Contents

1 計算幾何

1.1 基本儲存

```
struct Pt{
    double x, y;
};
struct Line{
    Pt st, ed;
};
struct Circle{
    Pt o; // 圓心
    double r;
};
struct Poly{
    int n; // n邊形
    vector<Pt> pts;
};
```

1.2 距離

歐基里德距離

$$d(p_1,p_2)=\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$$

曼哈頓距離

$$d(p_1,p_2)=|x_1-x_2|+|y_1-y_2|$$

1.3 內積、外積

$$\vec{v_1}\cdot\vec{v_2}=x_1x_2+y_1y_2$$

$$\vec{v_1}\times\vec{v_2}=x_1y_2-x_2y_1$$

1.4 多邊形面積

$$\frac{1}{2}|\sum_{i=1}^{n}\overrightarrow{OP_i}\times\overrightarrow{OP_{i+1}}|$$

1.5 點與線段距離

看點與線段兩端內積，若為負數則說明角度大於 90 度則距離為點到該端點的距離若內積皆 >=0，則距離為三角形面積/線段

1.6 判斷點是否在線段上

```
bool collinearity(Pt p1, Pt p2, Pt p3){ // 三點共線
    return cross(p2 - p1, p3 - p1) == 0;
}
bool inLine(Pt st, Pt ed, Pt p){ // 點是否在線上
    return collinearity(st, ed, p) && dot(p, st, ed) <
        0;
}
```

1.7 線段相交、交點

```
bool intersect(Pt a, Pt b, Pt c, Pt d){ // 線段相交
    return (cross(b - a, c - a) * cross(b - a, d - a) <
         0 && cross(d - c, a - c) * cross(d - c, b - c)
         < 0)
           || inLine(a, b, c) || inLine(a, b, d) ||
               inLine(c, d, a) || inLine(c, d, b);
}
Pt intersection(Pt a, Pt b, Pt c, Pt d){ // 線段交點
    assert(intersect(a, b, c, d)); // 沒有交點的狀況
    return a + cross(a - c, d - c) * (b - a) / cross(d
        - c, b - a);
}
```

1.8 點在多邊形內部

射線法: 若點在多邊形內，則隨機選一個方向的射線出現會碰到奇數次邊而如果碰到多邊形的點，如果射線碰到多邊形的點則重選 (需要特判點是否在多邊形的邊或頂點上)

1.9 凸包

```
 vector<Pt> convex_hull(vector<Pt> hull){
    sort(hull.begin(),hull.end());
    int top=0;
    vector<Pt> stk;
    for(int i=0;i<hull.size();i++){
        while(top>=2&&cross(stk[top-2],stk[top-1],hull[
            i])<=0)
            stk.pop_back(),top--;
        stk.push_back(hull[i]);
        top++;
    }
    for(int i=hull.size()-2,t=top+1;i>=0;i--){
        while(top>=t&&cross(stk[top-2],stk[top-1],hull[
            i])<=0)
            stk.pop_back(),top--;
        stk.push_back(hull[i]);
        top++;
    }
    stk.pop_back();
    return stk;
}
```

1.10 旋轉卡尺-最遠點對

```
double FarthestPair(vector<Pt> arr){ // 需要先凸包
    double ret=0;
    for(int i = 0, j = i+1; i<arr.size(); i++){
        while(distance(arr[i], arr[j]) <= distance(arr[
            i], arr[(j+1)%arr.size()]) ){
            j = (j+1) % arr.size();
        }
        ret = max(ret, distance(arr[i],arr[j]));
    }
    return ret;
}
```

1.11 極角排序

```
bool cmp(const Pt& lhs, const Pt rhs){
    if((lhs < Pt(0, 0)) ^ (rhs < Pt(0, 0)))
        return (lhs < Pt(0, 0)) < (rhs < Pt(0, 0));
    return (lhs ^ rhs) > 0;
} // 從 270 度開始逆時針排序
sort(P.begin(), P.end(), cmp);
```

1.12 皮克定理 (多邊形內整數點數量)

$$A = i + \frac{b}{2} - 1$$

A: 多邊形面積 i: 內部整數點個數 b: 線上整數點個數

1.13 三分搜-最小包覆圓

平面上給 n 個點，求出半徑最小的圓要包住所有的點。求出圓心位置與與最小半徑。複雜度 $(N\log^2 N)$

```
Pt arr[MXN];
double checky(double x, double y) { //搜半徑
  double cmax = 0;
  for(int i = 0; i < n; i++) {
    cmax = max(cmax,(arr[i].x - x) * (arr[i].x - x) +
                (arr[i].y - y) * (arr[i].y - y));
  }// 過程中回傳距離^2 避免不必要的根號運算
  return cmax;
}
double checkx(double x){ //有了x再搜y
    double yl = -1e9, yr = 1e9;
    while(yr - yl > EPS) {
        double ml = (yl+yl+yr) / 3, mr = (yl+yr+yr) /
            3;
        if (checky(x, ml) < checky(x, mr))    yr = mr;

        else                                  yl = ml;
    }
}
double xl = -1e9, xr = 1e9; //先搜x
while(xr - xl > EPS) {
  double ml = (xl+xl+xr) / 3, mr = (xl+xr+xr) / 3;
  if (checkx(ml) < checkx(mr))    xr = mr;
  else                            xl = ml;
}
```

1.14 旋轉矩陣、鏡射矩陣

逆時針轉 θ 角

$$\begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}$$

對與 x 軸正向夾角為 θ 的直線 L 鏡射

$$\begin{bmatrix} cos2\theta & sin2\theta \\ sin2\theta & -cos2\theta \end{bmatrix}$$

2 樹論

2.1 LCA

```
int timing;
int in[N],out[N];
void dfs(int u){
    in[u] = ++timing;//這時進入u
    for(int nxt : g[u])//跑過所有孩子
        dfs(nxt);
    out[u] = ++timing;//這時離開u
}
bool is_ancestor(int u,int v){ //用=因為自己是自己的祖
    先
    return in[u] <= in[v] && out[u] >= out[v]; //u是v的
        祖先
}
int getlca(int x, int y){
    if(is_ancestor(x, y))return x; // 如果 u 為 v 的祖
        先則 lca 為 u
    if(is_ancestor(y, x))return y; // 如果 v 為 u 的祖
        先則 lca 為 u
    for(int i=logN;i>=0;i--){    // 判斷 2^logN, 2^(
        logN-1),...2^1, 2^0 倍祖先
        if(!is_ancestor(anc[x][i], y)) // 如果 2^i 倍祖
            先不是 v 的祖先
            x = anc[x][i];            // 則往上移動
    }
    return anc[x][0]; // 回傳此點的父節點即為答案
}
int anc[N][logN]; //倍增法，從x往上走i步
signed main(){
    for(int i=1;i<=log2(N);i++){
        for(int now=1;now<=N;now++){
            anc[now][i]=anc[anc[now][i-1]][i-1];
        }
    }
}
```

2.2 換根 DP

```
void dfs(int u, int fa) {  // 預處裡dfs
  sz[u] = 1; // 以 u 為根的子樹數量
  dep[u] = dep[fa] + 1; // u 的深度
  for (int v : edge[u]) { //遍歷 u 的子節點
    if (v != fa) { //不等於父親
      dfs(v, u);
      sz[u] += sz[v];
    }
  }
}
void get_ans(int u, int fa) {  // 第二次dfs換根dp
  for (int v : edge[u]) { //遍歷子節點
    if (v != fa) {
      dp[v] = dp[u] - sz[v] * 2 + n; //轉移式
      get_ans(v, u);
    }
  }
}
```

3 資料結構

3.1 離散化

```
vector<int> tmp(arr);  //將arr複製到tmp
sort(tmp.begin(), tmp.end());
tmp.erase(unique(tmp.begin(), tmp.end()), tmp.end());
for (int i = 0; i < n; i++)
  arr[i] = lower_bound(tmp.begin(), tmp.end(), arr[i])
      - tmp.begin();
```

3.2 線段樹

```
// 區間修改 查詢區間和
#define cl(x) (x<<1)
#define cr(x) (x<<1)+1
int seg[4*N], lazy[4*N], arr[N];
void build(int id, int l, int r){
    if(l == r){
        seg[id] = arr[l];
        return;
    }
    int mid = (l + r) >> 1;
    build(cl(id), l, mid);
    build(cr(id), mid+1, r);
    seg[id] = seg[cl(id)] + seg[cr(id)];
}
void propagation(int id, int l, int r){
    if(lazy[id]){
        seg[id] += (r - l + 1) * lazy[id];
        if(l != r){
            lazy[cl(id)] += lazy[id];
            lazy[cr(id)] += lazy[id];
        }
```

```
        lazy[id] = 0;
    }
}
int query(int id, int l, int r, int ql, int qr){
    propagation(id, l, r);
    if (ql > r || qr < l) return 0;
    if(ql <= l && qr >= r) return seg[id];
    int mid = (l + r) >> 1;
    return query(cl(id), l, mid, ql, qr) + query(cr(id)
        , mid+1, r, ql, qr);
}
void update(int id, int l, int r, int sl, int sr, int v
    ){
    propagation(id, l, r);
    if(sl > r || sr < l) return;
    if(sl <= l && r <= sr){
        lazy[id] += v;
        propagation(id, l, r);
        return;
    }
    int mid = (l + r) >> 1;
    update(cl(id), l, mid, sl, sr, v);
    update(cr(id), mid+1, r, sl, sr, v);
    seg[id] = seg[cl(id)] + seg[cr(id)];
}
```

4 數論

4.1 階乘與模逆元

```
long long fac[MXN], inv[MXN];
fac[0] = 1; // 0! = 1
for(long long i = 1; i <= N; i++)
    fac[i] = fac[i-1] * i % MOD;
inv[N] = FastPow(fac[N], MOD-2); // 快速冪
for(long long i = N-1; i >=0; i--)
    inv[i] = inv[i+1] * (i+1) % MOD;
```

4.2 擴展歐基里德

```
int exgcd(int a,int b,long long &x,long long &y) {
    if(b == 0){x=1,y=0;return a;}
    int now=exgcd(b,a%b,y,x);
    y-=a/b*x;
    return now;
}
long long inv(long long a,long long m){ //求模逆元
    long long x,y;
    long long d=exgcd(a,m,x,y);
    if(d==1) return (x+m)%m;
    else return -1; //-1為無解
}
```

4.3 中國剩餘定理

```
LL exgcd(LL a,LL b,LL &x,LL &y){
    if(!b){
        x = 1, y = 0;
        return a;
    }
    int now=exgcd(b, a % b, y, x);
    y -= a / b * x;
    return now;
}
LL CRT(LL k, LL* a, LL* r) {
    LL n = 1, ans = 0;
    for (LL i = 1; i <= k; i++) {
        n = n * r[i];
    }
    for (LL i = 1; i <= k; i++) {
        LL m = n / r[i], b, y;
        exgcd(m, r[i], b, y);
        ans = (ans + a[i] * m * b % n) % n;
    }
    return (ans % n + n) % n;
}
```

4.4 進制轉換

```
int ntod(string str, int n){ // n進制轉10進制
    int ans = 0;
    for(int i = 0; i < str.size(); i++){
        if(str[i] >= '0' && str[i]<='9')
            ans = ans * n + str[i] - '0';
        else// 小寫減a 大寫減A
            ans = ans * n + str[i] - 'a' + 10;
    }
    return ans;
}
string dton(int num , int n){ // 10進制轉n進制
    string ans = "";
    do{
        int t = num % n;
        if(t >= 0 && t <= 9)
            ans += t + '0';
        else
            ans += t - 10 + 'a';
        num /= n;
    } while(num != 0);
    reverse(ans.begin(), ans.end());
    return ans;
}
```

4.5 O(1)mul

```
LL mul(LL x,LL y,LL mod){
  // LL ret=x*y-(LL)((long double)x/mod*y)*mod;
    //4捨5入，避免浮點數誤差
  LL ret=x*y-(LL)((long double)x*y/mod+0.5)*mod;
  return ret<0?ret+mod:ret;
}
```

4.6 Miller Rabin

```
// n < 4,759,123,141        3 :  2, 7, 61
// n < 2^64                       7 :
// 2, 325, 9375, 28178, 450775, 9780504, 1795265022
// 或前12個質數
#define LL __int128
LL magic[]={}
bool witness(LL a,LL n,LL u,int t){
  if(!a) return 0;
  LL x=mypow(a,u,n);
  for(int i=0;i<t;i++) {
    LL nx=mul(x,x,n);
    if(nx==1&&x!=1&&x!=n-1) return 1;
    x=nx;
  }
  return x!=1;
}
bool miller_rabin(LL n) {
  int s=(magic number size)
  // iterate s times of witness on n
  if(n<2) return 0;
  if(!(n&1)) return n == 2;
  ll u=n-1; int t=0;
  // n-1 = u*2^t
  while(!(u&1)) u>>=1, t++;
  while(s--){
    LL a=magic[s]%n;
    if(witness(a,n,u,t)) return 0;
  }
  return 1;
}
```

4.7 Pollard Rho

```
// does not work when n is prime  O(n^(1/4))
LL f(LL x, LL c, LL mod){ return add(mul(x,x,mod),c,mod
    ); }
LL pollard_rho(LL n) {
    LL c = 1, x = 0, y = 0, p = 2, q, t = 0;
    while (t++ % 128 or gcd(p, n) == 1) {
        if (x == y) c++, y = f(x = 2, c, n);
        if (q = mul(p, abs(x-y), n)) p = q;
        x = f(x, c, n); y = f(f(y, c, n), c, n);
    }
    return gcd(p, n);
}
```

4.8 FFT

```
// const int MAXN = 262144;
// (must be 2^k)
```

```
//steps: pre_fft->mul
typedef long double ld;
typedef complex<ld> cplx; //real() ,imag()
const ld PI = acosl(-1);
const cplx I(0, 1);
cplx omega[MAXN+1];
void pre_fft(){
  for(int i=0; i<=MAXN; i++)
    omega[i] = exp(i * 2 * PI / MAXN * I);
}
// n must be 2^k
void fft(int n, cplx a[], bool inv=false){
  int basic = MAXN / n;
  int theta = basic;
  for (int m = n; m >= 2; m >>= 1) {
    int mh = m >> 1;
    for (int i = 0; i < mh; i++) {
      cplx w = omega[inv ? MAXN-(i*theta%MAXN)
                         : i*theta%MAXN];
      for (int j = i; j < n; j += m) {
        int k = j + mh;
        cplx x = a[j] - a[k];
        a[j] += a[k];
        a[k] = w * x;
    } }
    theta = (theta * 2) % MAXN;
  }
  int i = 0;
  for (int j = 1; j < n - 1; j++) {
    for (int k = n >> 1; k > (i ^= k); k >>= 1);
    if (j < i) swap(a[i], a[j]);
  }
  if(inv) for (i = 0; i < n; i++) a[i] /= n;
}
cplx arr[MAXN+1];
inline void mul(int _n,ll a[],int _m,ll b[],ll ans[]){
  int n=1,sum=_n+_m-1;
  while(n<sum)
    n<<=1;
  for(int i=0;i<n;i++) {
    double x=(i<_n?a[i]:0),y=(i<_m?b[i]:0);
    arr[i]=complex<double>(x+y,x-y);
  }
  fft(n,arr);
  for(int i=0;i<n;i++)
    arr[i]=arr[i]*arr[i];
  fft(n,arr,true);
  for(int i=0;i<sum;i++)
    ans[i]=(long long int)(arr[i].real()/4+0.5);
}
```

4.9 約瑟夫問題

```
int josephus(int n, int m){ //n人每m次
    int ans = 0;
    for (int i=1; i<=n; ++i)
        ans = (ans + m) % i;
    return ans;
}
```

4.10 快速求歐拉函數

```
int prime[10010], phi[10010];
bool v[10010];
void quick_euler(){
  int cnt = 0;
  for(int i = 2; i <= N; ++i){
    if(!v[i]) prime[++cnt] = i, phi[i] = i - 1;
      // 若 i 是質數，所以 φ(i) = i - 1
    for(int j = 1; i * prime[j] <= N && j <= cnt; ++j){
      v[i * prime[j]] = 1;
      if(i % prime[j] == 0){
        phi[i * prime[j]] = phi[i] * prime[j];
        break;
      }
      else phi[i * prime[j]] = phi[i] * (prime[j] - 1)
          ;
    }
  }
}
```

5 圖論

5.1 最短路徑

dijkstra $O(V^2+E)$

```
vector<pair<int,int>>vec[N];
void dijkstra(int s,int t){//起點，終點
    int dis[N];
    for(int i=0;i<N;i++){//初始化
        dis[i]=INF;//值要設為比可能的最短路徑權重還要大
            的值
    }
    dis[s]=0;
    priority_queue<pii,vector<pii>,greater<pii>>pq;//以
        小到大排序
    pq.push({dis[s],s});
    while(pq.empty()==0){
        int u=pq.top().second;
        pq.pop();
        if(vis[u])continue;
        vis[u]=1;
        for(auto [v,w]:vec[u]){
            if(dis[u]+w<dis[v]){//鬆弛
                dis[v]=dis[u]+w;
                pq.push({dis[v],v});
            }
        }
    }
}
```

floyd-warshall $O(N^3)$

```
for(int k=1;k<=N;k++){//窮舉中繼點k
    for(int i=1;i<=N;i++){
        for(int j=1;j<=N;j++){//窮舉點對(i,j)
            dis[i][j]=min(dis[i][j],dis[i][k]+dis[k][j
                ]);
        }
    }
}
```

5.2 歐拉回路、漢米爾頓路徑

```
vector<int> path;
void dfs(int x){
    while(!edge[x].empty()){
        int u = edge[x].back();
        edge[x].pop_back();
        dfs(u);
    }
    path.push_back(x);
}
int main(){
    dfs(st);
    reverse(path.begin(),path.end());
    for(int i:path)    cout<<i<<' ';
    cout<<endl;
}
```

```
dp[3][26]=dp[3][11010] //現在的點為3，走過1,3,4這三個點
if( edge[i][j] && ( (1<<j) & s ) == 0 ){
    //i->j有邊且點j尚未走過
    dp[j][s|(1<<j)]=dp[i][s];
}
//以下為程式碼
for(int s=0;s<(1<<n);s++){//枚舉點集合
    for(int i=0;i<n;i++){//枚舉現在的點
        if(s&(1<<i)==0)continue;
        for(int j=0;j<n;j++){//枚舉下一個點
            if(i==j)continue;
            if( edge[i][j] && ( (1<<j) & s ) == 0 ){
                dp[j][s|(1<<j)]=dp[i][s];
            }
        }
    }
}
```

5.3 點雙連通分量

```
//step: init(n)->addEdge(u,v)->solve()
//return:二維vector
```

```
#define PB push_back
#define REP(i, n) for(int i = 0; i < n; i++)
struct BccVertex {
  int n,nScc,step,dfn[MXN],low[MXN];
  vector<int> E[MXN],sccv[MXN];
  int top,stk[MXN];
  void init(int _n) {
    n = _n; nScc = step = 0;
    for (int i=0; i<n; i++) E[i].clear();
  }
  void addEdge(int u, int v)
  { E[u].PB(v); E[v].PB(u); }
  void DFS(int u, int f) {
    dfn[u] = low[u] = step++;
    stk[top++] = u;
    for (auto v:E[u]) {
      if (v == f) continue;
      if (dfn[v] == -1) {
        DFS(v,u);
        low[u] = min(low[u], low[v]);
        if (low[v] >= dfn[u]) {
          int z;
          sccv[nScc].clear();
          do {
            z = stk[--top];
            sccv[nScc].PB(z);
          } while (z != v);
          sccv[nScc++].PB(u);
        }
      }else
        low[u] = min(low[u],dfn[v]);
  } }
  vector<vector<int>> solve() {
    vector<vector<int>> res;
    for (int i=0; i<n; i++)
      dfn[i] = low[i] = -1;
    for (int i=0; i<n; i++)
      if (dfn[i] == -1) {
        top = 0;
        DFS(i,i);
      }
    REP(i,nScc) res.PB(sccv[i]);
    return res;
  }
}graph;
```

5.4 強連通分量

```
//step: init(n)->addEdge(u,v)->solve()
//有nScc個強連通分量 bln是點i所在的連通分量編號
#define PB push_back
#define FZ(x) memset(x, 0, sizeof(x)) //fill zero
struct Scc{
  int n, nScc, vst[MXN], bln[MXN];
  vector<int> E[MXN], rE[MXN], vec;
  void init(int _n){
    n = _n;
    for (int i=0; i<= n; i++)
      E[i].clear(), rE[i].clear();
  }
  void addEdge(int u, int v){
    E[u].PB(v); rE[v].PB(u);
  }
  void DFS(int u){
    vst[u]=1;
    for (auto v : E[u]) if (!vst[v]) DFS(v);
    vec.PB(u);
  }
  void rDFS(int u){
    vst[u] = 1; bln[u] = nScc;
    for (auto v : rE[u]) if (!vst[v]) rDFS(v);
  }
  void solve(){
    nScc = 0;
    vec.clear();
    fill(vst, vst+n+1, 0);
    for (int i=0; i<n; i++)
      if (!vst[i]) DFS(i);
    reverse(vec.begin(),vec.end());
    fill(vst, vst+n+1, 0);
    for (auto v : vec)
      if (!vst[v]){
        rDFS(v); nScc++;
      }
  }
};
```

6 動態規劃

6.1 0/1 背包

$O(NW)$

```
for (int i = 1; i <= cnt; i++) //幾個物品
   for (int j = weight; j >= w[i]; j--) //從物品耐重上限
       枚舉到此物品的重量，代表每個最多選一次
     dp[j] = max(dp[j], dp[j - w[i]] + v[i]);
```

6.2 無限背包

$O(NW)$

```
for(int i = 1; i <= cnt; i++)
    for(int j = w[i]; j <= weight; j++)
        dp[j] = max(dp[j], dp[j - w[i]] + v[i]);
```

6.3 有限背包

$O(NW \log k)$

```
// 有限背包二進制拆分
int index = 0;
for(int i = 1; i <= m; i++){
    int c = 1, p, h, k;
    cin >> p >> h >> k;
    while(k > c){
        k -= c;
        list[++index].w = c * p;
        list[index].v = c * h;
        c *= 2;
    }
    list[++index].w = p * k;
    list[index].v = h * k;
}
// 之後再去做0/1背包
```

7 字串

7.1 字典樹

```
struct trie{
    struct node{
        node *nxt[26];
        int cnt, sz;
        node():cnt(0),sz(0){
            memset(nxt,0,sizeof(nxt));
        }
    };
    node *root;
    void init(){root = new node();}
    void insert(const string& s){
        node *now = root;
        for(auto i:s){
            now->sz++;
            if(now->nxt[i-'a'] == NULL){
                now->nxt[i-'a'] = new node();
            }
            now = now->nxt[i-'a'];
        }
        now->cnt++;
        now->sz++;
    }
};
```

7.2 KMP

```
vector<int> KMP(string s, string t){
    s = t + '@' + s;
    int sz = s.size();
    vector<int> pi(sz);
    for(int i = 1; i < sz; i++){
        int len = pi[i-1];
        while(len != 0 && s[i] != s[len]) len = pi[len
            -1];
        if(s[i] == s[len]) pi[i] = len+1;
    }
    return pi;
}
```

7.3 Hash

```
// 如果發生碰撞可以做雙重雜湊
const ll P = 75577; // p = 13331, 14341, 75577
const ll MOD = 998244353; //MOD = 1e9+7, 998244353,
    999997771
ll Hash[MXN];    //Hash[i] 為字串 [0,i] 的 hash值
void build(const string& s){
    int val = 0;
    for(int i=0; i<s.size(); i++){
        val = (val * P + s[i]) % MOD;
        Hash[i] = val;
    }
}
// h[l, r] = h[r] - h[l-1] * p^(r-l+1)
```

7.4 Suffix Array

```
const int N = 300010;
struct SA{
#define REP(i,n) for ( int i=0; i<int(n); i++ )
#define REP1(i,a,b) for ( int i=(a); i<=int(b); i++ )
  bool _t[N*2];
  int _s[N*2], _sa[N*2], _c[N*2], x[N], _p[N], _q[N*2],
      hei[N], r[N];
  int operator [] (int i){ return _sa[i]; }
  void build(int *s, int n, int m){
    memcpy(_s, s, sizeof(int) * n);
    sais(_s, _sa, _p, _q, _t, _c, n, m);
    mkhei(n);
  }
  void mkhei(int n){
    REP(i,n) r[_sa[i]] = i;
    hei[0] = 0;
    REP(i,n) if(r[i]) {
      int ans = i>0 ? max(hei[r[i-1]] - 1, 0) : 0;
      while(_s[i+ans] == _s[_sa[r[i]-1]+ans]) ans++;
      hei[r[i]] = ans;
    }
  }
  void sais(int *s, int *sa, int *p, int *q, bool *t,
      int *c, int n, int z){
    bool uniq = t[n-1] = true, neq;
    int nn = 0, nmxz = -1, *nsa = sa + n, *ns = s + n,
        lst = -1;
#define MS0(x,n) memset((x),0,n*sizeof(*(x)))
#define MAGIC(XD) MS0(sa, n); \
    memcpy(x, c, sizeof(int) * z); \
    XD; \
    memcpy(x + 1, c, sizeof(int) * (z - 1)); \
    REP(i,n) if(sa[i] && !t[sa[i]-1]) sa[x[s[sa[i
        ]-1]]++] = sa[i]-1; \
    memcpy(x, c, sizeof(int) * z); \
    for(int i = n - 1; i >= 0; i--) if(sa[i] && t[sa[i
        ]-1]) sa[--x[s[sa[i]-1]]] = sa[i]-1;
    MS0(c, z);
    REP(i,n) uniq &= ++c[s[i]] < 2;
    REP(i,z-1) c[i+1] += c[i];
    if (uniq) { REP(i,n) sa[--c[s[i]]] = i; return; }
    for(int i = n - 2; i >= 0; i--) t[i] = (s[i]==s[i
        +1] ? t[i+1] : s[i]<s[i+1]);
    MAGIC(REP1(i,1,n-1) if(t[i] && !t[i-1]) sa[--x[s[i
        ]]]=p[q[i]=nn++]=i);
    REP(i, n) if (sa[i] && t[sa[i]] && !t[sa[i]-1]) {
      neq=lst<0||memcmp(s+sa[i],s+lst,(p[q[sa[i]]+1]-sa
          [i])*sizeof(int));
      ns[q[lst=sa[i]]]=nmxz+=neq;
    }
    sais(ns, nsa, p + nn, q + n, t + n, c + z, nn, nmxz
        + 1);
    MAGIC(for(int i = nn - 1; i >= 0; i--) sa[--x[s[p[
        nsa[i]]]]] = p[nsa[i]]);
  }
}sa;
int H[ N ], SA[ N ];
void suffix_array(int* ip, int len) {
  // should padding a zero in the back
  // ip is int array, len is array length
  // ip[0..n-1] != 0, and ip[len] = 0
  ip[len++] = 0;
  sa.build(ip, len, 128);
  for (int i=0; i<len; i++) {
    H[i] = sa.hei[i + 1];
    SA[i] = sa._sa[i + 1];
  }
  // resulting height, sa array \in [0,len)
}
```

7.5 馬拉車

```
void z_value_pal(char *s,int len,int *z){
  len=(len<<1)+1;
  for(int i=len-1;i>=0;i--)
    s[i]=i&1?s[i>>1]:'@';
  z[0]=1;
  for(int i=1,l=0,r=0;i<len;i++){
    z[i]=i<r?min(z[l+l-i],r-i):1;
    while(i-z[i]>=0&&i+z[i]<len&&s[i-z[i]]==s[i+z[i]])
        ++z[i];
    if(i+z[i]>r) l=i,r=i+z[i];
} }
```

7.6 Zvalue

```
int z[MAXN];
void Z_value(const string& s) { //z[i] = lcp(s[1...],s[
    i...])
  int i, j, left, right, len = s.size();
  left=right=0; z[0]=len;
  for(i=1;i<len;i++) {
    j=max(min(z[i-left],right-i),0);
    for(;i+j<len&&s[i+j]==s[j];j++);
    z[i]=j;
    if(i+z[i]>right) {
      right=i+z[i];
      left=i;
}   }   }
```

7.7 minRotation

```
//rotate(begin(s),begin(s)+minRotation(s),end(s))
int minRotation(string s) {
  int a = 0, N = s.size(); s += s;
  rep(b,0,N) rep(k,0,N) {
    if(a+k == b || s[a+k] < s[b+k])
      {b += max(0, k-1); break;}
    if(s[a+k] > s[b+k]) {a = b; break;}
  } return a;
}
```

7.8 回文樹

```
// len[s]是對應的回文長度
// num[s]是有幾個回文後綴
// cnt[s]是這個回文子字串在整個字串中的出現次數
// fail[s]是他長度次長的回文後綴，aba的fail是a
const int MXN = 1000010;
struct PalT{
  int nxt[MXN][26],fail[MXN],len[MXN];
  int tot,lst,n,state[MXN],cnt[MXN],num[MXN];
  int diff[MXN],sfail[MXN],fac[MXN],dp[MXN];
  char s[MXN]={-1};
  int newNode(int l,int f){
    len[tot]=l,fail[tot]=f,cnt[tot]=num[tot]=0;
    memset(nxt[tot],0,sizeof(nxt[tot]));
    diff[tot]=(l>0?l-len[f]:0);
    sfail[tot]=(l>0&&diff[tot]==diff[f]?sfail[f]:f);
    return tot++;
  }
  int getfail(int x){
    while(s[n-len[x]-1]!=s[n]) x=fail[x];
    return x;
  }
  int getmin(int v){
    dp[v]=fac[n-len[sfail[v]]-diff[v]];
    if(diff[v]==diff[fail[v]])
        dp[v]=min(dp[v],dp[fail[v]]);
    return dp[v]+1;
  }
  int push(){
    int c=s[n]-'a',np=getfail(lst);
    if(!(lst=nxt[np][c])){
      lst=newNode(len[np]+2,nxt[getfail(fail[np])][c]);
      nxt[np][c]=lst; num[lst]=num[fail[lst]]+1;
```

```
    }
    fac[n]=n;
    for(int v=lst;len[v]>0;v=sfail[v])
        fac[n]=min(fac[n],getmin(v));
    return ++cnt[lst],lst;
  }
  void init(const char *_s){
    tot=lst=n=0;
    newNode(0,1),newNode(-1,1);
    for(;_s[n];) s[n+1]=_s[n],++n,state[n-1]=push();
    for(int i=tot-1;i>1;i--) cnt[fail[i]]+=cnt[i];
  }
}palt;
```

8 網路流

8.1 Dinic

```
#define PB push_back
#define SZ(x) (int)x.size()
struct Dinic{
  struct Edge{ int v,f,re; };
  int n,s,t,level[MXN];
  vector<Edge> E[MXN];
  void init(int _n, int _s, int _t){
    n = _n; s = _s; t = _t;
    for (int i=0; i<n; i++) E[i].clear();
  }
  void add_edge(int u, int v, int f){
    E[u].PB({v,f,SZ(E[v])});
    E[v].PB({u,0,SZ(E[u])-1});
  }
  bool BFS(){
    for (int i=0; i<n; i++) level[i] = -1;
    queue<int> que;
    que.push(s);
    level[s] = 0;
    while (!que.empty()){
      int u = que.front(); que.pop();
      for (auto it : E[u]){
        if (it.f > 0 && level[it.v] == -1){
          level[it.v] = level[u]+1;
          que.push(it.v);
    } } }
    return level[t] != -1;
  }
  int DFS(int u, int nf){
    if (u == t) return nf;
    int res = 0;
    for (auto &it : E[u]){
      if (it.f > 0 && level[it.v] == level[u]+1){
        int tf = DFS(it.v, min(nf,it.f));
        res += tf; nf -= tf; it.f -= tf;
        E[it.v][it.re].f += tf;
        if (nf == 0) return res;
    } }
    if (!res) level[u] = -1;
    return res;
  }
  int flow(int res=0){
    while ( BFS() )
      res += DFS(s,2147483647);
    return res;
} }flow;
```

8.2 最小花費最大流

```
struct MinCostMaxFlow{
typedef int Tcost;
  static const int MAXV = 20010;
  static const int INFf = 1000000;
  static const Tcost INFc  = 1e9;
  struct Edge{
    int v, cap;
    Tcost w;
    int rev;
    Edge(){}
    Edge(int t2, int t3, Tcost t4, int t5)
    : v(t2), cap(t3), w(t4), rev(t5) {}
  };
  int V, s, t;
  vector<Edge> g[MAXV];
  void init(int n, int _s, int _t){
    V = n; s = _s; t = _t;
    for(int i = 0; i <= V; i++) g[i].clear();
  }
  void addEdge(int a, int b, int cap, Tcost w){
    g[a].push_back(Edge(b, cap, w, (int)g[b].size()));
    g[b].push_back(Edge(a, 0, -w, (int)g[a].size()-1));
  }
  Tcost d[MAXV];
  int id[MAXV], mom[MAXV];
  bool inqu[MAXV];
  queue<int> q;
  pair<int,Tcost> solve(){
    int mxf = 0; Tcost mnc = 0;
    while(1){
      fill(d, d+1+V, INFc);
      fill(inqu, inqu+1+V, 0);
      fill(mom, mom+1+V, -1);
      mom[s] = s;
      d[s] = 0;
      q.push(s); inqu[s] = 1;
      while(q.size()){
        int u = q.front(); q.pop();
        inqu[u] = 0;
        for(int i = 0; i < (int) g[u].size(); i++){
          Edge &e = g[u][i];
          int v = e.v;
          if(e.cap > 0 && d[v] > d[u]+e.w){
            d[v] = d[u]+e.w;
            mom[v] = u;
            id[v] = i;
            if(!inqu[v]) q.push(v), inqu[v] = 1;
      } } }
      if(mom[t] == -1) break ;
      int df = INFf;
      for(int u = t; u != s; u = mom[u])
        df = min(df, g[mom[u]][id[u]].cap);
      for(int u = t; u != s; u = mom[u]){
        Edge &e = g[mom[u]][id[u]];
        e.cap             -= df;
        g[e.v][e.rev].cap += df;
      }
      mxf += df;
      mnc += df*d[t];
    }
    return {mxf,mnc};
} }flow;
```

9 小技巧

9.1 快讀/快寫

```
inline int read(){
    int x=0,f=1;
    char ch=getchar();
    while(ch<'0'||ch>'9'){
        if(ch=='-') f=-1;
        ch=getchar();
    }
    while(ch>='0' && ch<='9') x=x*10+ch-'0',ch=getchar
     ();
    return x*f;
}
void write(int x){
    if(x<0) putchar('-'),x=-x;
    if(x>9) write(x/10);
    putchar(x%10+'0');
    return;
}
```

9.2 隨機數

```
#include<iostream>
#include<random>
using namespace std;
signed main() {
    mt19937 mt(hash<string>(":poop:"));
    for(int i=1;i<=5;i++) cout<<mt()<<" \n"[i==5];
    return 0;
}
```

9.3 Windows 對拍

```
g++ ac.cpp -o ac
g++ wa.cpp -o wa
$i = 0
while ($true) {
    Write-Output "$i"
    python gen.py > input
    Get-Content input | .\ac.exe > ac.out
    Get-Content input | .\wa.exe > wa.out
    $acOut = Get-Content .\ac.out
    $waOut = Get-Content .\wa.out
    if (diff $acOut $waOut) {
        diff $acOut $waOut
        break
    }
    $i++
}
```

10 其他

```
               (__)    (__) ____/
            /| (oo) _  (oo)/----/_____    *
  _o\______/_|\_\/_/_|__\/|____|//////== *- *  * -
 /__________  \   AC |  AC | NO BUG /== -*
[_____/^^\______\____|_____/^^\______]    *- * -
      \__/                 \__/       Chong~Chong~Chong~
```

```

/ ￣￣￣＼
|  ー  ー \
|   O  O   |   / ￣￣￣￣￣￣￣＼
\     ^   /  <     我好想要AC    \
  ＼     \      \                /
  ／      ＼     \______________/
/  |      \ \
|  |       | |
```