```cpp
#include <cstring>
const int MAXN = 1e5+5;
bool is_prime[MAXN+1];
int prime[MAXN],prime_cnt=0;
//欧拉筛 O(n) prime[0]=2;
void e_getlist()
{
    memset(is_prime,1,sizeof(is_prime));
    is_prime[1]=0;
    for(int i=2;i<=MAXN;i++)
    {
        if(is_prime[i])
        {
            prime[prime_cnt++]=i;
        }
        for(int j=0;j<prime_cnt&&i*prime[j]<=MAXN;j++)
        {
            is_prime[i*prime[j]]=0;
            if(i%prime[j]==0)break;
        }
    }
}
//O(logn)求欧拉函数
int get_phi_logn(int n)
{
//    e_getlist();

    for(int i=0;prime[i]<=n;i++)
        if(n%prime[i]==0){
            n=n/prime[i]*(prime[i]-1);
        }
    return n;
}

//埃拉托尼斯特尼筛 O(n*loglogn)
void a_getlist()
{
    memset(is_prime,1,sizeof(is_prime));
    is_prime[1]=0;
    for(int i=2;i<=MAXN;i++)
    {
        if(is_prime[i])
            for(int j=i+i;j<=MAXN;j+=i)
                is_prime[j]=0;
    }
}

//O(n^(1/2))求欧拉函数
int get_phi(int n){
    int ans=n;
```

```
51      for(int i=2;i*i<=n;i++){
52          if(n%i==0){
53              ans=ans/i*(i-1);
54              while(n%i==0) n/=i;
55          }
56      }
57      if(n>1) ans = ans/n*(n-1);
58      return ans;
59  }
60
61
62
63  int P;
64  int qp(int a,int x)
65  {
66      if(x==1)
67          return a%P;
68      else if(x==0)
69          return 1;
70      else{
71          int re=1;
72          if(x%2)
73              re=a;
74          int y=qp(a,x/2);
75          return y*y%P*re%P;
76      }
77  }
78
```

```cpp

int Discrete_cnt;
int Discrete[MAXN];
void Discrete_initial(int a[],int n)
{
    for(int i=1;i<=n;i++)
        Discrete[i]=a[i];
    sort(Discrete+1,Discrete+1+n);
    Discrete_cnt = unique(Discrete+1,Discrete+1+n)-Discrete-1;
}
int get_Discrete(int x)
{
    return lower_bound(Discrete+1,Discrete+1+Discrete_cnt,x)-Discrete;
}


```

```cpp
#include <iostream>
#include <cstdio>
#include <cmath>
#include <cstring>
#include <vector>
#include <queue>
#include <map>
#include <algorithm>
#include <set>
#include <bitset>
#include <complex>
#include <stack>
#include <fstream>
#include <random>
#include <cstdlib>
#include <assert.h>
#include <time.h>

#define SET0(X) memset(X,0,sizeof(X));
#define SET_1(X) memset(X,-1,sizeof(X));
#define RAND(a, b) ((rand() % ((b)-(a)+1))+ (a))
#define rand() (rand()/double(RAND_MAX))
using namespace std;
typedef long long ll;
typedef unsigned long long ull;
const int MAXN = 5e5 + 4;
const double PI = acos(-1.0);
const double EPS = 1e-6;
const int INF = 0x3f3f3f3f;
const ll MOD = 1e9 + 7;

int main() {

    return 0;
}
```

```cpp
#include <string>
#include <cstring>
#include <cstdio>
using namespace std;

const int maxn = 400;

struct bign{
    int d[maxn], len;

    void clean() { while(len > 1 && !d[len-1]) len--; }

    bign()             { memset(d, 0, sizeof(d)); len = 1; }
    bign(int num)     { *this = num; }
    bign(char* num) { *this = num; }
    bign operator = (const char* num){
        memset(d, 0, sizeof(d)); len = strlen(num);
        for(int i = 0; i < len; i++) d[i] = num[len-1-i] - '0';
        clean();
        return *this;
    }
    bign operator = (int num){
        char s[20]; sprintf(s, "%d", num);
        *this = s;
        return *this;
    }

    bign operator + (const bign& b){
        bign c = *this; int i;
        for (i = 0; i < b.len; i++){
            c.d[i] += b.d[i];
            if (c.d[i] > 9) c.d[i]%=10, c.d[i+1]++;
        }
        while (c.d[i] > 9) c.d[i++]%=10, c.d[i]++;
        c.len = max(len, b.len);
        if (c.d[i] && c.len <= i) c.len = i+1;
        return c;
    }
    bign operator - (const bign& b){
        bign c = *this; int i;
        for (i = 0; i < b.len; i++){
            c.d[i] -= b.d[i];
            if (c.d[i] < 0) c.d[i]+=10, c.d[i+1]--;
        }
        while (c.d[i] < 0) c.d[i++]+=10, c.d[i]--;
        c.clean();
        return c;
    }
```

```
49      bign operator * (const bign& b)const{
50          int i, j; bign c; c.len = len + b.len;
51          for(j = 0; j < b.len; j++) for(i = 0; i < len; i++)
52              c.d[i+j] += d[i] * b.d[j];
53          for(i = 0; i < c.len-1; i++)
54              c.d[i+1] += c.d[i]/10, c.d[i] %= 10;
55          c.clean();
56          return c;
57      }
58      bign operator / (const bign& b){
59          int i, j;
60          bign c = *this, a = 0;
61          for (i = len - 1; i >= 0; i--)
62          {
63              a = a*10 + d[i];
64              for (j = 0; j < 10; j++) if (a < b*(j+1)) break
    ;
65              c.d[i] = j;
66              a = a - b*j;
67          }
68          c.clean();
69          return c;
70      }
71      bign operator % (const bign& b){
72          int i, j;
73          bign a = 0;
74          for (i = len - 1; i >= 0; i--)
75          {
76              a = a*10 + d[i];
77              for (j = 0; j < 10; j++) if (a < b*(j+1)) break
    ;
78              a = a - b*j;
79          }
80          return a;
81      }
82      bign operator += (const bign& b){
83          *this = *this + b;
84          return *this;
85      }
86
87      bool operator <(const bign& b) const{
88          if(len != b.len) return len < b.len;
89          for(int i = len-1; i >= 0; i--)
90              if(d[i] != b.d[i]) return d[i] < b.d[i];
91          return false;
92      }
93      bool operator >(const bign& b) const{return b < *this;}
94      bool operator<=(const bign& b) const{return !(b < *this
    );}
95      bool operator>=(const bign& b) const{return !(*this < b
```

```
 95 );}
 96     bool operator!=(const bign& b) const{return b < *this
    || *this < b;}
 97     bool operator==(const bign& b) const{return !(b < *
    this) && !(b > *this);}
 98
 99     string str() const{
100         char s[maxn]={};
101         for(int i = 0; i < len; i++) s[len-1-i] = d[i]+'0'
    ;
102         return s;
103     }
104 };
105
106 istream& operator >> (istream& in, bign& x)
107 {
108     string s;
109     in >> s;
110     x = s.c_str();
111     return in;
112 }
113
114 ostream& operator << (ostream& out, const bign& x)
115 {
116     out << x.str();
117     return out;
118 }
119
120
121 //------------------------暴力组合数学--------------------
122 const int MAXF = 205;
123 bign fact[MAXF];
124 void get_fact()
125 {
126     fact[0]=1;
127     fact[1]=1;
128     for(int i=2;i<MAXF;i++)
129         fact[i]=fact[i-1]*bign(i);
130 }
131 bign get_Cm(int n,int m)
132 {
133     return fact[n]/fact[m]/fact[n-m];
134 }
135 bign get_Pm(int n,int m)
136 {
137     return fact[n]/fact[n-m];
138 }
139
```

```cpp
//快速傅立叶变换
//------------------手写Complex-----------
#include <cmath>
const int MAXN = 5e5+4;

const double PI = acos(-1.0);
struct complex
{
    double r,i;
    complex(double _r = 0,double _i = 0)
    {
        r = _r; i = _i;
    }
    complex operator +(const complex &b)
    {
        return complex(r+b.r,i+b.i);
    }
    complex operator -(const complex &b)
    {
        return complex(r-b.r,i-b.i);
    }
    complex operator *(const complex &b)
    {
        return complex(r*b.r-i*b.i,r*b.i+i*b.r);
    }
};
void change(complex y[],int len)
{
    int i,j,k;
    for(i = 1, j = len/2;i < len-1;i++)
    {
        if(i < j)swap(y[i],y[j]);
        k = len/2;
        while( j >= k)
        {
            j -= k;
            k /= 2;
        }
        if(j < k)j += k;
    }
}
void fft(complex y[],int len,int on)
{
    change(y,len);
    for(int h = 2;h <= len;h <<= 1)
    {
        complex wn(cos(-on*2*PI/h),sin(-on*2*PI/h));
        for(int j = 0;j < len;j += h)
        {
            complex w(1,0);
```

```
51              for(int k = j;k < j+h/2;k++)
52              {
53                  complex u = y[k];
54                  complex t = w*y[k+h/2];
55                  y[k] = u+t;
56                  y[k+h/2] = u-t;
57                  w = w*wn;
58              }
59          }
60      }
61      if(on == -1)
62          for(int i = 0;i < len;i++)
63              y[i].r /= len;
64  }
65  //-----------------FWT----------------inv=1 正转换 inv=-
    1反转换
66
67
68  void FWT(long long a[],int len,int inv){
69      for(int d=1;d<len;d<<=1){
70          for(int m=d<<1,i=0;i<len;i+=m){
71              for(int j=0;j<d;j++){
72                  long long x = a[i+j],y=a[i+j+d];
73                  a[i+j]=x+y;
74                  a[i+j+d]=x-y;
75                  if(inv<0){
76                      a[i+j]>>=1;
77                      a[i+j+d]>>=1;
78                  }
79              }
80          }
81      }
82  }
83
```

```cpp

struct KM{
    int love[MAXN][MAXN];   // 记录每个妹子和每个男生的好感度
    int ex_girl[MAXN];      // 每个妹子的期望值
    int ex_boy[MAXN];       // 每个男生的期望值
    bool vis_girl[MAXN];    // 记录每一轮匹配匹配过的女生
    bool vis_boy[MAXN];     // 记录每一轮匹配匹配过的男生
    int match[MAXN];        // 记录每个男生匹配到的妹子 如果没有则为-1
    int slack[MAXN];        // 记录每个汉子如果能被妹子倾心最少还需要多少期望值

    bool dfs(int girl)
    {
        vis_girl[girl] = true;

        for (int boy = 0; boy < n; ++boy) {

            if (vis_boy[boy]) continue; // 每一轮匹配 每个男生只尝试一次

            int gap = ex_girl[girl] + ex_boy[boy] - love[girl][boy];

            if (gap == 0) {  // 如果符合要求
                vis_boy[boy] = true;
                if (match[boy] == -1 || dfs( match[boy] )) {    // 找到一个没有匹配的男生 或者该男生的妹子可以找到其他人
                    match[boy] = girl;
                    return true;
                }
            } else {
                slack[boy] = min(slack[boy], gap);  // slack 可以理解为该男生要得到女生的倾心 还需多少期望值 取最小值 备胎的样子【捂脸
            }
        }

        return false;
    }

    int solve()
    {
        memset(match, -1, sizeof match);    // 初始每个男生都没有匹配的女生
        memset(ex_boy, 0, sizeof ex_boy);   // 初始每个男生的期望值为0

        // 每个女生的初始期望值是与她相连的男生最大的好感度
        for (int i = 0; i < n; ++i) {
```

```
42              ex_girl[i] = love[i][0];
43              for (int j = 1; j < n; ++j) {
44                  ex_girl[i] = max(ex_girl[i], love[i][j]);
45              }
46          }
47
48          // 尝试为每一个女生解决归宿问题
49          for (int i = 0; i < n; ++i) {
50
51              fill(slack, slack + n, INF);    // 因为要取最小值 初始化为无穷大
52
53              while (1) {
54                  // 为每个女生解决归宿问题的方法是 ：如果找不到就降低期望值，直到找到为止
55
56                  // 记录每轮匹配中男生女生是否被尝试匹配过
57                  memset(vis_girl, false, sizeof vis_girl);
58                  memset(vis_boy, false, sizeof vis_boy);
59
60                  if (dfs(i)) break;  // 找到归宿 退出
61
62                  // 如果不能找到 就降低期望值
63                  // 最小可降低的期望值
64                  int d = INF;
65                  for (int j = 0; j < n; ++j)
66                      if (!vis_boy[j]) d = min(d, slack[j]);
67
68                  for (int j = 0; j < n; ++j) {
69                      // 所有访问过的女生降低期望值
70                      if (vis_girl[j]) ex_girl[j] -= d;
71
72                      // 所有访问过的男生增加期望值
73                      if (vis_boy[j]) ex_boy[j] += d;
74                      // 没有访问过的boy 因为girl们的期望值降低，距离得到女生倾心又进了一步！
75                      else slack[j] -= d;
76                  }
77              }
78          }
79
80          // 匹配完成 求出所有配对的好感度的和
81          int res = 0;
82          for (int i = 0; i < n; ++i)
83              res += love[ match[i] ][i];
84
85          return res;
86      }
87  }km;
88
```

```cpp

#include <cstring>

struct _edge{
    int v,nex;
};
struct graph{
    _edge edges[MAXE];
    int head[MAXN];
    int edge_cnt;
    void initial()
    {
        edge_cnt=0;
        memset(head,0,sizeof(head));
        edge_cnt=0;
    }
    void add(int u,int v)
    {
        edge_cnt++;
        edges[edge_cnt].v=v;
        edges[edge_cnt].nex=head[u];
        head[u]=edge_cnt;
    }
}g;


int dfn[MAXN]={0};
int low[MAXN]={0};
int total_dfs=0;
int parent[MAXN]={0};

int is_av[MAXN]={0};
int av_cnt=0;

void articulation_dfs(int now)
{
    total_dfs++;
    dfn[now]=low[now]=total_dfs;
    int son_num=0;
    for(int k=g.head[now];k;k=g.edges[k].nex)
    {
        int v=g.edges[k].v;
        if(!dfn[v]){
            son_num++;
            parent[v]=now;

            articulation_dfs(v);
            if(parent[now]!=-1&&low[v]>=dfn[now]){
                is_av[now]=1;
                av_cnt++;
```

```
 51             }
 52             if(parent[now]==-1&&son_num>1){
 53                 is_av[now]=1;
 54                 av_cnt++;
 55             }
 56             low[now]=min(low[now],low[v]);
 57         }
 58         else if(v!=parent[now]){
 59             low[now]=min(low[now],dfn[v]);
 60         }
 61     }
 62 }
 63 void at_solve(int n)
 64 {
 65     SET0(dfn);
 66     SET0(low);
 67     SET0(parent);
 68     SET0(is_av);
 69     total_dfs=0;
 70     av_cnt=0;
 71
 72     parent[1]=-1;
 73     articulation_dfs(1);
 74 }
 75
 76
 77 bool vis[MAXN];
 78 int at_belong[MAXN];
 79 int p_size[MAXN];
 80 int at_cnt;
 81 void build_dfs(int now)
 82 {
 83     vis[now]=1;
 84     at_belong[now]=at_cnt;
 85     p_size[at_cnt]++;
 86     for(int k=g.head[now];k;k=g.edges[k].nex){
 87         int v=g.edges[k].v;
 88         if(!vis[v]&&!is_av[v])
 89             build_dfs(v);
 90     }
 91 }
 92
 93 set<int> inset[MAXN];
 94 int outde[MAXN],inde[MAXN];
 95 graph at_g;
 96 void build_arti_graph(int n)
 97 {
 98     at_g.initial();
 99     SET0(p_size);
100     SET0(vis);
```

```
101     SET0(at_belong);
102     SET0(outde);
103     SET0(inde);
104     for(int i=1;i<=n;i++)
105         inset[i].clear();
106     at_cnt=0;
107     for(int i=1;i<=n;i++)
108     {
109         if(!vis[i])
110         {
111             at_cnt++;
112             if(!is_av[i]){
113                 build_dfs(i);
114             }
115             else{
116                 at_belong[i]=at_cnt;
117                 p_size[at_cnt]++;
118             }
119         }
120     }
121
122     for(int i=1;i<=n;i++){
123         //        printf("%d %d\n",i,at_belong[i]);
124         for(int k=g.head[i];k;k=g.edges[k].nex){
125             int v=g.edges[k].v;
126             if(at_belong[i]!=at_belong[v]&&!inset[
    at_belong[i]].count(at_belong[v])){
127                 at_g.add(at_belong[i],at_belong[v]);
128                 outde[at_belong[i]]++;
129                 inde[at_belong[v]]++;
130                 inset[at_belong[i]].insert(at_belong[v]);
131             }
132         }
133     }
134 }
135 void test_at_graph()
136 {
137     for(int i=1;i<=at_cnt;i++){
138         printf("%d->(no:%d size:%d)->%d:",inde[i],i,p_size
    [i],outde[i]);
139         for(int k=at_g.head[i];k;k=at_g.edges[k].nex)
140         {
141             printf(" %d",at_g.edges[k].v);
142         }
143         printf("\n");
144     }
145 }
146
147
```

```cpp
//Dinic
const int INF = 0x5fffffff;
const int MAXN = 4e2+5;

struct adj_graph{
    int n;
    int adj[MAXN][MAXN];
    inline void update(int u,int v,int a){
        adj[u][v]+=a;
    }
    void initial()
    {
        SET0(adj);
    }
    adj_graph(){
        initial();
    }


    void debug()
    {
        static int dbnum=1;
        printf("dbnum:%d\n",dbnum++);
        for(int i=1;i<=n;i++)
        {
            for(int j=1;j<=n;j++)
            {
                if(adj[i][j])
                {
                    printf("%d-->%d  w:%d\n",i,j,adj[i][j])
;
                }
            }
        }
    }
}g;


int level[MAXN];
bool get_level(int start,int terminal)
{
    SET0(level);
    queue<int> q;
    q.push(start);
    level[start]=1;

    while(q.size()) {
        int now=q.front();
        q.pop();
        for(int v=1;v<=g.n;v++)
```

```
50          {
51              if(g.adj[now][v]>0){
52                  if(!level[v]){
53                      level[v]=level[now]+1;
54                      if(v==terminal)
55                          return true;
56                      q.push(v);
57                  }
58              }
59          }
60      }
61      return false;
62  }
63
64  int augment_cur_dfs(int now,int terminal,int low_cap)
65  {
66      if(now==terminal){
67          return low_cap;
68      }
69      for(int v=1;v<=g.n;v++)
70      {
71          if(g.adj[now][v]>0
72           &&level[v]==level[now]+1)
73          {
74              int re=augment_cur_dfs(v,terminal,min(low_cap,g
    .adj[now][v]));
75
76              if(re)
77              {
78                  g.update(now,v,-re);
79                  g.update(v,now,re);
80                  return re;
81              }
82          }
83      }
84      return 0;
85  }
86
87  int dinic(int start,int terminal)
88  {
89      int re=0;
90      while (get_level(start,terminal)) {
91  //          g.debug();
92          re+=augment_cur_dfs(start,terminal,INF);
93      }
94      return re;
95  }
96
97  //--------------------结构体版本------------------
98
```

```
 99 const int INF = 0x3f3f3f3f;
100 const int MAXN = 5e4+5;
101 const int MAXE = 2e6+5;
102 struct _edge{
103     int u,v,nex;
104     int cap;
105 };
106 struct adj_graph{
107     _edge edges[MAXE];
108     int head[MAXN];
109     int edge_cnt;
110     void initial()
111     {
112         edge_cnt=0;
113         memset(head,-1,sizeof(head));
114     }
115     void add(int u,int v,int cap)
116     {
117         edges[edge_cnt].v=v;
118         edges[edge_cnt].u=u;
119         edges[edge_cnt].nex=head[u];
120         edges[edge_cnt].cap=cap;
121         head[u]=edge_cnt;
122         edge_cnt++;
123 
124         edges[edge_cnt].v=u;
125         edges[edge_cnt].u=v;
126         edges[edge_cnt].nex=head[v];
127         edges[edge_cnt].cap=0;
128         head[v]=edge_cnt;
129         edge_cnt++;
130     }
131     adj_graph(){initial();}
132     void debug(int n)
133     {
134         for(int now=0;now<=n;now++)
135         {
136             for(int k=head[now];k!=-1;k=edges[k].nex)
137             {
138                 int v=edges[k].v;
139                 printf("%d-->%d  %d\n",now,v,edges[k].cap)
    ;
140             }
141         }
142     }
143 };
144 
145 struct dinic{
146     adj_graph g;
147     int level[MAXN];
```

```
148     bool get_level(int start,int terminal)
149     {
150         SET0(level);
151         queue<int>q;
152         q.push(start);
153         level[start]=1;
154         while(q.size()){
155             int now=q.front();
156             q.pop();
157 
158             for(int k=g.head[now];k!=-1;k=g.edges[k].nex){
159                 int v=g.edges[k].v;
160                 if(g.edges[k].cap>0&&!level[v]){
161                     level[v]=level[now]+1;
162                     if(v==terminal)
163                         return true;
164                     q.push(v);
165                 }
166             }
167         }
168         return false;
169     }
170     int augment_cur_dfs(int now,int terminal,int max_flow)
171     {
172         if(now==terminal)
173             return max_flow;
174         int re=0;
175         for(int k=g.head[now];k!=-1;k=g.edges[k].nex)
176         {
177             int v=g.edges[k].v;
178             if(level[v]==level[now]+1)
179             {
180                 int f=augment_cur_dfs(v,terminal,min(
    max_flow-re,g.edges[k].cap));
181                 g.edges[k].cap-=f;
182                 g.edges[k^1].cap+=f;
183                 re+=f;
184                 if(re==max_flow)return re;
185             }
186         }
187         return re;
188     }
189 
190     int solve(int start,int terminal){
191         int maxflow=0;
192         while(get_level(start,terminal)){
193             maxflow+=augment_cur_dfs(start,terminal,INF);
194         }
195         return maxflow;
196     }
```

```
197 }din;
198
199
```

```cpp
const int MAXN = 1e5+4;
const int MAXE = 2e5+4;
struct graph {
    int head[MAXN];
    int cnt=0;
    struct edge {
        int v,nex;
        int w;
    }edges[MAXE];
    void initial(){
        SET_1(head);
        cnt=0;
    }
    graph(){initial();}

    void add(int u,int v,int w){
        edges[cnt].v=v;
        edges[cnt].w=w;
        edges[cnt].nex=head[u];
        head[u]=cnt;
        cnt++;
    }

    void debug(int n){
        for(int i=1;i<=n;i++){
            for(int k=head[i];k!=-1;k=edges[k].nex){
                int v=edges[k].v;
                int w=edges[k].w;
                printf("%d -> %d with %d\n",i,v,w);
            }
        }
    }

    void debuge(){
        for(int i=0;i<cnt;i++){
            printf("x -> %d with %d\n",edges[i].v,edges[i].
w);
        }
    }
}g;
```

```cpp
//Tarjan
#include <cstring>
#include<stack>
const int MAXN = 5e5+5;
const int MAXE = 5e5+5;
struct _edge{
    int v,nex;
};
struct graph{
    _edge edges[MAXE];
    int head[MAXN];
    int edge_cnt;
    void initial()
    {
        edge_cnt=0;
        memset(head,0,sizeof(head));
        edge_cnt=0;
    }
    void add(int u,int v)
    {
        edge_cnt++;
        edges[edge_cnt].v=v;
        edges[edge_cnt].nex=head[u];
        head[u]=edge_cnt;
    }
}g;

int dfn[MAXN]={0};
int low[MAXN]={0};
stack<int> st;
bool inst[MAXN]={0};
int belong[MAXN]={0};
int taj_cnt=0;
int total=0;

void tarjan_dfs(int x)
{
    total++;
    dfn[x]=total;
    low[x]=total;
    inst[x]=1;
    st.push(x);

    for(int k=g.head[x];k;k=g.edges[k].nex)
    {
        int v=g.edges[k].v;
        if(dfn[v]==0)
        {
            tarjan_dfs(v);
            low[x]=min(low[x],low[v]);
```

```
 51          }
 52          else if(inst[v])
 53          {
 54              low[x]=min(low[x],dfn[v]);
 55          }
 56      }
 57      if(low[x]==dfn[x])
 58      {
 59          int u;
 60          ++taj_cnt;
 61          while(1)
 62          {
 63              u=st.top();
 64              st.pop();
 65              inst[u]=0;
 66              belong[u]=taj_cnt;
 67
 68              //              cout<<u<<" ";
 69              if(u==x)
 70                  break;
 71          }
 72          //        cout<<endl;
 73      }
 74 }
 75 void tarjan(int n)
 76 {
 77      total=0;taj_cnt=0;
 78      memset(dfn,0,sizeof(dfn));
 79      memset(inst,0,sizeof(inst));
 80      memset(low,0,sizeof(low));
 81      while(!st.empty()) st.pop();
 82      for(int i=1;i<=n;i++)
 83          if(dfn[i]==0)tarjan_dfs(i);
 84 }
 85
 86 graph ta;
 87 set<int> inset[MAXN];
 88 void build_taj_graph(int n)
 89 {
 90      ta.initial();
 91      for(int i=1;i<=n;i++)
 92      {
 93          for(int k=g.head[i];k;k=g.edges[k].nex)
 94          {
 95              int v=g.edges[k].v;
 96              if(belong[i]!=belong[v]&&!inset[i].count(v))
 97              {
 98                  ta.add(belong[i],belong[v]);
 99                  inset[belong[i]].insert(belong[v]);
100              }
```

```
101         }
102     }
103 }
104
105
106 int outde[MAXN],inde[MAXN];
107 graph ta;
108 set<int> inset[MAXN];
109 void build_taj_graph(int n)
110 {
111     memset(outde,0,sizeof(outde));
112     memset(inde,0,sizeof(inde));
113     for(int i=1;i<=taj_cnt;i++)
114         inset[i].clear();
115
116     ta.initial();
117
118
119     for(int i=1;i<=n;i++)
120     {
121         for(int k=g.head[i];k;k=g.edges[k].nex)
122         {
123             int v=g.edges[k].v;
124             if(belong[i]!=belong[v]&&!inset[belong[i]].
    count(belong[v]))
125             {
126                 ta.add(belong[i],belong[v]);
127                 outde[belong[i]]++;
128                 inde[belong[v]]++;
129
130                 inset[belong[i]].insert(belong[v]);
131             }
132         }
133     }
134 }
135 void test_taj_graph()
136 {
137     for(int i=1;i<=taj_cnt;i++){
138         printf("%d->(no:%d size:%d)->%d:",inde[i],i,p_size
    [i],outde[i]);
139         for(int k=ta.head[i];k;k=ta.edges[k].nex)
140         {
141             printf(" %d",ta.edges[k].v);
142         }
143         printf("\n");
144     }
145 }
146
```

```cpp
//MinCostMaxFlow
//----------------------这个板子身经百战----------------------
const int INF = 0x3f3f3f3f;
const int MAXN = 2e4;
const int MAXE = 2e5+5;
struct _edge{
    int u,v,nex;
    int cap,cost;
};
struct adj_graph{
    _edge edges[MAXE];
    int head[MAXN];
    int edge_cnt;
    void initial()
    {
        edge_cnt=0;
        memset(head,-1,sizeof(head));
    }
    void add(int u,int v,int cap,int cost)
    {
        edges[edge_cnt].v=v;
        edges[edge_cnt].u=u;
        edges[edge_cnt].nex=head[u];
        edges[edge_cnt].cap=cap;
        edges[edge_cnt].cost=cost;
        head[u]=edge_cnt;
        edge_cnt++;

        edges[edge_cnt].v=u;
        edges[edge_cnt].u=v;
        edges[edge_cnt].nex=head[v];
        edges[edge_cnt].cap=0;
        edges[edge_cnt].cost=-cost;
        head[v]=edge_cnt;
        edge_cnt++;

    }
    adj_graph(){initial();}
    void debug(int n)
    {
        for(int now=0;now<=n;now++)
        {
            for(int k=head[now];k!=-1;k=edges[k].nex)
            {
                int v=edges[k].v;
                printf("%d-->%d  %d %d\n",now,v,edges[k].
cap,edges[k].cost);
            }
        }
    }
```

```
50 };
51 struct min_cost_max_flow{
52     adj_graph g;
53     int dis[MAXN];
54     int pre[MAXN];
55     bool inq[MAXN];
56     queue<int> q;
57     bool spfa(int start,int terminal)
58     {
59         while (q.size())
60             q.pop();
61         for(int i=0;i<MAXN;i++)
62             dis[i]=INF;
63         SET_1(pre);
64         SET0(inq);
65 
66         q.push(start);
67         inq[start]=1;
68         dis[start]=0;
69         while (q.size()) {
70             int u=q.front();
71             q.pop();
72             inq[u]=0;
73             for(int k=g.head[u];k!=-1;k=g.edges[k].nex)
74             {
75                 int v=g.edges[k].v;
76                 if(g.edges[k].cap&&dis[u]+g.edges[k].cost<
   dis[v])
77                 {
78                     dis[v]=dis[u]+g.edges[k].cost;
79                     pre[v]=k;
80                     if(!inq[v])
81                     {
82                         q.push(v);
83                         inq[v]=1;
84                     }
85                 }
86             }
87         }
88         return dis[terminal]<INF;
89     }
90     int solve(int start,int terminal,int &maxflow)
91     {
92         //            static int no=0;
93         int mincost=0;
94         maxflow=0;
95         while (spfa(start,terminal)) {
96 
97             //                printf("no:%d",no++);
98             int flow=INF;
```

```
 99              for(int k=pre[terminal];k!=-1;k=pre[g.edges[k]
    .u])
100                  flow=min(flow,g.edges[k].cap);
101              for(int k=pre[terminal];k!=-1;k=pre[g.edges[k]
    .u])
102              {
103                  g.edges[k].cap -= flow;
104                  g.edges[k^1].cap += flow;
105                  mincost += g.edges[k].cost * flow;
106              }
107              maxflow+=flow;
108
109              //            printf(" %d %d\n",mincost,
    maxflow);
110          }
111          return mincost;
112      }
113 }mcmf;
114
```

```cpp
const int MAXN = 5e4+4;
char s[MAXN],t[MAXN];
int nex[MAXN];
//kmp 匹配
void get_nex(const char *pat,int next[])
{
    int p,patlen=strlen(pat);
    next[0]=-1;
    for(int i=1;i<patlen;i++)
    {
        p=next[i-1];
        while (p!=-1&&pat[p+1]!=pat[i])
            p=next[p];
        if(p==-1)
            next[i]=pat[0]==pat[i]?0:-1;
        else
            next[i]=p+1;
    }
}

vector<int> ans;
void KMP_match(char *s,char *t)
{
    get_nex(t,nex);
    int ps=0,pt=0,tlen=strlen(t);
    while(s[ps]) {
        if(s[ps]==t[pt]){
            ps++;pt++;
        }
        else{
            if(pt==0)
                ps++;
            else
                pt=nex[pt-1]+1;
        }
        if(pt==tlen){
            ans.push_back(ps);
            pt=0;
        }
    }
}

//拓展KMP
void ext_get_next(const char* pat,int next[])
{
    int patlen=strlen(pat);
    int a,p;
    next[0]=patlen;
    for(int i=1,j=-1;i<patlen;i++,j--)
    {
```

```
51          if(j<0||i+next[i-a]>=p){
52              if(j<0){
53                  p=i;
54                  j=0;
55              }
56              while (p<patlen&&pat[p]==pat[j]) {
57                  p++;
58                  j++;
59              }
60              next[i]=j;
61              a=i;
62          }
63          else
64              next[i]=next[i-a];
65      }
66  }
67
```

```cpp
const int MAXN = 1e5+4;

const int N = 26 ;

struct Palindromic_Tree {
    int next[MAXN][N] ;//next指针，next指针和字典树类似，指向的串为当前串两端加上同一个字符构成
    int fail[MAXN] ;//fail指针，失配后跳转到fail指针指向的节点
    int cnt[MAXN] ; //表示节点i表示的本质不同的串的个数（建树时求出的不是完全的，最后count()函数跑一遍以后才是正确的）
    int num[MAXN] ; //表示以节点i表示的最长回文串的最右端点为回文串结尾的回文串个数
    int len[MAXN] ;//len[i]表示节点i表示的回文串的长度（一个节点表示一个回文串）
    int S[MAXN] ;//存放添加的字符
    int last ;//指向新添加一个字母后所形成的最长回文串表示的节点。
    int n ;//表示添加的字符个数。
    int p ;//表示添加的节点个数。

    int newnode ( int l ) {//新建节点
        for ( int i = 0 ; i < N ; ++ i ) next[p][i] = 0 ;
        cnt[p] = 0 ;
        num[p] = 0 ;
        len[p] = l ;
        return p ++ ;
    }

    void init () {//初始化
        p = 0 ;
        newnode (  0 ) ;
        newnode ( -1 ) ;
        last = 0 ;
        n = 0 ;
        S[n] = -1 ;//开头放一个字符集中没有的字符，减少特判
        fail[0] = 1 ;
    }

    int get_fail ( int x ) {//和KMP一样，失配后找一个尽量最长的
        while ( S[n - len[x] - 1] != S[n] ) x = fail[x] ;
        return x ;
    }

    void add ( int c ) {
        c -= 'a' ;
        S[++ n] = c ;
        int cur = get_fail ( last ) ;//通过上一个回文串找这个回文串的匹配位置
        if ( !next[cur][c] ) {//如果这个回文串没有出现过，说明出现了一个新的本质不同的回文串
            int now = newnode ( len[cur] + 2 ) ;//新建节点
```

```
45              fail[now] = next[get_fail ( fail[cur] )][c] ;//
    和AC自动机一样建立fail指针，以便失配后跳转
46              next[cur][c] = now ;
47              num[now] = num[fail[now]] + 1 ;
48          }
49          last = next[cur][c] ;
50          cnt[last] ++ ;
51      }
52
53      void count () {
54          for ( int i = p - 1 ; i >= 0 ; -- i ) cnt[fail[i]]
    += cnt[i] ;
55          //父亲累加儿子的cnt，因为如果fail[v]=u，
    则u一定是v的子回文串!
56      }
57  } pt;
58
59  //双向回文树
60  const int CHARSET_SIZE = 26;
61  struct PAM {
62      int ch[MAXN * 2], pos[2];
63      int node_cnt, last[2];
64      int nxt[MAXN][CHARSET_SIZE];
65      int len[MAXN];
66      int fail[MAXN];
67
68      void init() {
69          pos[0] = MAXN;
70          pos[1] = MAXN - 1;
71          ch[MAXN - 1] = ch[MAXN] = -1;
72          node_cnt = 1;
73          len[0] = 0;
74          len[1] = -1;
75          fail[0] = 1;
76          last[0] = last[1] = 0;
77          SET0(nxt);
78      }
79
80      int getFail(int u, int d) {
81          if (d)
82              for (; ch[pos[1] - len[u] - 1] != ch[pos[1]]; u
     = fail[u]);
83          else
84              for (; ch[pos[0] + len[u] + 1] != ch[pos[0]]; u
     = fail[u]);
85          return u;
86      }
87
88      void insert(int c, int d) {
89          if (!d) {
```

```
 90             ch[--pos[0]] = c;
 91             ch[pos[0] - 1] = -1;
 92         } else {
 93             ch[++pos[1]] = c;
 94             ch[pos[1] + 1] = -1;
 95         }
 96         int u = getFail(last[d], d);
 97         if (!nxt[u][c]) {
 98             int v = ++node_cnt;
 99             len[v] = len[u] + 2;
100             fail[v] = nxt[getFail(fail[u], d)][c];
101             nxt[u][c] = v;
102         }
103         u = nxt[u][c];
104         last[d] = u;
105         if (len[u] == pos[1] - pos[0] + 1)
106             last[d ^ 1] = u;
107     }
108 
109     void debug() {
110         printf("0\n");
111         dfs(0, 0);
112         printf("1\n");
113         dfs(1, 0);
114         printf("size:%d\n", node_cnt - 1);
115     }
116 
117     void dfs(int now, int dep) {
118         for (int i = 0; i < 26; i++) {
119             int nxtnow = nxt[now][i];
120             if (nxtnow != 0) {
121                 for (int j = 0; j < dep; j++) {
122                     printf(" ");
123                 }
124                 printf("%c\n", i + 'a');
125                 dfs(nxtnow, dep + 1);
126             }
127         }
128     }
129 }
130 
131 //双向可持久化回文树
132 const int CHARSET_SIZE = 26;
133 
134 struct PAM {
135     int ch[MAXN * 2], pos[2];
136     int node_cnt, last[MAXN][2];
137     int nxt[MAXN][CHARSET_SIZE];
138     int len[MAXN];
139     int fail[MAXN];
```

```
140
141     //可持久化
142     int n;//版本
143     int cc[MAXN];//第i个版本插入的字符
144     int u[MAXN];//第i个版本的父亲
145     int b[MAXN];//第i个版本插入的方向
146
147     inline void init() {
148         n = 0;
149         SET_1(cc);
150
151         pos[0] = MAXN;
152         pos[1] = MAXN - 1;
153         ch[MAXN - 1] = ch[MAXN] = -1;
154         node_cnt = 1;
155         len[0] = 0;
156         len[1] = -1;
157
158         fail[0] = 1;
159         last[0][0] = last[0][1] = 0;
160         SET0(nxt);
161
162     }
163
164     inline int getFail(int u, int d) {
165         if (d)
166             for (; ch[pos[1] - len[u] - 1] != ch[pos[1]];
    u = fail[u]) {
167                 assert(u != fail[u]);
168             }
169         else
170             for (; ch[pos[0] + len[u] + 1] != ch[pos[0]];
    u = fail[u]) {
171                 assert(u != fail[u]);
172             }
173         return u;
174     }
175
176     inline void back() {
177         if (!b[n]) {
178             ch[pos[0]] = -1;
179             ++pos[0];
180         } else {
181             ch[pos[1]] = -1;
182             --pos[1];
183         }
184
185         if (cc[n] != -1) {
186             nxt[u[n]][cc[n]] = 0;
187             u[n] = 0;
```

```
188             fail[node_cnt]=0;
189             cc[n]=-1;
190             node_cnt--;
191         }
192         n--;
193     }
194 
195 
196     inline void insert(int c, int d) {
197         n++;
198         b[n] = d;
199         if (!d) {
200             pos[0]--;
201             ch[pos[0]] = c;
202             ch[pos[0] - 1] = -1;
203         } else {
204             pos[1]++;
205             ch[pos[1]] = c;
206             ch[pos[1] + 1] = -1;
207         }
208         u[n] = getFail(last[n - 1][d], d);
209         if (!nxt[u[n]][c]) {
210             cc[n] = c;
211             int v = ++node_cnt;
212 
213             len[v] = len[u[n]] + 2;
214             fail[v] = nxt[getFail(fail[u[n]], d)][c];
215 
216             nxt[u[n]][c] = v;
217         }
218         int uu = nxt[u[n]][c];
219         last[n][d] = uu;
220         if (len[uu] == pos[1] - pos[0] + 1)
221             last[n][d ^ 1] = uu;
222         else
223             last[n][d ^ 1] = last[n - 1][d ^ 1];
224 
225     }
226 
227     void debug() {
228         printf("---------------\n");
229         printf("0\n");
230         dfs(0, 0);
231         printf("1\n");
232         dfs(1, 0);
233         printf("version:%d last[%d][0]:%d last[%d][1]:%d\n
    ", n, n, last[n][0], n, last[n][1]);
234         printf("pos[%d][0]:%d pos[%d][1]:%d\n", n - 1, pos
    [0], n - 1, pos[1]);
235 
```

```
236         printf("size:%d\n", node_cnt - 1);
237         printf("---------------\n");
238     }
239
240     void dfs(int now, int dep) {
241         printf("fail[%d]=%d len[%d]=%d\n", now, fail[now],
    now, len[now]);
242         for (int i = 0; i < 26; i++) {
243             int nxtnow = nxt[now][i];
244             if (nxtnow != 0) {
245                 for (int j = 0; j < dep + 1; j++) {
246                     printf("-");
247                 }
248                 printf("%c\n", i + 'a');
249                 dfs(nxtnow, dep + 1);
250             }
251         }
252
253     }
254 }  PM;
```

```cpp
/*
*suffix array
*倍增算法 O(n*logn)
*待排序数组长度为 n, 放在 0 n-1 中, 在最后面补一个 0
*da(str ,sa,rk,height, n , );//注意是 n;
*例如:
*n = 8;
* num[] = { 1, 1, 2, 1, 1, 1, 1, 2, $ }; 注意 num 最后一位为 0, 其他
大于 0
*rk[] = 4, 6, 8, 1, 2, 3, 5, 7, 0 ;rk[0 n-1] 为有效值, rk[n] 必定为 0 无效值
*sa[] = 8, 3, 4, 5, 0, 6, 1, 7, 2 ;sa[1 n] 为有效值, sa[0] 必定为 n 是
无效值
*height[]= 0, 0, 3, 2, 3, 1, 2, 0, 1 ;height[2 n] 为有效值
kuangbin
ACM Template of kuangbin
 *
*/
#include <bits/stdc++.h>
using namespace std;
const int MAXN = 20010;
int t1[MAXN], t2[MAXN], c[MAXN]; //求 SA 数组需要的中间变量, 不需要赋值
//待排序的字符串放在 s 数组中, 从 s[0] 到 s[n-1], 长度为 n, 且最大值小于 m,
//除 s[n-1] 外的所有 s[i] 都大于 0, r[n-1]=0
//函数结束以后结果放在 sa 数组中
bool cmp(int *r, int a, int b, int l) {
    return r[a] == r[b] && r[a + l] == r[b + l];
}
void da(int str[], int sa[], int rk[], int height[], int n, int m) {
    str[n]=0;
    n++;
    int i, j, p, *x = t1, *y = t2;
//第一轮基数排序, 如果 s 的最大值很大, 可改为快速排序
    for (i = 0; i < m; i++)c[i] = 0;
    for (i = 0; i < n; i++)c[x[i] = str[i]]++;
    for (i = 1; i < m; i++)c[i] += c[i - 1];
    for (i = n - 1; i >= 0; i--)  sa[--c[x[i]]] = i;
    for (j = 1; j <= n; j <<= 1) {
        p = 0;
//直接利用 sa 数组排序第二关键字
        for (i = n - j; i < n; i++)y[p++] = i; //后面的 j 个数第二关键字为空的最小
        for (i = 0; i < n; i++)if (sa[i] >= j)y[p++] = sa[i] - j;
//这样数组 y 保存的就是按照第二关键字排序的结果
```

```
44 //基数排序第一关键字
45         for (i = 0; i < m; i++)c[i] = 0;
46         for (i = 0; i < n; i++)c[x[y[i]]]++;
47         for (i = 1; i < m; i++)c[i] += c[i - 1];
48         for (i = n - 1; i >= 0; i--)sa[--c[x[y[i]]]] = y[i]
   ;
49 //根据 sa 和 x 数组计算新的 x 数组
50         swap(x, y);
51         p = 1; x[sa[0]] = 0;
52         for (i = 1; i < n; i++)
53             x[sa[i]] = cmp(y, sa[i - 1], sa[i], j) ? p - 1
   : p++;
54         if (p >= n)break;
55         m = p;//下次基数排序的最大值
56     }
57     int k = 0;
58     n--;
59     for (i = 0; i <= n; i++)rk[sa[i]] = i;
60     for (i = 0; i < n; i++) {
61         if (k) k--;
62         j = sa[rk[i] - 1];
63         while (str[i + k] == str[j + k])k++;
64         height[rk[i]] = k;
65     }
66 }
67 int rk[MAXN], height[MAXN];
68 int RMQ[MAXN];
69 int mm[MAXN];
70
71 int best[20][MAXN];
72 void initRMQ(int n) {
73     mm[0] = -1;
74     for (int i = 1; i <= n; i++)
75         mm[i] = ((i & (i - 1)) == 0) ? mm[i - 1] + 1 : mm[i
    - 1];
76     for (int i = 1; i <= n; i++)best[0][i] = i;
77     for (int i = 1; i <= mm[n]; i++)
78         for (int j = 1; j + (1 << i) - 1 <= n; j++) {
79             int a = best[i - 1][j];
80             int b = best[i - 1][j + (1 << (i - 1))];
81             if (RMQ[a] < RMQ[b])best[i][j] = a;
82             else best[i][j] = b;
83         }
84 }
85 int askRMQ(int a, int b) {
86     int t;
87     t = mm[b - a + 1];
88     b -= (1 << t) - 1;
89     a = best[t][a]; b = best[t][b];
90     return RMQ[a] < RMQ[b] ? a : b;
```

```
 91 }
 92 int lcp(int a, int b) {
 93     a = rk[a]; b = rk[b];
 94     if (a > b)swap(a, b);
 95     return height[askRMQ(a + 1, b)];
 96 }
 97 char str[MAXN];
 98 int r[MAXN];
 99 int sa[MAXN];
100
101 /*
102 * 后缀数组
103 * DC3 算法，复杂度 O(n)
104 * 所有的相关数组都要开三倍
105 */
106
107 #define F(x) ((x)/3+((x)%3==1?0:tb))
108 #define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)
109
110 int wa[MAXN*3],wb[MAXN*3],wv[MAXN*3],wss[MAXN*3];
111 int c0(int *r,int a,int b){
112     return r[a] == r[b] && r[a+1] == r[b+1] && r[a+2] == r
    [b+2];
113 }
114 int c12(int k,int *r,int a,int b){
115     if(k == 2)
116         return r[a] < r[b] || ( r[a] == r[b] && c12(1,r,a+
    1,b+1) );
117     else return r[a] < r[b] || ( r[a] == r[b] && wv[a+1] <
     wv[b+1] );
118 }
119 void sort(int *r,int *a,int *b,int n,int m)
120 {
121     int i;
122     for(i=0;i<n;i++)
123         wv[i]=r[a[i]];
124     for(i=0;i<m;i++)
125         wss[i]=0;
126     for(i=0;i<n;i++)
127         wss[wv[i]]++;
128     for(i=1;i<m;i++)
129         wss[i]+=wss[i-1];
130     for(i=n-1;i>=0;i--)
131         b[--wss[wv[i]]]=a[i];
132 }
133 void dc3(int *r,int *sa,int n,int m){
134
135     int i,j,*rn = r+n;
136     int *san=sa+n,ta=0,tb=(n+1)/3,tbc=0,p;
137     r[n]=r[n+1]=0;
```

```
138     for(i=0;i<n;i++)
139         if(i%3!=0)
140             wa[tbc++]=i;
141     sort(r+2,wa,wb,tbc,m);
142     sort(r+1,wb,wa,tbc,m);
143     sort(r,wa,wb,tbc,m);
144     for(p=1,rn[F(wb[0])]=0,i=1;i<tbc;i++)
145         rn[F(wb[i])]=c0(r,wb[i-1],wb[i])?p-1:p++;
146     if(p<tbc)dc3(rn,san,tbc,p);
147     else for(i=0;i<tbc;i++) san[rn[i]]=i;
148     for(i=0;i<tbc;i++) if(san[i]<tb)wb[ta++]=san[i]*3;
149     if(n%3==1) wb[ta++]=n-1;
150     sort(r,wb,wa,ta,m);
151     for(i=0;i<tbc;i++)wv[wb[i]=G(san[i])] =i;
152     for(i=0,j=0,p=0;i<ta&&j<tbc;p++)
153         sa[p]=c12(wb[j]%3,r,wa[i],wb[j])?wa[i++]:wb[j++];
154     for(;i<ta;p++)sa[p]=wa[i++];
155     for(;j<tbc;p++)sa[p]=wb[j++];
156 }
157
158 void da(int str[],int sa[],int rank[],int height[],int n,
    int m){
159     for(int i=n;i<n*3;i++)
160         str[i]=0;
161     dc3(str,sa,n+1,m);
162     int i,j,k=0;
163     for(i=0;i<=n;i++)rank[sa[i]]=i;
164     for(i=0;i<n;i++){
165         if(k)k--;
166         j=sa[rank[i]-1];
167         while(str[i+k]==str[j+k])k++;
168         height[rank[i]]=k;
169     }
170 }
171
172
```

```cpp
const int CHAR = 26;
struct SAM_Node
{
    SAM_Node *fa, *next[CHAR];
    int len;
    int id, pos;
    SAM_Node() {}
    SAM_Node(int _len)
    {
        fa = 0;
        len = _len;
        memset(next, 0, sizeof(next));
    }
};
struct SAM{
    SAM_Node SAM_node[MAXN * 2], *SAM_root, *SAM_last;
    int SAM_size;
    SAM_Node *newSAM_Node(int len)
    {
        SAM_node[SAM_size] = SAM_Node(len);
        SAM_node[SAM_size].id = SAM_size;
        return &SAM_node[SAM_size++];
    }
    SAM_Node *newSAM_Node(SAM_Node *p)
    {
        SAM_node[SAM_size] = *p;
        SAM_node[SAM_size].id = SAM_size;
        return &SAM_node[SAM_size++];
    }
    void SAM_init()
    {
        SAM_size = 0;
        SAM_root = SAM_last = newSAM_Node(0);
        SAM_node[0].pos = 0;
    }
    void SAM_add(int x, int len)
    {
        SAM_Node *p = SAM_last, *np = newSAM_Node(p->len +
1);
        np->pos = len;
        SAM_last = np;
        for (; p && !p->next[x]; p = p->fa)
            p->next[x] = np;
        if (!p)
        {
            np->fa = SAM_root;
            return;
        }
        SAM_Node *q = p->next[x];
        if (q->len == p->len + 1)
```

```
50          {
51              np->fa = q;
52              return;
53          }
54          SAM_Node *nq = newSAM_Node(q);
55          nq->len = p->len + 1;
56          q->fa = nq;
57          np->fa = nq;
58          for (; p && p->next[x] == q; p = p->fa)
59              p->next[x] = nq;
60      }
61      void SAM_build(char *s)
62      {
63          SAM_init();
64          int len = strlen(s);
65          for (int i = 0; i < len; i++)
66              SAM_add(s[i] - 'a', i + 1);
67      }
68  }sam;
69
```

```cpp
int min_max_expression(const char *s,int slen,bool is_min)
{
    int k=0,p[2]={0,1};
    while(p[0]<slen&&p[1]<slen&&k<slen) {
        int t=s[(p[0]+k)%slen]-s[(p[1]+k)%slen];
        if(t==0)
            k++;
        else{
            if(t>0)
                p[1-is_min]+=k+1;
            else
                p[0+is_min]+=k+1;

            if(p[0]==p[1])
                p[1]++;
            k=0;
        }
    }
    return min(p[0],p[1]);
}

```

```cpp
namespace manacher {
    const int MAXN=1e5+5;
    char ns[MAXN * 2];
    //&#1#2#3#3#2#1#
    char *get_ns(const char *s, char *ns) {
        int l = strlen(s);
        ns[0] = '$';
        for (int i = 1; i <= l * 2; i++) {
            ns[i++] = '#';
            ns[i] = s[(i - 1) / 2];
        }
        ns[l * 2 + 1] = '#';
        ns[l * 2 + 2] = 0;
        return ns;
    }

    int Len[MAXN * 2];

    //Len[i] - 1 就是回文串长度
    void manacher(const char *s) {
        int i;
        int mx = 0, id;
        get_ns(s, ns);
        int nslen = strlen(ns);
        memset(Len, 0, sizeof(Len));
        for (i = 1; i < nslen - 1; i++) {
            if (mx > i)
                Len[i] = min(Len[2 * id - i], mx - i);  //2*id-i是i关于id的对称点
            else   //越过mx则暴力判断
                Len[i] = 1;
            while (ns[i + Len[i]] == ns[i - Len[i]])  //回文匹配
                Len[i]++;
            if (Len[i] + i > mx) {    //更新mx和id
                mx = Len[i] + i;
                id = i;
            }
        }
    }

    //判断[L,R]是不是回文串，L和R是原字符串的下标
    bool is_palindrome(int L, int R) {
        return Len[L + R + 2] >= R - L + 2 ? 1 : 0;
    }

    //x 是ns下标，返回原字符串的下标
    const int Lp(int x) { return (x / 2 - Len[x] / 2); }

    const int Rp(int x) { return (x / 2 + (Len[x] - 1) / 2
```

```
48 - 1); }
49 }
```

```cpp
//  ST表

#include <iostream>
#include <algorithm>
#include <cmath>

#define MAXN 400000
using namespace std;
int st[MAXN][20] = {0};
int data[MAXN] = {0};
int lg2[MAXN] = {0};

void initial(int *value, int n) {
    for (int i = 1; i <= n; i++) {
        st[i][0] = value[i];
        lg2[i] = log2(i);
    }
    for (int j = 1; (1 << j) <= n; j++) {
        for (int i = 1; i + (1 << j) - 1 <= n; i++)
            st[i][j] = max(st[i][j - 1], st[i + (1 << (j - 1))][j - 1]);
    }
}

int query(int L, int R) {
    int k = lg2[R - L + 1];
    return max(st[L][k], st[R - (1 << k) + 1][k]);
}

int main(int argc, const char *argv[]) {
    int n;
    cin >> n;
    for (int i = 1; i <= n; i++) {
        cin >> data[i];
    }
    initial(data, n);
    for (int i = 1; i <= n; i++) {
        for (int j = 0; j <= lg2[n]; j++) {
            cout << st[i][j] << " ";
        }
        cout << endl;
    }
    int m;
    cin >> m;
    for (int i = 1; i <= m; i++) {
        int L, R;
        cin >> L >> R;
        cout << query(L, R) << endl;
    }
    return 0;
```

```
50 }
51
52
```

```cpp
const int MAXN = 1e5+4;
const int M = MAXN * 30;
int Root[MAXN],lson[M],rson[M],data[M];
int build(int l,int r)
{
    int root=tot++;
    data[root]=0;
    if(l!=r)
    {
        int mid=(l+r)>>1;
        lson[root]=build(l,mid);
        rson[root]=build(mid+1,r);
    }
    return root;
}

int update(int root,int pos,int val){
    int newroot = tot++,res=newroot;
    data[newroot]=data[root]+val;
    int l=1,r=Discrete_cnt;
    while (l<r) {
        int mid = (l+r)>>1;
        if(pos<=mid)
        {
            lson[newroot]=tot++;
            rson[newroot]=rson[root];
            newroot=lson[newroot];
            root=lson[root];
            r=mid;
        }
        else
        {
            rson[newroot]=tot++;
            lson[newroot]=lson[root];
            newroot=rson[newroot];
            root=rson[root];
            l=mid+1;
        }
        data[newroot]=data[root]+val;
    }
    return res;
}

```

```cpp
//  并查集
//  ⚠查询时要用query(x)不能直接parent[x]⚠

const int MAXN = 10005;
int parent[MAXN];
int query(int x)
{
    if(x!=parent[x]) return parent[x]=query(parent[x]);
    return parent[x];
}
void combine(int a,int b)
{
    parent[query(b)]=query(a);
}
void initial()
{
    for(int i=0;i<MAXN;i++)
        parent[i]=i;
}


```

```cpp
//整数计算几何
#include <cmath>

typedef long long ll;
const double PI = acos(-1.0);


struct Point {
    int x, y;

    //按x坐标排序
    bool operator<(const Point &b) const { return x < b.x; }

    Point() {}

    Point(int x, int y) : x(x), y(y) {}


    //平方距离
    ll sqdis_to(const Point &b) const {
        return (ll) (x - b.x) * (ll) (x - b.x) + (ll) (y - b.y) * (ll) (y - b.y);
    }

    //浮点距离
    double dis_to(const Point &b) const {
        return sqrt(sqdis_to(b));
    }

    //模长
    double length() {
        return sqrt(sqdis_to(Point(0, 0)));
    }

    //加减法
    Point operator+(const Point &b) const {
        return Point(x + b.x, y + b.y);
    }

    Point operator-(const Point &b) const {
        return Point(x - b.x, y - b.y);
    }

    //对位乘积
    Point digit_product(const Point &b) const {
        return Point(x * b.x, y * b.y);
    }

    //点乘 内积
```

```
49      ll operator*(const Point &b) const {
50          return (ll) x * (ll) b.x + (ll) y * (ll) b.y;
51      }
52
53
54      //叉积
55      ll cross_product(const Point &b) const {
56          return (ll) x * (ll) b.y - (ll) y * (ll) b.x;
57      }
58
59      //以此点为原点的叉积
60      ll cross_product(const Point &a, const Point &b) const
    {
61          return (a - *this).cross_product(b - *this);
62      }
63
64      //复数乘积
65      Point complex_product(const Point &b) const {
66          return Point(x * b.x - y * b.y, x * b.y + y * b.x);
67      }
68
69      friend Point operator*(const Point &a, int b) {
70          return Point(a.x * b, a.y * b);
71      }
72
73      friend Point operator/(const Point &a, int b) {
74          return Point(a.x / b, a.y / b);
75      }
76
77      //相等
78      bool operator==(const Point &b) const {
79          if (x == b.x && y == b.y)
80              return true;
81          else
82              return false;
83      }
84
85      //于x抽正方向的夹角[0,2*PI)
86      double theta() {//和原点连线与x轴正方向的夹角，0~2*PI
87          if (x == 0) {
88              if (y > 0)
89                  return PI / 2;
90              else if (y == 0)
91                  return 0;
92              else if (y < 0)
93                  return PI * 3 / 2;
94          } else if (y == 0) {
95              if (x > 0)
96                  return 0;
97              else if (x < 0)
```

```
 98                 return PI;
 99         } else {
100             double t = atan(1.0 * y / x);
101             if (x < 0)
102                 return t + PI;
103             else {
104                 if (y < 0)
105                     return t + 2 * PI;
106                 else
107                     return t;
108             }
109         }
110     }
111 
112     bool point_in_polygen(Point p[], int n) const {
113         //点需要逆时针排列
114         //是否被包含在凸包内部，包含边界
115         if (n == 1) {
116             if (*this == p[0])
117                 return true;
118             else
119                 return false;
120         } else if (n == 2) {
121             if (this->cross_product(p[0], p[1]) == 0 && (p
    [0] - *this) * (p[1] - *this) <= 0)
122                 return true;
123             else
124                 return false;
125         } else {
126             for (int i = 0; i < n; i++) {
127                 if (p[i].cross_product(*this, p[(i + 1) %
    n]) > 0) {
128                     return false;
129                 }
130             }
131             return true;
132         }
133     }
134 
135     friend bool seg_cross(Point &a,Point &b,Point &c,Point
     &d){
136         if(a.cross_product(b,c)*a.cross_product(b,d)<0
137            &&c.cross_product(d,a)*c.cross_product(d,b)<0){
138             return true;
139         }else{
140             if(c.cross_product(a,b)==0&&(a-c)*(b-c)<=0)
141                 return true;
142             if(d.cross_product(a,b)==0&&(a-d)*(b-d)<=0)
143                 return true;
144             if(a.cross_product(c,d)==0&&(c-a)*(d-a)<=0)
```

```
145                 return true;
146             if(b.cross_product(c,d)==0&&(c-b)*(d-b)<=0)
147                 return true;
148         }
149         return false;
150     }
151 
152     void debug() {
153         printf("(%d %d)\n", x, y);
154     }
155 
156     void debugl() {
157         printf("(%d %d)", x, y);
158     }
159 };
160 
161 
162 //极角排序
163 Point O = Point(0, 0);//原点
164 bool rotate_cmp(Point a, Point b) {
165 
166     if ((a - O).cross_product(b - O) == 0)//
    如果两个点幅角相等，则近点靠前,近点靠前才能保证原点在第一个
167         return O.sqdis_to(a) < O.sqdis_to(b);
168     else
169         return O.cross_product(a, b) > 0;
170 }
171 
172 //Graham扫描法 O(n*log(n))
173 struct Graham {
174     Point p[MAXN];//数据
175     Point st[MAXN];//点栈
176     int tail = 0;
177 
178     //solve之后：st中的点就是凸包，tail即是凸包中点的个数
179     void solve(int n) {
180         //找到y最小点
181         int x = 1;
182         for (int i = 0; i < n; i++) {
183             if (p[i].y < p[x].y || (p[i].y == p[x].y && p[
    i].x < p[x].x)) {
184                 x = i;
185             }
186         }
187         //极角排序
188         O = p[x];
189         sort(p, p + n, rotate_cmp);
190 
191         //开始Graham
192         tail = 0;
```

```
193         st[tail++] = p[0];
194         for (int i = 1; i < n; i++) {
195             if (tail < 2) {
196                 //不能保证第二个点一定是凸包顶点
197                 st[tail++] = p[i];
198             }
199             while (st[tail - 2].cross_product(p[i], st[
    tail - 1]) >= 0 && tail >= 2) {
200                 tail--;
201             }
202             st[tail++] = p[i];
203         }
204     }
205 
206     bool point_in(Point &b) {
207         return b.point_in_polygen(st, tail);
208     }
209 
210     bool intersect(Graham &b) {
211         //是否和另一个凸包相交
212 
213         //判断任意两边是否跨立
214         //这里改成 >=2 会wa很奇怪
215         if (tail == 2 && b.tail == 2)
216             for (int i = 0; i < tail; i++) {
217                 for (int j = 0; j < b.tail; j++) {
218                     if (st[i].cross_product(st[(i + 1) %
    tail], b.st[j]) *
219                         st[i].cross_product(st[(i + 1) %
    tail], b.st[(j + 1) % b.tail]) < 0
220                         && b.st[j].cross_product(b.st[(j +
     1) % b.tail], st[i]) *
221                            b.st[j].cross_product(b.st[(j +
     1) % b.tail], st[(i + 1) % tail]) < 0)
222                         return true;
223                 }
224             }
225 
226         //其他情况相互判断有没有在其内部
227         for (int i = 0; i < tail; i++) {
228             if (b.point_in(st[i])) {
229                 return true;
230             }
231         }
232         for (int i = 0; i < b.tail; i++) {
233             if (point_in(b.st[i])) {
234                 return true;
235             }
236         }
237         return false;
```

```
238     }
239
240     //RC旋转卡壳
241     double rotating_caliper() {
242         ll max_dis = 0;
243         if (tail == 2) {
244             max_dis = st[0].sqdis_to(st[1]);
245         } else {
246             st[tail] = st[0];
247             int j = 2;
248             for (int i = 0; i < tail; i++) {
249                 while (abs(st[i].cross_product(st[i + 1],
    st[j])) < abs(st[i].cross_product(st[i + 1], st[j + 1])))
    {
250                     j = (j + 1) % tail;
251                 }
252                 max_dis = max(max_dis, max(st[j].sqdis_to(
    st[i]), st[j].sqdis_to(st[i + 1])));
253             }
254         }
255         return sqrt(max_dis);
256     }
257
258     void debug() {
259         for (int i = 0; i < tail; i++)
260             st[i].debug();
261         cout << endl;
262     }
263 };
264
265
266
267
```

```cpp
//浮点计算几何
const int MAXN = 1e5+4;
const double PI = acos(-1.0);
const int INF = 0x3f3f3f3f;
const double rad_to_deg = 360/2/PI;
const double deg_to_rad = 2*PI/360;
const double EPS = 1e-6;
int sgn(double x){
    if(abs(x)<EPS)
        return 0;
    else if(x<0)
        return -1;
    else
        return 1;
}
struct Point {
    double x,y;
    bool operator < (const Point& b) const
    {return x<b.x;}
    Point(double x,double y):x(x),y(y){}
    Point(){}
    double sqdis_to(const Point&b){
        return (x-b.x)*(x-b.x)+(y-b.y)*(y-b.y);
    }
    double dis_to(const Point&b){
        return sqrt((x-b.x)*(x-b.x)+(y-b.y)*(y-b.y));
    }
    double length(){
        return dis_to(Point(0,0));
    }
    Point normalize(){
        return (*this)*(1/length());
    }

    friend double operator* (const Point &a,const Point &b)
    {
        return a.x*b.x+a.y*b.y;
    }
    Point operator * (const double &b) const{
        return Point(x*b,y*b);
    }
    friend Point operator *(double a,const Point &b){
        return Point(b.x*a,b.y*a);
    }
    friend Point operator + (const Point&a,const Point &b)
    {
        return Point(a.x+b.x,a.y+b.y);
    }
    friend Point operator - (const Point&a,const Point &b){
        return Point(a.x-b.x,a.y-b.y);
```

```
49      }
50      bool operator == (const Point &b) const{
51          if(sgn(x-b.x)==0&&sgn(y-b.y)==0)
52              return 1;
53          return 0;
54      }
55      double cross_product(const Point&a, const Point &b){
56          return (a.x-x)*(b.y-y)-(a.y-y)*(b.x-x);
57      }
58      double theta(){//和原点连线与x轴正方向的夹角，0～2*PI
59          if(sgn(x)==0){
60              if(sgn(y)>0)
61                  return PI/2;
62              else if(sgn(y)==0)
63                  return 0;
64              else
65                  return PI*3/2;
66          }else if(sgn(y)==0){
67              if(sgn(x)>0)
68                  return 0;
69              else
70                  return PI;
71          }else{
72              double t = atan(y/x);
73              if(sgn(x)<0)
74                  return t+PI;
75              else{
76                  if(sgn(y)<0)
77                      return t+2*PI;
78                  else
79                      return t;
80              }
81          }
82      }
83      Point complex_mul(const Point&b)const{
84          return Point(x*b.x-y*b.y,x*b.y+y*b.x);
85      }
86      Point rotate(double rad){
87          return this->complex_mul(unit(rad));
88      }
89      Point rotate(const Point& b,double rad){
90          Point d = *this-b;
91          return b+d.rotate(rad);
92      }
93      static Point unit(double rad){
94          return Point(cos(rad),sin(rad));
95      }
96      void debuge(){
97          printf("(%lf %lf)\n",x,y);
98      }
```

```
 99 };
100
101 struct Circle{
102     Point c;
103     double r;
104     Circle(){}
105     Circle(Point c,int r):c(c),r(r){}
106     int relation(Point p){
107         double d = sgn(p.dis_to(c)-r);
108         if(d>0)
109             return 0;//圆外
110         else if(d==0)
111             return 1;//圆上
112         else
113             return 2;//圆内
114     }
115     int relation(Circle b){
116         double dis = c.dis_to(b.c);
117         if(sgn(dis-r-b.r)>0){
118             return 5;
119         }else if(sgn(dis-r-b.r)==0){
120             return 4;
121         }else{
122             double idi = abs(r-b.r);
123             if(sgn(dis-idi)>0){
124                 return 3;
125             }else if(sgn(dis-idi)==0){
126                 return 2;
127             }else{
128                 return 1;
129             }
130         }
131         //5 相离
132         //4 外切
133         //3 相交
134         //2 内切
135         //1 内含
136     }
137     int cross(Circle b,Point &c1,Point &c2){
138         int rel = relation(b);
139         if(rel==5||rel==1){
140             return 0;
141         }
142         double R=b.r;
143         Point dv = c-b.c;
144         double t1=asin((R*R-r*r+dv*dv)/(2*R*sqrt(dv*dv))),
    t2=atan(dv.x/dv.y);
145         if(sgn(dv.y)<0) t2-=PI;
146         double theta1 = t1-t2,theta2 = PI-t1-t2;
147         c1 = b.c+Point(R*cos(theta1),R*sin(theta1));
```

```
148         c2 = b.c+Point(R*cos(theta2),R*sin(theta2));
149         if(rel==4||rel==2)
150             return 1;
151         else
152             return 2;
153     }
154 }
155
156
157 /*三角形外心*/
158 Point circumcenter(Point A,Point B,Point C)
159 {
160     Point ret;
161     double a1=B.x-A.x,b1=B.y-A.y,c1=(a1*a1+b1*b1)/2;
162     double a2=C.x-A.x,b2=C.y-A.y,c2=(a2*a2+b2*b2)/2;
163     double d=a1*b2-a2*b1;
164     ret.x=A.x+(c1*b2-c2*b1)/d;
165     ret.y=A.y+(a1*c2-a2*c1)/d;
166     return ret;
167 }
168
169
170 /*最小覆盖圆 c为圆心，r为半径 */
171 double min_cover_circle(Point *p,Point &c,int n)
172 {
173     double r;
174     random_shuffle(p,p+n);
175     c=p[0]; r=0;
176     for(int i=1;i<n;i++)
177     {
178         if(p[i].dis_to(c)>r+EPS)   //第一个点
179         {
180             c=p[i]; r=0;
181             for(int j=0;j<i;j++)
182                 if(p[j].dis_to(c)>r+EPS)  //第二个点
183                 {
184                     c.x=(p[i].x+p[j].x)/2;
185                     c.y=(p[i].y+p[j].y)/2;
186                     r=p[j].dis_to(c);
187                     for(int k=0;k<j;k++)
188                         if(p[k].dis_to(c)>r+EPS)  //第三个点
189                         {   //求外接圆圆心，三点必不共线
190                             c=circumcenter(p[i],p[j],p[k])
    ;
191                             r=p[i].dis_to(c);
192                         }
193                 }
194         }
195     }
196     return r;
```

```
197 }
198
```

```cpp
//多边形与圆相交面积
const double eps = 1e-10;
inline int equal_to_0 (double a)
{
    if (a > eps) return 1;
    else if (a >= -eps) return 0;
    else return -1;
}
class Vector
{
public:
    double x, y;
    Vector (void) {}
    Vector (double x0, double y0) : x(x0), y(y0) {}
    double operator * (const Vector& a) const { return x * a.y - y * a.x; }
    double operator % (const Vector& a) const { return x * a.x + y * a.y; }
    Vector verti (void) const { return Vector(-y, x); }
    double length (void) const { return sqrt(x * x + y * y); }
    Vector adjust (double len)
    {
        double ol = len / length();
        return Vector(x * ol, y * ol);
    }
    Vector oppose (void) { return Vector(-x, -y); }
};
class point
{
public:
    double x, y;
    point (void) {}
    point (double x0, double y0) : x(x0), y(y0) {}
    Vector operator - (const point& a) const { return Vector(x - a.x, y - a.y); }
    point operator + (const Vector& a) const { return point(x + a.x, y + a.y); }
};
class segment
{
public:
    point a, b;
    segment (void) {}
    segment (point a0, point b0) : a(a0), b(b0) {}
    point intersect (const segment& s) const
    {
        Vector v1 = s.a - a, v2 = s.b - a, v3 = s.b - b, v4 = s.a - b;
        double s1 = v1 * v2, s2 = v3 * v4;
```

```
45          double se = s1 + s2;
46          s1 /= se, s2 /= se;
47          return point(a.x * s2 + b.x * s1, a.y * s2 + b.y *
    s1);
48      }
49      point pverti (const point& p) const
50      {
51          Vector t = (b - a).verti();
52          segment uv(p, p + t);
53          return intersect(uv);
54      }
55      bool on_segment (const point& p) const
56      {
57          if (equal_to_0(min(a.x, b.x) - p.x) <= 0 &&
    equal_to_0(p.x - max(a.x, b.x)) <= 0 &&
58              equal_to_0(min(a.y, b.y) - p.y) <= 0 &&
    equal_to_0(p.y - max(a.y, b.y)) <= 0) return true;
59          else return false;
60      }
61 };
62
63
64 double kuras_area (point a, point b, double cx, double cy,
    double radius)
65 {
66      point ori(cx, cy);
67      int sgn = equal_to_0((b - ori) * (a - ori));
68      double da = (a - ori).length(), db = (b - ori).length()
    ;
69      int ra = equal_to_0(da - radius), rb = equal_to_0(db -
    radius);
70      double angle = acos(((b - ori) % (a - ori)) / (da * db)
    );
71      segment t(a, b); point h, u; Vector seg;
72      double ans, dlt, mov, tangle;
73
74      if (equal_to_0(da) == 0 || equal_to_0(db) == 0) return
    0;
75      else if (sgn == 0) return 0;
76      else if (ra <= 0 && rb <= 0) return fabs((b - ori) * (a
     - ori)) / 2 * sgn;
77      else if (ra >= 0 && rb >= 0)
78      {
79          h = t.pverti(ori);
80          dlt = (h - ori).length();
81          if (!t.on_segment(h) || equal_to_0(dlt - radius) >=
     0)
82              return radius * radius * (angle / 2) * sgn;
83          else
84          {
```

```
 85                 ans = radius * radius * (angle / 2);
 86                 tangle = acos(dlt / radius);
 87                 ans -= radius * radius * tangle;
 88                 ans += radius * sin(tangle) * dlt;
 89                 return ans * sgn;
 90             }
 91         }
 92         else
 93         {
 94             h = t.pverti(ori);
 95             dlt = (h - ori).length();
 96             seg = b - a;
 97             mov = sqrt(radius * radius - dlt * dlt);
 98             seg = seg.adjust(mov);
 99             if (t.on_segment(h + seg)) u = h + seg;
100             else u = h + seg.oppose();
101             if (ra == 1) swap(a, b);
102             ans = fabs((a - ori) * (u - ori)) / 2;
103             tangle = acos(((u - ori) % (b - ori)) / ((u - ori)
    .length() * (b - ori).length()));
104             ans += radius * radius * (tangle / 2);
105             return ans * sgn;
106         }
107 }
108
```