d,0) ? NO : a.p+n/d*a.d);
}

double area(const gon &g){
  double result=0;
  REP(i,g.size()) result+=cross(g[i],g[(i+1)%g.size()]);
  return fabs(result/2);
}

// DO NOT WORK WHEN p IS ON EDGE OF g
bool insideGon(gon &g,pt p){
  double args=0;
  REP(i,g.size()) args+=angle(g[i], p, g[(i+1)%g.size()]);
  return eq(fabs(args),pi*2);
}

// cut the polygon by a line (counterclockwise side remains)
// may not be robust, CONSIDER USING long double!
gon cutPolygon(const gon &g,line l){
  gon result;
  REP(i,g.size()){
    pt p,a=g[i],b=g[(i+1)%g.size()];
    if(ccw(l,a)>=0) result.push_back(a);
    if(cross(l.d,a-l.p)*cross(l.d,b-l.p)<-eps&&(p=crossPoint(l,seg(a,b)))!=NO)
      result.push_back(p);
  }

  return result;
}

// NEED TO BE VERIFIED AGAIN DUE TO CHANGE OF line() and ccw()
gon ConvexHull(gon &g){
  sort(g.begin(),g.end());
  gon ch[2];
  REP(a,2){
    gon &c=ch[a];
    REP(i,g.size()){
      for(int s=c.size();
          s>1&&ccw((line){c[s-2],c[s-2]-c[s-1]},g[i])<=0;c.pop_back(),--s);
      c.push_back(g[i]);
    }

    c.pop_back();
    reverse(g.begin(),g.end());
  }

  ch[0].insert(ch[0].end(),ch[1].begin(),ch[1].end());
  return ch[0];
}

// NOT VERIFIED YET
inline bool intersectSS(seg a,seg b){
  return (ccw(a,b.p)*ccw(a,b.p+b.d)<=0) && (ccw(b,a.p)*ccw(b,a.p+a.d)<=0);
}

inline bool intersectLS(line a,seg b){
  return cross(a.d,b.p-a.p)*cross(a.d,b.p+b.d-a.p)<eps;
}
#include <complex>
#include <vector>

// constants
const double eps=1e-9;
const double inf=1e256;
const double pi=3.1415926535897932;

// type definition
typedef complex<double> pt;
struct seg{ pt p,d; seg(pt a,pt b) : p(a), d(b-a){}};
```

```
typedef seg line;

typedef vector<pt> gon;

// basic function
namespace std{
  bool operator<(const pt &a,const pt b){
    return a.real()<b.real() || (a.real()==b.real() && a.imag()<b.imag());
  }
};

inline bool eq(double a,double b){ return fabs(a-b)<eps; }
inline bool pteq(pt a,pt b){ return eq(real(a),real(b)) && eq(imag(a),imag(b));}

inline double dot  (pt a,pt b){ return real(conj(a)*b); }
inline double cross(pt a,pt b){ return imag(conj(a)*b); }

// counterclockwise
//                 +1
// +2 --- (l.p) ----0---> (l.p+l.d) --- -2
//                 -1              (Same as Mr. Maebara's)
inline int ccw(line l,pt p){
  p-=l.p;
  double c=cross(l.d,p);
  if(!eq(c,0)) return (c<0 ? -1 : 1);
  if(dot(l.d,p)<-eps) return 2;
  if(norm(l.d)+eps<norm(p)) return -2;
  return 0;
}

// smaller angle of (a->b->c), in -pi to pi, negative when clockwise
inline double angle(pt a,pt b,pt c){
  return arg((a-b)/(c-b));
}

// cross point of two lines
// Verify @ UOA 1283 (The Most Distant Point from the Sea)
const pt NO=pt(inf,inf);
pt crossPoint(line a,line b){
  double n=cross(b.p-a.p,b.d), d=cross(a.d,b.d);
  return (eq(d,0) ? NO : a.p+n/d*a.d);
}


// area of polygon, O(n)
double area(const gon &g){
  double result=0;
  REP(i,g.size()) result+=cross(g[i],g[(i+1)%g.size()]);
  return fabs(result/2);
}

// check whether p is inside g
// maybe it works even if g is concave (not verified)
// DO NOT WORK WHEN p IS ON EDGE OF g
// Verify @ UVa 109 (SCUD Busters) (only convex)
bool insideGon(gon &g,pt p){
  double args=0;
  REP(i,g.size()) args+=angle(g[i], p, g[(i+1)%g.size()]);
  return eq(fabs(args),pi*2);
}

// cut the polygon by a line (counterclockwise side remains)
// Verified @ UOA 1283 (The Most Distant Point from the Sea)
// may not be robust, CONSIDER USING long double!
gon cutPolygon(const gon &g,line l){
  gon result;
  REP(i,g.size()){
    pt p,a=g[i],b=g[(i+1)%g.size()];
    if(ccw(l,a)>=0) result.push_back(a);
    if(cross(l.d,a-l.p)*cross(l.d,b-l.p)<-eps&&(p=crossPoint(l,seg(a,b)))!=NO)
      result.push_back(p);
  }

  return result;
}

// NEED TO BE VERIFIED AGAIN DUE TO CHANGE OF line() and ccw()
// Andrew's algorithm, O(n log(n)), g will be destructed (sorted)
// Verify @ UVa 109 (SCUD Busters)
gon ConvexHull(gon &g){
  sort(g.begin(),g.end());
  gon ch[2];
  REP(a,2){
    gon &c=ch[a];
    REP(i,g.size()){
      for(int s=c.size();
          s>1&&ccw((line){c[s-2],c[s-2]-c[s-1]},g[i])<=0;c.pop_back(),--s);
      c.push_back(g[i]);
    }

    c.pop_back();
    reverse(g.begin(),g.end());
```

```
  }

  ch[0].insert(ch[0].end(),ch[1].begin(),ch[1].end());
  return ch[0];
}

// NOT VERIFIED YET
inline bool intersectSS(seg a,seg b){
  return (ccw(a,b.p)*ccw(a,b.p+b.d)<=0) && (ccw(b,a.p)*ccw(b,a.p+a.d)<=0);
}

inline bool intersectLS(line a,seg b){
  return cross(a.d,b.p-a.p)*cross(a.d,b.p+b.d-a.p)<eps;
}
/*******************************************************
  Shortest Path Algorithms
  * data structures are defined in graph_tree.cpp.
*******************************************************/

/*
  Bellman_Ford
    Arguments :
      1. Edges : List of the edges
      2. int   : number of the nodes
      3. int   : an initial node

    Return value :
      bool : true => no negative cycle

    Verification
      UVa558 Wormholes (Only negative cycle detection)
*/
bool Bellman_Ford(Edges &e,int n,int begin=0){
  int cost[n];
  int path[n];
  REP(i,n) cost[i]=inf,path[i]=begin;
  cost[begin]=0; path[begin]=-1;

  REP(i,n){
    REP(j,e.size()){
      if (cost[e[j].d]>cost[e[j].s]+e[j].w){
        cost[e[j].d]=cost[e[j].s]+e[j].w;
        path[e[j].d]=e[j].s;
      }
    }
  }

  REP(i,e.size())
    if (cost[e[i].d]>cost[e[i].s]+e[i].w)
      return false;
  return true;
}
/*
  Union-Find Tree

  NOTE : names of nodes are 1-ORIGIN. Do not use 0.
  Verification : UVa 11354 Bond
*/
class UFTNode {
public:
  int left,right,parent,weight;
  UFTNode(){}
};

class UnionFindTree{
  vector<int> v;
  vector<UFTNode> node;
public:
  UnionFindTree(int n){
    v.resize(n+1,-1);
    node.resize(1,Node());
  }

  void build(Edges &es){
    sort(es.begin(),es.end());
    REP(i,es.size())
      unite(es[i].s,es[i].d,es[i].w);
  }

  int findset(int n){
    return (v[n]==-1 ? -1 : (v[n]=findroot(v[n])));
  }
  int findroot(int n){
    return (node[n].parent==-1 ? n : (node[n].parent=findroot(node[n].parent)));
  }
  void unite(int a,int b,int weight){
    int sa=findset(a),sb=findset(b);

    if (!(sa==sb && sa!=-1 && sb!=-1)){
      node.push_back(Node());
      int nodenum=node.size()-1;
      node[nodenum].left=(sa==-1 ? -a : sa);
```

```
      node[nodenum].right=(sb==-1 ? -b : sb);
      node[nodenum].parent=-1;
      node[nodenum].weight=weight;
      if (sa==-1) v[a]=nodenum;
      else        node[sa].parent=nodenum;
      if (sb==-1) v[b]=nodenum;
      else        node[sb].parent=nodenum;
      nodenum++;
    }
  }

  int getroot() const {
    return node.size()-1;
  }

  void print(int root,int depth=0){
    if (node[root].right<0){
      REP(i,depth+1) cout << "    " ;
      cout << '[' << -node[root].right << ':' << v[-node[root].right] << ']' << endl;
    }
    else {
      print(node[root].right,depth+1);
    }

    REP(i,depth) cout << "    " ;
    cout << node[root].weight << ':' << node[root].pivot << endl;

    if (node[root].left<0){
      REP(i,depth+1) cout << "    " ;
      cout << '[' << -node[root].left << ':' << v[-node[root].left] << ']' << endl;
    }
    else {
      print(node[root].left,depth+1);
    }
  }
};
/*
  network.cpp
  Algorithms of Network Flow
*/


/*
  Edmonds-Karp
  Find the Max-flow of the network
  Input : int cap[N][N] (global)   := capacity of an edge of between two node
          int s,d,n     (argument) := source, destination, number of node
  Time  : O(E^2 log(U)) (U := size of max-flow)
  A BUG WAS FIXED, BUT NOT VERIFIED
*/

const int N=101;
const int nil=-1;
const int inf=1<<20;

int cap[N][N];
int net[N][N];
int res[N][N];

struct Edge{
  int s,d,c;
  Edge(int s,int d,int c) : s(s),d(d),c(c){}
};

int findpath(int s,int d,int n,vint &p){
  int cost[N],parent[N];
  REP(i,n) cost[i]=0,parent[i]=nil;
  REP(i,n) REP(j,n) res[i][j]=cap[i][j]-net[i][j];
  queue<Edge> qu;

  for(qu.push(Edge(nil,s,inf));qu.size();qu.pop()){
    Edge e=qu.front();
    if(e.c<=cost[e.d]) continue;
    cost[e.d]=e.c;
    parent[e.d]=e.s;
    REP(i,n) qu.push(Edge(e.d,i,min(e.c,res[e.d][i])));
  }

  p=vint(1,d);
  for(int n=parent[d];n!=nil;n=parent[n])
    p.push_back(n);
  reverse(ALL(p));

  return cost[d];
}

int EdmondsKarp(int s,int d,int n){
  vint p;
  for(int c;c=findpath(s,d,n,p);)
    REP(i,p.size()-1) net[p[i]][p[i+1]]+=c, net[p[i+1]][p[i]]-=c;
  int result=0;
  REP(i,n) result+=net[i][d];
```

```
  return result;
}

/*
  Dinic
  Find the Max-flow of the network
  Input : int cap[N][N] (global)   := capacity of an edge of between two node
                        (argument)   := source, destination, number of node
  Time  : O(V^2 E)
  Verify : UOA2076 Flame of Nucleus
*/

int net[N][N];
int cap[N][N];
int res[N][N];
const int nil=-1;

void calcLevel(int *l,int n,int s){
  REP(i,n) l[i]=inf;
  queue<pair<int,int> > qu;
  for(qu.push(make_pair(s,0));qu.size();qu.pop()){
    int pos=qu.front().first,cost=qu.front().second;
    if(cost<l[pos]){
      l[pos]=cost;
      REP(i,n) if(cap[pos][i]-net[pos][i]>0) qu.push(make_pair(i,cost+1));
    }
  }
}

int DFS(int s,int d,int n,vint &p,int c=inf){
  int cc;
  p.push_back(s);
  if(s==d) return c;
  REP(i,n) if(res[s][i]>0 && (cc=DFS(i,d,n,p,min(c,res[s][i])))) return cc;
  p.pop_back();
  return 0;
}

int Dinic(int n,int src,int snk){
  int result=0;
  static int level[N];
  memset(net,0,sizeof net);

  for(vint p;calcLevel(level,n,src),level[snk]<inf;){
    REP(i,n) REP(j,n)
      res[i][j]=(level[i]+1==level[j] ? cap[i][j]-net[i][j] : 0);

    for(int c;p.clear(),c=DFS(src,snk,n,p);result+=c)
      REP(i,p.size()-1)
        net[p[i]][p[i+1]]+=c,net[p[i+1]][p[i]]-=c,res[p[i]][p[i+1]]-=c;
  }

  return result;
}
// LCS
const int maxl;   // DON'T FORGET +1!
typedef pair<int,int> p;

int len[maxl][maxl];
p path[maxl][maxl];

int LCS(string &a,string &b){
  const int A=a.size(), B=b.size();
  REP(i,A) path[i][0]=p(i-1,0);
  REP(j,B) path[0][j]=p(0,j-1);
  REP(i,A+1) REP(j,B+1) len[i][j]=0;

  FOR(i,1,A+1) FOR(j,1,B+1){
    p& n=path[i][j];
    if(a[i-1]==b[j-1]) n=p(i-1,j-1), len[i][j]++;
    else if(len[i-1][j]<len[i][j-1]) n=p(i,j-1);
    else n=p(i-1,j);
    len[i][j]+=len[n.first][n.second];
  }

  return len[A][B];
}

void LCSsequence(string &a,string &b,string &c){
  int i=a.size(),j=b.size();
  c.clear(); c.reserve(len[i][j]);

  while(i&&j){
    if(path[i][j]==p(i-1,j-1))
      c+=a[i-1];
    p n=path[i][j];
    i=n.first, j=n.second;
  }

  reverse(c.begin(),c.end());
}
```

```
list<T> LIS(vector<T> &v){
  list<T> l;
  for(int i=0;i<v.size();i++){
    if(l.empty()) l.push_front(v[i]);
    else if(l.back()<v[i]) l.push_back(v[i]);
    else {
      list<T>::iterator it=lower_bound(l.begin(),l.end(),v[i]);
      if(v[i]<*it) *it=v[i];
    }
  }
  return l;
}


/*
  Shortest Common Supersequence
  CAUTION : Any string in input must not contain any other string.
  Verification : UOA 2022  Princess, a Cryptanalyst
*/
const int maxn=11;
int adj[maxn][maxn];

void calcadj(vector<string> &v){
  REP(i,v.size()) REP(j,v.size()){
    adj[i][j]=0;
    if (i==j) continue;
    for(int k=min(v[i].length(),v[j].length());k>0 && !adj[i][j];k--)
      if (v[j].substr(v[j].length()-k,k)==v[i].substr(0,k))
        adj[i][j]=k;
  }
}

string compute(vector<string> &v){
  calcadj(v);

  const int inf=1<<20;
  const int n=v.size();
  const string sinf(256,255);

  int cost[1<<n][n];
  struct{int s,p;} path[1<<n][n],p;
  REP(i,1<<n) REP(j,n)
    cost[i][j]=-inf,path[i][j].s=path[i][j].p=-1;
  REP(i,n) cost[1<<i][i]=0;

  REP(i,1<<n) REP(j,n){
    if (!(i&(1<<j))) continue;
    REP(k,n){
      if (i&(1<<k)) continue;
      const int t=(i|1<<k);
      if (cost[t][k]<cost[i][j]+adj[j][k] ||
          (cost[t][k]==cost[i][j]+adj[j][k] &&
           v[path[t][k].p].substr(adj[path[t][k].p][k],
                                  v[path[t][k].p].length()-
                                  adj[path[t][k].p][k])>
           v[j].substr(adj[j][k],v[j].length()-adj[j][k]))){

        cost[t][k]=cost[i][j]+adj[j][k];
        path[t][k].s=i;
        path[t][k].p=j;
      }
    }
  }

  int maxcost=*max_element(cost[(1<<n)-1],cost[(1<<n)-1]+n);
  string ans=sinf;

  REP(i,n){
    p.s=(1<<n)-1;
    p.p=i;
    if (cost[p.s][p.p]!=maxcost) continue;
    vector<int> ind;
    while(p.s>=0) ind.push_back(p.p),p=path[p.s][p.p];
    string result="";

    REP(i,ind.size()-1)
      result+=v[ind[i]].substr(0,v[ind[i]].length()-adj[ind[i+1]][ind[i]]);
    result+=v[ind.back()];
    ans=min(ans,result);
  }
  return ans;
}
#include <cstdio>
#include <queue>
#include <utility>

using namespace std;

#define NE 10000
#define NV 100
```

```
int ne, nv;
int src[2 * NE], dest[2 * NE];
int edge[NV], next[2 * NE];

#define REV(e) (2 * ne - 1 - (e))

void cons(){
  int e, v;
  for(e = 0; e < ne; e++)
    { src[REV(e)] = dest[e]; dest[REV(e)] = src[e]; }
  for(v = 0; v < nv; v++) edge[v] = -1;
  for(e = 0; e < 2 * ne; e++)
    { next[e] = edge[v = src[e]]; edge[v] = e; }
}

long long cap[2 * NE];

int level[NV];

int bfs(int s, int t){
  queue<int> q;
  int e, v, w;
  for(v = 0; v < nv; v++) level[v] = -1;
  level[s] = 0; q.push(s);
  while(!q.empty()){
    v = q.front(); q.pop();
    for(e = edge[v]; e + 1; e = next[e])
      if(cap[e] && level[w = dest[e]] == -1)
        { level[w] = level[v] + 1; q.push(w); }
  }
  return level[t] + 1;
}

long long flow(int s, int t, long long f){
  long long g = 0;
  int e, v;
  if(s == t) return f;
  for(e = edge[s]; e + 1; e = next[e])
    if(level[v = dest[e]] == level[s] + 1 &&
       (g = cap[e]) && (g = f < g ? f : g) && (g = flow(v, t, g)))
      { cap[e] -= g; cap[REV(e)] += g; break; }
  return g;
}

#define Inf ((long long)((1ull << 63) - 1))

void dinic(int s, int t){
  int e;
  while(bfs(s, t))while(flow(s, t, Inf));
}

long long cost[2 * NE];

long long dist[NV];

void dijkstra(int s){
  priority_queue< pair< long long, int > > q;
  long long c;
  int e, v, w;
  for(v = 0; v < nv; v++) dist[v] = Inf;
  q.push(make_pair(dist[s] = 0, s));
  while(!q.empty()){
    c = -q.top().first; v = q.top().second; q.pop();
    if(c == dist[v])for(e = edge[v]; e + 1; e = next[e])
      if(cap[e] && (c = dist[v] + cost[e]) < dist[w = dest[e]])
        q.push(make_pair(-(dist[w] = c), w));
  }
}

char visited[NV];

long long flow2(int s, int t, long long f){
  long long g = 0;
  int e, v;
  if(s == t) return f;
  visited[s] = 1;
  for(e = edge[s]; e + 1; e = next[e])
    if(cost[e] == 0 && visited[v = dest[e]] == 0 &&
       (g = cap[e]) && (g = (f < g ? f : g)) && (g = flow2(v, t, g)))
      { cap[e] -= g; cap[REV(e)] += g; break; }
  visited[s] = 0;
  return g;
}

int primalDual(int s, int t, long long f){
  long long g;
  int i;
  for(i = 0; i < ne; i++)
    { cap[REV(i)] = 0; cost[REV(i)] = -cost[i]; }
  while(f){
    dijkstra(s);
    if(dist[t] == Inf) return -1;
```

```
    for(i = 0; i < 2 * ne; i++)
      cost[i] -= dist[dest[i]] - dist[src[i]];
    while(g = flow2(s, t, f)) f -= g;
  }
  return 0;
}

#define N (1 << 8)

int cap[N][N], level[N], n, sink;

int bfs(int s){
  int q[N], *dq, *eq, sz;
  int v, w;
  dq = eq = q; sz = 0;
  for(v = 0; v < n; v++) level[v] = -1;
  level[s] = 0;
  *eq = s; if(++eq == q + N) eq = q; sz++;
  while(sz){
    v = *dq; if(++dq == q + N) dq = q; sz--;
    for(w = 0; w < n; w++)
      if(cap[v][w] && level[w] == -1){
        level[w] = level[v] + 1;
        *eq = w; if(++eq == q + N) eq = q; sz++;
      }
  }
  return level[sink] + 1;
}

int flow(int s, int f){
  int g, h;
  int v;
  if(s == sink) return f;
  h = 0;
  for(v = 0; v < n; v++)
    if(level[v] == level[s] + 1 &&
       (g = cap[s][v]) && (g = f < g ? f : g) && (g = flow(v, g)))
      { cap[s][v] -= g; cap[v][s] += g; f -= g; h += g; }
  return h;
}

#define Inf ((int)((1u << 31) - 1))

void gomoryHu(int cap0[N][N], int p[N], int c[N]){
  int u, v, w;
  p[0] = -1;
  for(u = 1; u < n; u++) p[u] = 0;
  for(u = 1; u < n; u++){
    for(v = 0; v < n; v++)
      for(w = 0; w < n; w++)
        cap[v][w] = cap0[v][w];
    sink = p[u];
    c[u] = 0;
    while(bfs(u)) c[u] += flow(u, Inf);
    for(v = 0; v < n; v++)
      if(v != u && p[v] == p[u] && level[v] != -1) p[v] = u;
  }
}

void makeTable(int p[N], int c[N], int table[N][N]){
  int mn[N], m;
  int i, j, v;
  for(j = 0; j < n; j++){
    for(v = 0; v < n; v++) mn[v] = Inf;
    for(v = j; p[v] + 1; v = p[v]){
      mn[p[v]] = mn[v];
      if(c[v] < mn[p[v]]) mn[p[v]] = c[v];
    }
    for(i = 0; i < n; i++){
      m = Inf;
      for(v = i; v != j && mn[v] == Inf; v = p[v])
        if(c[v] < m) m = c[v];
      if(mn[v] < m) m = mn[v];
      table[j][i] = m;
    }
  }
}
```

バグ取りのチェック項目

自分がハマってしまった典型的罠を追加しましょう。

バグが取れない …

* 定義した operator は正しいか。特に、return false が必要なもの。
* テストケースの全入力を終わる前に、ループを終了していないか。

Wrong Answer

* 問題番号は合っているか。
* 各変数の初期化はきちんとなされているか。
* 前に処理したデータセットの影響を受けることはないか。
* 境界値に対して正しい処理をできるか。
* デバックに使った出力を残していないか。
* 計算の結果が NaN にならないか（ゼロ割り算などで）
* 浮動小数点出力で負の 0 が出ることはないか。
* 符号付き演算に、配列サイズなどの符号無し整数を入れていると符号無しに拡張され、負の値が扱えなくなる。
* 必要なソートを行っているか。
* 問題を正しく理解しているか。
* 根本的なアルゴリズムは間違っていないか。
* 工夫したコードを書いて、自分で上手いなと思っていたところが実は間違っていたという「落ち」はないか。
* シフト演算を含む式の演算子の適用順位は正しいか。
* minv, maxv 等の値は初期化されているか。
* n と MAX の使い方は正しいか。n = テストケースに対するサイズ、MAX = 対応する最大値。

Runtime Error

* 問題番号は合っているか。
* 大きすぎる静的配列をクラス内部や関数内部で確保していないか。
* 配列の範囲外にアクセスするような危険性は無いか。
* 再帰を利用する場合に循環してしまう可能性は無いか。
* 再帰があまりに深すぎないか。再帰の最大の深さを保障できるか。
* 0 除算をしている可能性は無いか。
* 値を返すメソッドで return をきちんと行っているか。

Time Limit Exceeded

* 問題番号は合っているか。
* オーダーの計算は正しいか。
* ループの脱出条件や再帰の終端条件が正しくかつ安全か。
* 動的に巨大なメモリを確保している可能性はないか。
* 入力を量に適した方法で取っているか。
* 操作に適したデータ構造を利用しているか。
* 同じ計算を潜在的に何度も実行していないか。
* 再帰をループに展開できないか。
* bool の vector はありがた迷惑な最適化で、最悪のパフォーマンスになる可能性がある。