hz 的板子

hz

2024 年 11 月 1 日

目录

1 高精度及优化技巧

1.1 快读快写

```
#include <bits/stdc++.h>
using namespace std;

inline void read(int &n)
{
  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 << 1) + (x << 3) + (ch ^ 48);
    ch = getchar();
  }
  n = x * f;
}
inline void write(int n)
{
  if (n < 0)
  {
    putchar('-');
    n *= -1;
  }
  if (n > 9)
  write(n / 10);
  putchar(n % 10 + '0');
}

```

1.2 边读入边取模

```cpp
#include <bits/stdc++.h>
using namespace std;

inline int read(int mod)
{
  int x = 0;
  bool g = false;
  char c = getchar();
  while (c < '0' || c > '9')
  c = getchar();
  while (c >= '0' && c <= '9')
  {
    x = (x << 3) + (x << 1) + (c ^ '0');
    if (x >= mod)
    x %= mod, g = true;
    c = getchar();
  }
  if (g)
  return (x + mod);
  else
  return x;
}

```

1.3 离散化

```cpp
// arr[i] 为初始数组,下标范围为 [1, n]

for (int i = 1; i <= n; ++i)
  tmp[i] = arr[i];
std::sort(tmp + 1, tmp + n + 1);
int len = std::unique(tmp + 1, tmp + n + 1) - (tmp + 1);
for (int i = 1; i <= n; ++i)
  arr[i] = std::lower_bound(tmp + 1, tmp + len + 1, arr[i])
- tmp;
```

```

```

或

```
// std::vector<int> arr;
std::vector<int> tmp(arr);  // tmp 是 arr 的一个副本
std::sort(tmp.begin(), tmp.end());
tmp.erase(std::unique(tmp.begin(), tmp.end()), tmp.end());
for (int i = 0; i < n; ++i)
  arr[i] = std::lower_bound(tmp.begin(), tmp.end(), arr[i])
- tmp.begin();

```

1.4 高精度

1.4.1 普通整数高精度

```
#include <bits/stdc++.h>
using namespace std;

struct Number
{
  string s;
  bool sign = 0;
};
int compare(string str1, string str2)
{
  if (str1.length() > str2.length())
  return 1;
  else if (str1.length() < str2.length())
  return -1;
  else
  return str1.compare(str2);
}
string add(string str1, string str2)
{
  string str;
  int len1 = str1.length();
```

```
22        int len2 = str2.length();
23        if (len1 < len2)
24        {
25          for (int i = 1; i <= len2 - len1; i++)
26          str1 = "0" + str1;
27        }
28        else
29        {
30          for (int i = 1; i <= len1 - len2; i++)
31          str2 = "0" + str2;
32        }
33        len1 = str1.length();
34        int cf = 0;
35        int temp;
36        for (int i = len1 - 1; i >= 0; i--)
37        {
38          temp = str1[i] - '0' + str2[i] - '0' + cf;
39          cf = temp / 10;
40          temp %= 10;
41          str = char(temp + '0') + str;
42        }
43        if (cf != 0)
44        str = char(cf + '0') + str;
45        return str;
46      }
47      string sub(string str1, string str2)
48      {
49        string str;
50        int tmp = str1.length() - str2.length();
51        int cf = 0;
52        for (int i = str2.length() - 1; i >= 0; i--)
53        {
54          if (str1[tmp + i] < str2[i] + cf)
55          {
56            str = char(str1[tmp + i] - str2[i] - cf + '0' + 10) +
    str;
57            cf = 1;
58          }
```

```
59          else
60          {
61            str = char(str1[tmp + i] - str2[i] - cf + '0') + str;
62            cf = 0;
63          }
64        }
65        for (int i = tmp - 1; i >= 0; i--)
66        {
67          if (str1[i] - cf >= '0')
68          {
69            str = char(str1[i] - cf) + str;
70            cf = 0;
71          }
72          else
73          {
74            str = char(str1[i] - cf + 10) + str;
75            cf = 1;
76          }
77        }
78        str.erase(0, str.find_first_not_of('0'));
79        return str;
80      }
81      string mul(string str1, string str2)
82      {
83        string str;
84        int len1 = str1.length();
85        int len2 = str2.length();
86        string tempstr;
87        for (int i = len2 - 1; i >= 0; i--)
88        {
89          tempstr = "";
90          int temp = str2[i] - '0';
91          int t = 0;
92          int cf = 0;
93          if (temp != 0)
94          {
95            for (int j = 1; j <= len2 - 1 - i; j++)
96            tempstr += "0";
```

```
97            for (int j = len1 - 1; j >= 0; j--)
98            {
99              t = (temp * (str1[j] - '0') + cf) % 10;
100             cf = (temp * (str1[j] - '0') + cf) / 10;
101             tempstr = char(t + '0') + tempstr;
102           }
103           if (cf != 0)
104           tempstr = char(cf + '0') + tempstr;
105         }
106         str = add(str, tempstr);
107       }
108       str.erase(0, str.find_first_not_of('0'));
109       return str;
110     }
111     void div(string str1, string str2, string &quotient, string
    &residue)
112     {
113       quotient = residue = "";
114       if (str2 == "0")
115       {
116         quotient = residue = "ERROR";
117         return;
118       }
119       if (str1 == "0")
120       {
121         quotient = residue = "0";
122         return;
123       }
124       int res = compare(str1, str2);
125       if (res < 0)
126       {
127         quotient = "0";
128         residue = str1;
129         return;
130       }
131       else if (res == 0)
132       {
133         quotient = "1";
```

```
134         residue = "0";
135         return;
136       }
137       else
138       {
139         int len1 = str1.length();
140         int len2 = str2.length();
141         string tempstr;
142         tempstr.append(str1, 0, len2 - 1);
143         for (int i = len2 - 1; i < len1; i++)
144         {
145           tempstr = tempstr + str1[i];
146           tempstr.erase(0, tempstr.find_first_not_of('0'));
147           if (tempstr.empty())
148           tempstr = "0";
149           for (char ch = '9'; ch >= '0'; ch--)
150           {
151             string str, tmp;
152             str = str + ch;
153             tmp = mul(str2, str);
154             if (compare(tmp, tempstr) <= 0)
155             {
156               quotient = quotient + ch;
157               tempstr = sub(tempstr, tmp);
158               break;
159             }
160           }
161         }
162         residue = tempstr;
163       }
164       quotient.erase(0, quotient.find_first_not_of('0'));
165       if (quotient.empty())
166       quotient = "0";
167     }
168
```

1.4.2 小数除法高精

不会，别急

1.4.3 fft

(不会用, 能不用就不用)

```
#include<stdio.h>
#include<string.h>
typedef unsigned char byte;
typedef unsigned int word;
typedef unsigned long long ull;
typedef long long ll;
const word mod=1004535809,size=21;
//NTT模数,结果不超过(1<<size)
char io[(1<<size)+1];//输入输出
word a[1<<size],b[1<<size];//多项式/高精
word realid[1<<size],root[1<<size],inverse[1<<size];
//迭代后系数所在的位置,单位根及其逆元
word i,id,floor;
ll num1,num2;
char *top;
//循环变量
inline ll pow(ll a,ll b){//快速幂
  register ll ans=1;
  for(;b;b>>=1){
    if(b&1) (ans*=a)%=mod;
    (a*=a)%=mod;
  }
  return ans;
}
inline void loading(){//预处理迭代后位置,单位根及其逆元
  root[0]=inverse[0]=1;
  num1=pow(3,mod>>size);
  num2=pow(num1,mod-2);
  for(i=1;i<1<<size;i++){
    root[i]=num1*root[i-1]%mod;
    inverse[i]=num2*inverse[i-1]%mod;
```

```
32          for(id=i,floor=0;floor<size;floor++,id>>=1)
33          realid[i]=realid[i]<<1|(id&1);
34        }
35      }
36      inline void read(){
37        //直接用fread,然后指针前移读入数据,效率较高,写法较简单
38        //因fread特性,DEV-CPP下不能以数字结尾(不停读入最后一个字符
        ),但linux下可以
39        top=io+(1<<size);
40        fread(io+1,1,1<<size,stdin);
41        while('0'>*top||*top>'9') top--;
42        for(i=0;'0'<=*top&&*top<='9';top--,i++)
43        a[realid[i]]=*top-'0';//直接放到迭代后的位置
44        while('0'>*top||*top>'9') top--;
45        for(i=0;'0'<=*top&&*top<='9';top--,i++)
46        b[realid[i]]=*top-'0';//b同理
47      }
48      inline void DFT(){ //非迭代版DFT
49        for(floor=0;floor<size;floor++)
50        for(i=0;i<1<<size;i+=(1<<(floor+1)))
51        for(id=0;id<1<<floor;id++){
52          num1=a[i+id];//蝴蝶变换
53          num2=a[i+id+(1<<floor)];
54          (num2*=root[id<<size-floor-1])%=mod;
55          a[i+id]=(num1+num2)%mod;//放回原位
56          a[i+id+(1<<floor)]=(num1+mod-num2)%mod;
57
58          num1=b[i+id];//b同理
59          num2=b[i+id+(1<<floor)];
60          (num2*=root[id<<size-floor-1])%=mod;
61          b[i+id]=(num1+num2)%mod;
62          b[i+id+(1<<floor)]=(num1+mod-num2)%mod;
63        }
64      }
65      inline void IDFT(){
66        for(floor=0;floor<21;floor++)//与DFT相同
67        for(i=0;i<0x200000;i+=1<<floor+1)
68        for(id=0;id<1<<floor;id++){
```

```
69          num1=root[i+id];
70          num2=root[i+id+(1<<floor)];
71          (num2*=inverse[id<<20-floor])%=mod;//乘上单位根逆元
72          root[i+id]=(num1+num2)%mod;
73          root[i+id+(1<<floor)]=(num1+mod-num2)%mod;
74        }
75      }
76      inline void write(){//fwrite输出,同输入
77        num2=pow(1<<size,mod-2);
78        top=io+(1<<size);
79        num1=0;
80        for(i=0;i<1<<size;i++){
81          num1+=num2*root[i]%mod;//最后要乘上n的逆元
82          top--;
83          *top=num1%10+'0';
84          num1/=10;
85        }
86        while(*top=='0'&&top!=io+(1<<size)) top++;
87        if(top==io+(1<<size)) putchar('0');
88        else fwrite(top,1,io+(1<<size)-top,stdout);
89      }
90      int main(){
91        loading();
92        read();
93        DFT();
94        for(i=0;i<1<<size;i++)
95        root[realid[i]]=(ll)(a[i])*b[i]%mod;//点值相乘
96        IDFT();
97        write();
98        return 0;
99      }
100
```

1.5 随机数

```
1     template <class T>
```

```
T randint(T l, T r = 0) // 生成随机数建议用<random>里的引擎和分布，而不是rand()模数，那样保证是均匀分布
{
  static mt19937 eng(time(0));
  if (l > r)
  swap(l, r);
  uniform_int_distribution<T> dis(l, r);
  return dis(eng);
}

```

1.6 手写哈希

```
template<typename key_t, typename type> struct hash_table {
  static const int maxn = 1000010;
  static const int table_size = 11110007;
  int first[table_size], nxt[maxn], sz; //init: memset(first, 0, sizeof(first)), sz = 0
  key_t id[maxn];
  type data[maxn];
  type& operator[] (key_t key) {
    const int h = key % table_size;
    for (int i = first[h]; i; i = nxt[i])
    if (id[i] == key)
    return data[i];
    int pos = ++sz;
    nxt[pos] = first[h];
    first[h] = pos;
    id[pos] = key;
    return data[pos] = type();
  }
  bool count(key_t key) {
    for (int i = first[key % table_size]; i; i = nxt[i])
    if (id[i] == key)
    return true;
    return false;
  }
```

```
24      type get(key_t key) { //如果key对应的值不存在，则返回type
    ()。
25          for (int i = first[key % table_size]; i; i = nxt[i])
26          if (id[i] == key)
27          return data[i];
28          return type();
29        }
30      };
31
```

1.7 linux 对拍

```
1   #include <algorithm>
2   using namespace std;
3   int main(){
4     int T = 10000;
5     int tot = 0;
6     while(T--){
7       tot++;
8       cout << tot << " ";
9       system("./rand; ./std; ./tmp");
10      if(system("diff std.out tmp.out")){
11        cout << "WA" << endl;
12        return 0;
13      }
14      else cout << "AC" << endl;
15    }
16  }
17
```

```
1   make data
2   make 001
3   make 002
4
5   ((cnt=1))
6
7   while true
```

```
do
./data > in
./001 < in > 1.out
./002 < in > 2.out
if diff 1.out 2.out; then
printf "# $((cnt++)) Accepted\n";
else
notify-send "Gary"
break
fi
done
```

2 简单算法备忘录

2.1 三分答案

```
ll ans = 1e18;
while (l + 1 < r) // 直到只剩不到三个数为止
{
  mid = (l + r) / 2;
  ll a1 = run(mid - 1), a2 = run(mid + 1);
  if (a1 > a2)
  ans = min(ans, a1), l = mid;
  else
  ans = min(ans, a2), r = mid;
}
ans = min(ans, run(l));
ans = min(ans, run(r));
cout << ans << endl;
```

2.2 树状数组

```
void add(int k, int x) {
```

```
    while (k < N - 100) {
      tree[k] += x;
      k += lowbit(k);
    }
  }
  int sum(int k) {
    int ans = 0;
    while (k) {
      ans += tree[k];
      k -= lowbit(k);
    }
    return ans;
  }

```

2.3 线段树

```
  struct seg_tree {
    using type = i64;
    #define ls(x) x << 1
    #define rs(x) x << 1 | 1
    #define val(x) tree[x].val
    #define tag(x) tree[x].tag

    struct node {
      int ls, rs;
      type val, tag;
    }tree[N << 2];
    void pushdown(int p, int len) {
      if (len == 1) {
        return;
      }
      tag(ls(p)) = (tag(p) + tag(ls(p))) % mod;
      tag(rs(p)) = (tag(p) + tag(rs(p))) % mod;
      val(ls(p)) = (val(ls(p)) + tag(p) * (len - len / 2) %
mod) % mod;
```

```
19        val(rs(p)) = (val(rs(p)) + tag(p) * (len / 2) % mod) %
    mod;
20        tag(p) = 0;
21      }
22      void pushup(int p) {
23        val(p) = (val(ls(p)) + val(rs(p))) % mod;
24      }
25      void build(vector<int> &a, int l = 1, int r = n, int p =
    1) {
26        if (l == r) {
27          val(p) = a[l - 1];
28          return;
29        }
30        int mid = l + r >> 1;
31        build(a, l, mid, ls(p));
32        build(a, mid + 1, r, rs(p));
33        pushup(p);
34      }
35      void update(int l, int r, int k, int nl = 1, int nr = n,
    int p = 1) {
36        if (l <= nl && r >= nr) {
37          val(p) = (val(p) + k * (nr - nl + 1) % mod) % mod;
38          if (nl != nr) {
39            tag(p) = (tag(p) + k) % mod;
40          }
41          return;
42        }
43        pushdown(p, nr - nl + 1);
44        int mid = nl + nr >> 1;
45        if (mid >= l) {
46          update(l, r, k, nl, mid, ls(p));
47        }
48        if (mid < r) {
49          update(l, r, k, mid + 1, nr, rs(p));
50        }
51        pushup(p);
52      }
53      type query(int l, int r, int nl = 1, int nr = n, int p =
```

```
1) {
54        if (l <= nl && r >= nr) {
55          return val(p);
56        }
57        pushdown(p, nr - nl + 1);
58        int mid = nl + nr >> 1;
59        type ans = 0;
60        if (mid >= l) {
61          ans = (ans + query(l, r, nl, mid, ls(p))) % mod;
62        }
63        if (mid < r) {
64          ans = (ans + query(l, r, mid + 1, nr, rs(p))) % mod;
65        }
66        return ans;
67      }
68      void debug(int num = 1) {
69        for (int i = 1; i <= n; i += num) {
70          query(i, min(n, i + num - 1));
71        }
72        // cout << endl;
73      }
74    }
75
```

2.4 权值线段树

```
1     void insert(int v) // 插入
2     {
3       update(v, 1);
4     }
5     void remove(int v) // 删除
6     {
7       update(v, -1);
8     }
9     int countl(int v)
10    {
11      return query(L, v - 1);
```

```
12      }
13      int countg(int v)
14      {
15        return query(v + 1, R);
16      }
17      int rank(int v) // 求排名
18      {
19        return countl(v) + 1;
20      }
21      int kth(int k, int p = 1, int cl = L, int cr = R) // 求指定
        排名的数
22      {
23        if (cl == cr)
24        return cl;
25        int mid = (cl + cr - 1) / 2;
26        if (val(ls(p)) >= k)
27        return kth(k, ls(p), cl, mid); // 往左搜
28        else
29        return kth(k - val(ls(p)), rs(p), mid + 1, cr); // 往右搜
30      }
31      int pre(int v) // 求前驱
32      {
33        int r = countl(v);
34        return kth(r);
35      }
36      int suc(int v) // 求后继
37      {
38        int r = val(1) - countg(v) + 1;
39        return kth(r);
40      }
41
```

2.5 动态开点线段树

$MAXV$ 一般能开多大开多大，例如内存限制 $128M$ 时可以开到八百万左右

```
#define ls(x) tree[x].ls
#define rs(x) tree[x].rs
#define val(x) tree[x].val
#define mark(x) tree[x].mark
const int MAXV = 8e6;
int L = 1, R = 1e5, cnt = 1;
struct node
{
  ll val, mark;
  int ls, rs;
} tree[MAXV];
void upd(int &p, int x, int len)
{
  if (!p) p = ++cnt;
  val(p) += x * len;
  mark(p) += x;
}
void push_down(int p, int len)
{
  if (len <= 1) return;
  upd(ls(p), mark(p), len - len / 2);
  upd(rs(p), mark(p), len / 2);
  mark(p) = 0;
}
ll query(int l, int r, int p = 1, int cl = L, int cr = R)
{
  if (cl >= l && cr <= r) return val(p);
  push_down(p, cr - cl + 1);
  ll mid = (cl + cr - 1) / 2, ans = 0;
  if (mid >= l) ans += query(l, r, ls(p), cl, mid);
  if (mid < r) ans += query(l, r, rs(p), mid + 1, cr);
  return ans;
}
void update(int l, int r, int d, int p = 1, int cl = L, int
cr = R)
{
  if (cl >= l && cr <= r) return (void)(val(p) += d * (cr -
cl + 1), mark(p) += d);
```

```
37        push_down(p, cr - cl + 1);
38        int mid = (cl + cr - 1) / 2;
39        if (mid >= l) update(l, r, d, ls(p), cl, mid);
40        if (mid < r) update(l, r, d, rs(p), mid + 1, cr);
41        val(p) = val(ls(p)) + val(rs(p));
42      }
43
```

3 数据结构

3.1 分块

```
1     int main() {
2       int n, m;
3       cin >> n >> m;
4       int sq = sqrt(n);
5       for (int i = 1; i <= sq; ++i) {
6         st[i] = ed[i - 1] + 1;
7         ed[i] = n / sq * i;
8       }
9       ed[sq] = n;
10      for (int i = 1; i <= sq; ++i) {
11        for (int j = st[i]; j <= ed[i]; ++j) {
12          bel[j] = i;
13        }
14      }
15      for (int i = 1; i <= sq; ++i) {
16        size[i] = ed[i] - st[i] + 1;
17      }
18
```

3.2 树状数组

nlgn, 单点修改, 区间查询

```
1     int lowbit(int x) {
```

```
    return x & (-x);
  }
  void add(int k, int x) {
    while (k < N - 100) {
      tree[k] += x;
      k += lowbit(k);
    }
    return;
  }

  int sum(int x) {
    int ans = 0;
    while (x) {
      ans += tree[x];
      x -= lowbit(x);
    }
    return ans;
  }

```

3.3 ST 表

```
#include <bits/stdc++.h>
using namespace std;

const int N = 1e5 + 1e3;

int f[N][21];

int main() {
  int n, m;
  cin >> n >> m;
  for (int i = 1; i <= n; ++i) {
    int x;
    scanf("%d", &x);
    f[i][0] = x;
  }
```

```
    for (int i = 1; i <= 20; ++i) {
      for (int j = 1; j + (1 << i) - 1 <= n; ++j) {
        f[j][i] = max(f[j][i - 1], f[j + (1 << (i - 1))][i -
1]);
      }
    }
    while (m--) {
      int l, r;
      scanf ("%d %d", &l, &r);
      int s = __lg(r - l + 1);
      printf("%d\n", max(f[l][s], f[r - (1 << s) + 1][s]));
    }
    return 0;
  }

```

3.4 线段树

3.4.1 最大字段和

```
#include <bits/stdc++.h>
using namespace std;
#define sum(x) tree[x].sum
#define maxl(x) tree[x].maxl
#define maxr(x) tree[x].maxr
#define ans(x) tree[x].ans
#define mark(x) tree[x].mark

const int N = 5e5 + 100;

typedef struct node {
  int sum, maxl, maxr, ans;
  int mark;
  node() {
    sum = maxl = maxr = ans = 0;
    mark = 1e9;
  }
```

```
18        node(int x) {
19          sum = x;
20          maxl = x;
21          maxr = x;
22          ans = x;
23          mark = 1e9;
24        }
25      }T;
26
27      T init(int x) {
28        T node(x);
29        return node;
30      }
31      T tree[N << 2];
32      int a[N];
33      int n;
34      void push_down(int p) {
35        if (mark(p) != 1e9) {
36          tree[p << 1] = tree[p << 1 | 1] = init(mark(p));
37          mark(p << 1) = mark(p << 1 | 1) = mark(p);
38        }
39      }
40      void push_up(int p) {
41        sum(p) = sum(p << 1) + sum(p << 1 | 1);
42        maxl(p) = max(maxl(p << 1), sum(p << 1) + maxl(p << 1 | 1)
    );
43        maxr(p) = max(maxr(p << 1 | 1), sum(p << 1 | 1) + maxr(p
    << 1));
44        ans(p) = max(maxr(p << 1) + maxl(p << 1 | 1), max(ans(p <<
     1), ans(p << 1 | 1)));
45      }
46      T push_up(T A, T B) {
47        T nn;
48        nn.sum = A.sum + B.sum;
49        nn.maxl = max(A.maxl, A.sum + B.maxl);
50        nn.maxr = max(B.maxr, B.sum + A.maxr);
51        nn.ans = max(A.maxr + B.maxl, max(A.ans, B.ans));
52        return nn;
```

```
53      }
54      void build(int p = 1, int l = 1, int r = n) {
55        if (l == r) {
56          tree[p] = init(a[l]);
57          return;
58        }
59        int mid = l + r >> 1;
60        build(p << 1, l, mid);
61        build(p << 1 | 1, mid + 1, r);
62        push_up(p);
63      }
64      void update(int l, int r, int k, int p = 1, int cl = 1, int
    cr = n) {
65        if (l > cr || r < cl) {
66          return;
67        }
68        if (l <= cl && r >= cr) {
69          tree[p] = init(k);
70          return;
71        }
72        int mid = cl + cr >> 1;
73        push_down(p);
74        update(l, r, k, p << 1, cl, mid);
75        update(l, r, k, p << 1 | 1, mid + 1, cr);
76        push_up(p);
77      }
78      T query(int l, int r, int p = 1, int cl = 1, int cr = n) {
79        if (l <= cl && r >= cr) {
80          return tree[p];
81        }
82        int mid = cl + cr >> 1;
83        push_down(p);
84        if (l > mid) {
85          return query(l, r, p << 1 | 1, mid + 1, cr);
86        }
87        if (r < mid + 1) {
88          return query(l, r, p << 1, cl, mid);
89        }
```

```
90      return push_up(query(l, r, p << 1, cl, mid), query(l, r, p
    << 1 | 1, mid + 1, cr));
91    }
92    int main() {
93      int m;
94      cin >> n >> m;
95      for (int i = 1; i <= n; ++i) {
96        cin >> a[i];
97      }
98      build();
99      while (m--) {
100       int op;
101       cin >> op;
102       if (op == 1) {
103         int a, b;
104         cin >> a >> b;
105         if (a > b) {
106           swap(a, b);
107         }
108         cout << query(a, b).ans << endl;
109       } else {
110         int p, s;
111         cin >> p >> s;
112         update(p, p, s);
113       }
114     }
115     system("pause");
116   }
117
```

3.5 可持久化

3.5.1 可持久化数组 (单点修改单点查询)

```
1   #include <bits/stdc++.h>
2   using namespace std;
3
```

```
4   const int N = 3e7 + 1e2;
5   const int M = 1e6 + 1e2;
6
7   #define ls(x) tree[x].ls
8   #define rs(x) tree[x].rs
9   #define sum(x) tree[x].sum
10
11  typedef struct node {
12    int ls, rs;
13    int sum;
14  }T;
15
16  T tree[N];
17  int a[M];
18  int root[M];
19  int cnt;
20
21  int n;
22
23  void push_up(int p) {
24    sum(p) = sum(ls(p)) + sum(rs(p));
25  }
26  void build(int p = cnt, int l = 1, int r = n) {
27    if (l == r) {
28      sum(p) = a[l];
29      return;
30    }
31    int mid = l + r >> 1;
32    ls(p) = ++cnt;
33    rs(p) = ++cnt;
34    build(ls(p), l, mid);
35    build(rs(p), mid + 1, r);
36  }
37  void update(int q, int k, int root, int p, int cl = 1, int
    cr = n) {
38    //cout << q << ' ' << k << ' ' << root << ' ' << p << ' '
    << cl << ' ' << cr << endl;
39    if (cl == cr) {
```

```
40          sum(p) = k;
41          return;
42        }
43        int mid = cl + cr >> 1;
44        if (q <= mid) {
45          ls(p) = ++cnt;
46          rs(p) = rs(root);
47          update(q, k, ls(root), ls(p), cl, mid);
48        } else {
49          ls(p) = ls(root);
50          rs(p) = ++cnt;
51          update(q, k, rs(root), rs(p), mid + 1, cr);
52        }
53        push_up(p);
54      }
55      int query(int q, int p, int cl = 1, int cr = n) {
56        if (cl == cr) {
57          return sum(p);
58        }
59        int mid = cl + cr >> 1;
60        if (q <= mid) {
61          return query(q, ls(p), cl, mid);
62        } else {
63          return query(q, rs(p), mid + 1, cr);
64        }
65      }
66      int main() {
67        ios::sync_with_stdio(0);
68        int m;
69        cin >> n >> m;
70        for (int i = 1; i <= n; ++i) {
71          cin >> a[i];
72        }
73        root[0] = ++cnt;
74        build();
75        for (int i = 1; i <= m; ++i) {
76          int op, a, b;
77          cin >> a >> op >> b;
```

```
78          if (op == 1) {
79            root[i] = ++cnt;
80            int k;
81            cin >> k;
82            update(b, k, root[a], root[i]);
83          } else {
84            root[i] = root[a];
85            cout << query(b, root[a]) << endl;
86          }
87        }
88        system("pause");
89      }
90
```

3.5.2 主席树 (区间第 k 小)

```
1   #include <bits/stdc++.h>
2   using namespace std;
3
4   #define int long long
5   #define val(x) tree[x].val
6   #define ls(x) tree[x].ls
7   #define rs(x) tree[x].rs
8
9   const int N = 1e6;
10
11  typedef struct node {
12    int val;
13    int ls, rs;
14  }T;
15  T tree[N << 5];
16
17  int roots[N << 5], cnt = 1;
18  int a[N + 1000], b[N + 1000], c[N + 1000];
19
20  int dis(int n) {
21    memcpy(b, a, sizeof a);
```

```
22        sort(b + 1, b + n + 1);
23        int x = unique(b + 1, b + n + 1) - b - 1;
24        for (int i = 1; i <= n; ++i) {
25          c[i] = lower_bound(b + 1, b + x + 1, a[i]) - b;
26        }
27        return x;
28      }
29
30      void build(int p, int l, int r) {
31        val(p) = 0;
32        if (l == r) {
33          return;
34        }
35        ls(p) = ++cnt, rs(p) = ++cnt;
36        int mid = (l + r) >> 1;
37        build(ls(p), l, mid);
38        build(rs(p), mid + 1, r);
39      }
40      void update(int x, int p, int q, int l, int r) {
41        val(q) = val(p) + 1;
42        if (l == r) {
43          return;
44        }
45        ls(q) = ls(p), rs(q) = rs(p);
46        int mid = (l + r) >> 1;
47        if (x <= mid) {
48          ls(q) = ++cnt;
49          update(x, ls(p), ls(q), l, mid);
50        } else {
51          rs(q) = ++cnt;
52          update(x, rs(p), rs(q), mid + 1, r);
53        }
54      }
55
56      int query(int k, int p, int q, int l, int r) {
57        if (l == r) {
58          return b[l];
59        }
```

```
60      int mid = (l + r) >> 1;
61      //cout << l << " " << r << endl;
62      if (val(ls(q)) - val(ls(p)) >= k) {
63        return query(k, ls(p), ls(q), l, mid);
64      } else {
65        return query(k - val(ls(q)) + val(ls(p)), rs(p), rs(q),
    mid + 1, r);
66      }
67    }
68    main() {
69      ios::sync_with_stdio(0);
70      int n, m;
71      cin >> n >> m;
72      for (int i = 1; i <= n; ++i) {
73        cin >> a[i];
74      }
75      int x = dis(n);
76      build(1, 1, x);
77      roots[0] = 1;
78      for (int i = 1; i <= n; ++i) {
79        roots[i] = ++cnt;
80        update(c[i], roots[i - 1], roots[i], 1, x);
81      }
82      while (m--) {
83        int l, r, k;
84        cin >> l >> r >> k;
85        cout << query(k, roots[l - 1], roots[r], 1, x) << endl;
86      }
87      return 0;
88    }
89
```

3.6 线段树合并

```
1     namespace segtree {
2       typedef struct node {
3         int ls, rs;
```

```
4         int val, id;
5       }T;
6       int n = 1e5, cnt;
7       T tree[M];
8       int root[N];
9       void pushup(int p) {
10        val(p) = std::max(val(ls(p)), val(rs(p)));
11        if (val(ls(p)) >= val(rs(p))) {
12          id(p) = id(ls(p));
13        } else {
14          id(p) = id(rs(p));
15        }
16      }
17      void add(int pos, int d, int &p, int l = 1, int r = n) {
18        //std::cout << pos << ' ' << d << ' ' << p << ' ' << l
    << ' ' << r << std::endl;
19        if (!p) {
20          p = ++cnt;
21        }
22        if (l == r) {
23          id(p) = l;
24          val(p) += d;
25          return;
26        }
27        int mid = l + r >> 1;
28        if (pos <= mid) {
29          add(pos, d, ls(p), l, mid);
30        } else {
31          add(pos, d, rs(p), mid + 1, r);
32        }
33        pushup(p);
34        return;
35      }
36      int merge(int p, int q, int l = 1, int r = n) {
37        if (!p || !q) {
38          return p + q;
39        }
40        if (l == r) {
```

```
41            val(p) += val(q);
42            return p;
43          }
44          int mid = l + r >> 1;
45          ls(p) = merge(ls(p), ls(q), l, mid);
46          rs(p) = merge(rs(p), rs(q), mid + 1, r);
47          pushup(p);
48          return p;
49        }
50      }
51
```

3.7 带权并查集

```
1     int f[N];
2     int d[N];
3     void init(int n) {
4       for (int i = 0; i <= n; ++i) {
5         f[i] = i;
6       }
7     }
8     int find(int x) {
9       if (f[x] == x) {
10        return x;
11      }
12      int t = f[x];
13      f[x] = find(f[x]);
14      d[x] += d[t];
15      return f[x];
16    }
17    void merge(int x, int y, int v) {
18      int fx = find(x), fy = find(y);
19      if (fx == fy) {
20        return;
21      }
22      f[fx] = fy;
23      d[fx] = v + d[y] - d[x];
```

```
24      }
25
```

3.8 可撤销并查集

并查集按大小合并

```
1     class DSU {
2       private:
3       using p = pair<int, int>;
4       const static int N = 2e6;
5       int f[N + 100], sz[N + 100];
6       vector<p> stk;
7       public:
8       DSU() {
9         for (int i = 0; i <= N; ++i) {
10          f[i] = i;
11          sz[i] = 1;
12        }
13      }
14      int find(int x) {
15        if (f[x] == x) {
16          return x;
17        }
18        return find(f[x]);
19      }
20      bool merge(int x, int y) {
21        int fx = find(x), fy = find(y);
22        if (fx == fy) {
23          stk.push_back({-1, -1});
24          return 0;
25        }
26        if (sz[fx] > sz[fy]) {
27          swap(fx, fy);
28        }
29        f[fx] = fy;
30        sz[fy] += sz[fx];
31        stk.push_back({fx, fy});
```

```
32          return 1;
33        }
34        bool undo() {
35          if (!stk.empty()) {
36            auto [x, y] = stk.back();
37            stk.pop_back();
38            if (x == -1 && y == -1) {
39              return 0;
40            }
41            f[x] = x;
42            sz[y] -= sz[x];
43            return 1;
44          }
45          return 0;
46        }
47      }dsu;
48
```

3.9 莫队

$O(1)$ 时间扩展序列

```
1   #include <bits/stdc++.h>
2   using namespace std;
3
4   const int MAXN = 5e4 + 100, MAXQ = 5e4 + 100, MAXM = 5e4 +
    100;
5   int sq;
6   struct query // 把询问以结构体方式保存
7   {
8     int l, r, k, id;
9     bool operator<(const query &o) const // 重载<运算符，奇偶
    化排序
10    {
11      // 这里只需要知道每个元素归属哪个块，而块的大小都是sqrt(
    n)，所以可以直接用l/sq
12      if (l / sq != o.l / sq)
13      return l < o.l;
```

```
14          if (l / sq & 1)
15          return r < o.r;
16          return r > o.r;
17        }
18      } Q[MAXQ];
19      int A[MAXN], ans[MAXQ], Cnt[MAXM], l = 1, r = 0;
20      inline void add(int p){}
21      inline void del(int p){}
22      void solve(int i) {
23        while (l > Q[i].l)
24        add(--l);
25        while (r < Q[i].r)
26        add(++r);
27        while (l < Q[i].l)
28        del(l++);
29        while (r > Q[i].r)
30        del(r--);
31      }
32      int main()
33      {
34        int n, q;
35        cin >> n >> q;
36        sq = sqrt(n);
37        for (int i = 1; i <= n; ++i) {
38          cin >> A[i];
39        }
40        for (int i = 0; i < q; ++i) {
41          cin >> Q[i].l >> Q[i].r >> Q[i].k;
42          Q[i].id = i; // 把询问离线下来
43        }
44        sort(Q, Q + q);                                    // 排序
45        for (int i = 0; i < q; ++i)
46        {
47          solve(i);
48          //存储答案
49          //ans[Q[i].id] = ?
50        }
51        for (int i = 0; i < q; ++i)
```

```
52          printf("%d\n", ans[i]); // 按编号顺序输出
53        system("pause");
54      }
55
```

3.10 珂朵莉树

```
1       struct node
2       {
3         ll l, r;
4         mutable ll v;
5         node(ll l, ll r, ll v) : l(l), r(r), v(v) {}
6         bool operator<(const node &o) const { return l < o.l; }
7       };
8       set<node> tree;
9       auto split(ll pos)
10      {
11        auto it = tree.lower_bound(node(pos, 0, 0));
12        if (it != tree.end() && it->l == pos)
13        return it;
14        it--;
15        ll l = it->l, r = it->r, v = it->v;
16        tree.erase(it);
17        tree.insert(node(l, pos - 1, v));
18        return tree.insert(node(pos, r, v)).first;
19      }
20      void assign(ll l, ll r, ll v)
21      {
22        auto end = split(r + 1), begin = split(l);
23        tree.erase(begin, end);
24        tree.insert(node(l, r, v));
25      }
26
```

3.11 平衡树 (fhq-Treap)

```
struct node {
  int ls, rs, key, sz, val;
}fhq[N];
int cnt;
int create(int k) {
  fhq[++cnt] = {0, 0, rand(), 1, k};
  return cnt;
}
int root;
void pushup(int u) {
  sz(u) = sz(ls(u)) + sz(rs(u)) + 1;
}
void split(int u, int v, int &x, int &y) {
  if (!u) {
    x = y = 0;
    return;
  }
  if (val(u) > v) {
    y = u;
    split(ls(u), v, x, ls(u));
  } else {
    x = u;
    split(rs(u), v, rs(u), y);
  }
  pushup(u);
}
int merge(int x, int y) {
  if (!x || !y) {
    return x + y;
  }
  if (key(x) < key(y)) {
    ls(y) = merge(x, ls(y));
    pushup(y);
    return y;
  } else {
    rs(x) = merge(rs(x), y);
    pushup(x);
```

```
38          return x;
39        }
40      }
41      void insert(int k) {
42        int x, y;
43        create(k);
44        split(root, k, x, y);
45        root = merge(x, merge(cnt, y));
46      }
47      void del(int k) {
48        int x, y, z;
49        split(root, k, x, y);
50        split(x, k - 1, x, z);
51        z = merge(ls(z), rs(z));
52        root = merge(merge(x, z), y);
53      }
54      int kth(int k, int u = root) {
55        if (sz(ls(u)) + 1 > k) {
56          return kth(k, ls(u));
57        } else if (sz(ls(u)) + 1 == k) {
58          return u;
59        } else {
60          return kth(k - sz(ls(u)) - 1, rs(u));
61        }
62      }
63      int find_rank(int x) {
64        int x1, y1;
65        split(root, x - 1, x1, y1);
66        int ans = sz(x1) + 1;
67        root = merge(x1, y1);
68        return ans;
69      }
70      int find_pre(int x) {
71        int x1, y1;
72        split(root, x - 1, x1, y1);
73        int ans = x1;
74        while (rs(ans)) {
75          ans = rs(ans);
```

```
76        }
77        root = merge(x1, y1);
78        return ans;
79      }
80      int find_next(int x) {
81        int x1, y1;
82        split(root, x, x1, y1);
83        int ans = y1;
84        while (ls(ans)) {
85          ans = ls(ans);
86        }
87        root = merge(x1, y1);
88        return ans;
89      }
90
```

4 树上问题

4.1 点分治

```cpp
#include <bits/stdc++.h>
using namespace std;
#define int long long

const int N = 5e4 + 1e2;

vector<int> g[N];
int n, k;
int ans;

namespace CenDec {
  int sz[N];
  int del[N];
  int nn;
  void dfs(int p, int fa = 0) {
    sz[p] = 1;
    int mss = 0;
    for (int x : g[p]) {
      if (!del[x] && x != fa) {
        dfs(x, p);
        if (nn) {
          return;
        }
        sz[p] += sz[x];
        mss = max(mss, sz[x]);
      }
    }
    mss = max(n - sz[p], mss);
    if (mss <= n / 2) {
      nn = p;
      sz[fa] = n - sz[p];
      return;
    }
  }
```

```
35      int a[N];
36      map<int, int> mp;
37      void dfs2(int p, int fa, int len) {
38        if (len > k) {
39          return;
40        }
41        ans += a[k - len] + (len == k);
42        //cout << ans << endl;
43        mp[len]++;
44        for (int x : g[p]) {
45          if (!del[x] && x != fa) {
46            dfs2(x, p, len + 1);
47          }
48        }
49      }
50      void run(int p) {
51        for (int x : g[p]) {
52          if (!del[x]) {
53            dfs2(x, p, 1);
54            for (auto p : mp) {
55              a[p.first] += p.second;
56            }
57            mp.clear();
58          }
59        }
60        del[p] = 1;
61        for (int i = 0; i <= N - 100; ++i) {
62          a[i] = 0;
63        }
64        for (int x : g[p]) {
65          if (!del[x]) {
66            n = sz[x];
67            nn = 0;
68            dfs(x);
69            run(nn);
70          }
71        }
72      }
```

```
73      }
74      main() {
75        system("pause");
76      }
77
```

4.2 树链剖分

```
1     #include <bits/stdc++.h>
2     using namespace std;
3
4     #define int long long
5     const int N = 5e5 + 1e2;
6
7     vector<int> g[N];
8     int dep[N], fa[N], sz[N], hson[N];
9     int top[N], dfn[N], mxdfn[N];
10    int cnt;
11
12    int tree[N << 2], mark[N << 2];
13
14    int a[N], b[N];
15    int n;
16    int mod = LONG_LONG_MAX;
17
18    void dfs1(int p, int d = 1)
19    {
20      dep[p] = d;
21      int size = 1, ma = 0;
22      for (int u : g[p])
23      {
24        if (dep[u])
25        {
26          continue;
27        }
28        dfs1(u, d + 1);
29        fa[u] = p;
```

```
30          size += sz[u];
31          if (sz[u] > ma)
32          {
33            hson[p] = u;
34            ma = sz[u];
35          }
36        }
37        sz[p] = size;
38      }
39      void dfs2(int p)
40      {
41        dfn[p] = ++cnt;
42        if (hson[p])
43        {
44          top[hson[p]] = top[p];
45          dfs2(hson[p]);
46        }
47        for (int u : g[p])
48        {
49          if (top[u])
50          {
51            continue;
52          }
53          top[u] = u;
54          dfs2(u);
55        }
56        mxdfn[p] = cnt;
57      }
58      void push_down(int p, int len)
59      {
60        mark[p << 1] += mark[p];
61        mark[p << 1 | 1] += mark[p];
62        tree[p << 1] += mark[p] * (len - len / 2) % mod;
63        tree[p << 1 | 1] += mark[p] * (len / 2) % mod;
64        mark[p << 1] %= mod;
65        mark[p << 1 | 1] %= mod;
66        tree[p << 1] %= mod;
67        tree[p << 1 | 1] %= mod;
```

```
68        mark[p] = 0;
69      }
70      void build(int p = 1, int l = 1, int r = n)
71      {
72        if (l == r)
73        {
74          tree[p] = a[l];
75          return;
76        }
77        int mid = (l + r) >> 1;
78        build(p << 1, l, mid);
79        build(p << 1 | 1, mid + 1, r);
80        tree[p] = (tree[p << 1] + tree[p << 1 | 1]) % mod;
81      }
82      void update(int l, int r, int k, int p = 1, int rl = 1, int
    rr = n)
83      {
84        if (l > rr || r < rl)
85        {
86          return;
87        }
88        if (l <= rl && r >= rr)
89        {
90          tree[p] += k * (rr - rl + 1) % mod;
91          tree[p] %= mod;
92          if (rl != rr)
93          {
94            mark[p] += k;
95            mark[p] %= mod;
96          }
97          return;
98        }
99        push_down(p, rr - rl + 1);
100       int mid = (rl + rr) >> 1;
101       update(l, r, k, p << 1, rl, mid);
102       update(l, r, k, p << 1 | 1, mid + 1, rr);
103       tree[p] = tree[p << 1] + tree[p << 1 | 1];
104       tree[p] %= mod;
```

```
105     }
106     int query(int l, int r, int p = 1, int rl = 1, int rr = n)
107     {
108       if (l > rr || r < rl)
109       {
110         return 0;
111       }
112       if (l <= rl && r >= rr)
113       {
114         return tree[p];
115       }
116       int mid = (rl + rr) >> 1;
117       push_down(p, rr - rl + 1);
118       return (query(l, r, p << 1, rl, mid) + query(l, r, p << 1
    | 1, mid + 1, rr)) % mod;
119     }
120     void update_path(int x, int y, int k)
121     {
122       while (top[x] != top[y])
123       {
124         if (dep[top[x]] > dep[top[y]])
125         {
126           update(dfn[top[x]], dfn[x], k);
127           x = fa[top[x]];
128         }
129         else
130         {
131           update(dfn[top[y]], dfn[y], k);
132           y = fa[top[y]];
133         }
134       }
135       if (dep[x] > dep[y])
136       {
137         update(dfn[y], dfn[x], k);
138       }
139       else
140       {
141         update(dfn[x], dfn[y], k);
```

```
142       }
143     }
144     int query_path(int x, int y)
145     {
146       int ans = 0;
147       while (top[x] != top[y])
148       {
149         if (dep[top[x]] > dep[top[y]])
150         {
151           ans += query(dfn[top[x]], dfn[x]);
152           x = fa[top[x]];
153         }
154         else
155         {
156           ans += query(dfn[top[y]], dfn[y]);
157           y = fa[top[y]];
158         }
159         ans %= mod;
160       }
161       if (dep[x] > dep[y])
162       {
163         ans += query(dfn[y], dfn[x]);
164       }
165       else
166       {
167         ans += query(dfn[x], dfn[y]);
168       }
169       return ans % mod;
170     }
171     void update_subtree(int x, int k)
172     {
173       update(dfn[x], mxdfn[x], k);
174     }
175     int query_subtree(int x)
176     {
177       return query(dfn[x], mxdfn[x]);
178     }
179
```

4.3 长链剖分

```
int root;
int hson[N], fa[21][N], len[N], dfn[N], mdfn[N], top[N], dep[N];
int cnt;
void dfs1(int u = root, int f = -1)
{
  if (f != -1)
  {
    fa[0][u] = f;
    dep[u] = dep[f] + 1;
  }
  len[u] = 1;
  for (int x : g[u])
  {
    if (x == f)
    {
      continue;
    }
    dfs1(x, u);
    if (len[u] < len[x] + 1)
    {
      hson[u] = x;
      len[u] = len[x] + 1;
    }
  }
}
void dfs2(int u = root, int f = -1)
{
  dfn[u] = ++cnt;
  if (f == -1)
  {
    top[u] = u;
  }
```

```
33        for (int x : g[u])
34        {
35          if (x == f)
36          {
37            continue;
38          }
39          if (x == hson[u])
40          {
41            top[x] = top[u];
42          }
43          else
44          {
45            top[x] = x;
46          }
47          dfs2(x, u);
48        }
49      }
50      void po()
51      {
52        dfs1();
53        dfs2();
54      }
55
```

4.4 求树上 k 级祖先

```
1       int root;
2       int hson[N], fa[21][N], len[N], dfn[N], mdfn[N], top[N], dep
        [N];
3       int cnt;
4       void dfs1(int u = root, int f = -1)
5       {
6         if (f != -1)
7         {
8           fa[0][u] = f;
9           dep[u] = dep[f] + 1;
10        }
```

```
11        int maxn = 0;
12        len[u] = 1;
13        for (int x : g[u])
14        {
15          if (x == f)
16          {
17            continue;
18          }
19          dfs1(x, u);
20          if (len[u] < len[x] + 1)
21          {
22            hson[u] = x;
23            len[u] = len[x] + 1;
24          }
25        }
26        // len[u]++;
27      }
28      void dfs2(int u = root, int f = -1)
29      {
30        dfn[u] = ++cnt;
31        if (f == -1)
32        {
33          top[u] = u;
34        }
35        for (int x : g[u])
36        {
37          if (x == f)
38          {
39            continue;
40          }
41          if (x == hson[u])
42          {
43            top[x] = top[u];
44          }
45          else
46          {
47            top[x] = x;
48          }
```

```
49          dfs2(x, u);
50        }
51      }
52      void po()
53      {
54        dfs1();
55        dfs2();
56      }
57      int n;
58      vector<int> fro[N];
59      vector<int> bac[N];
60      void pre()
61      {
62        po();
63        for (int i = 1; i <= __lg(n); ++i)
64        {
65          for (int j = 1; j <= n; ++j)
66          {
67            fa[i][j] = fa[i - 1][fa[i - 1][j]];
68          }
69        }
70        for (int i = 1; i <= n; ++i)
71        {
72          if (top[i] == i)
73          {
74            int p = i;
75            for (int j = 1; j <= len[i]; ++j)
76            {
77              fro[i].push_back(p);
78              if (fa[0][p])
79              {
80                p = fa[0][p];
81              }
82            }
83            p = i;
84            for (int j = 1; j <= len[i] + 1; ++j)
85            {
86              bac[i].push_back(p);
```

```
            if (hson[p])
            {
              p = hson[p];
            }
          }
        }
      }
    }
    int query(int x, int k)
    {
      if (k == 0)
      {
        return x;
      }
      int xx = __lg(k);
      int q = fa[xx][x];
      int tt = top[q];
      int d = k - (1 << xx);
      if (dep[q] - dep[tt] >= d)
      {
        return bac[tt][dep[q] - dep[tt] - d];
      }
      else
      {
        return fro[tt][d - dep[q] + dep[tt]];
      }
    }

```

4.5 树上启发式合并

```
#include <bits/stdc++.h>
using namespace std;

const int N = 1e5 + 1e2;
using i64 = long long;
vector<int> g[N];
```

```
7     int hson[N], sz[N];
8     int c[N];
9     void dfs1(int u = 1, int f = -1) {
10      int maxn = 0;
11      for (int x : g[u]) {
12        if (x == f) {
13          continue;
14        }
15        dfs1(x, u);
16        sz[u] += sz[x];
17        if (maxn < sz[x]) {
18          maxn = sz[x];
19          hson[u] = x;
20        }
21      }
22      sz[u]++;
23    }
24
25    i64 cnt[N], ans[N], sum, maxn;
26    void add(int u) {}
27    void del(int u) {}
28    void subtree(bool flag, int u, int f = -1) {
29      if (flag) {
30        add(u);
31      } else {
32        del(u);
33      }
34      for (int x : g[u]) {
35        if (x == f) {
36          continue;
37        }
38        subtree(flag, x, u);
39      }
40    }
41    void dfs(int u = 1, int f = -1, bool flag = 1) {
42      for (int x : g[u]) {
43        if (x == f || x == hson[u]) {
44          continue;
```

```
45          }
46          dfs(x, u, 0);
47        }
48        if (hson[u]) {
49          dfs(hson[u], u, 1);
50        }
51        for (int x : g[u]) {
52          if (x == f || x == hson[u]) {
53            continue;
54          }
55          subtree(1, x, u);
56        }
57        add(u);
58        //subtree(1, u, f);
59        ans[u] = sum;
60        //sum = 0, maxn = 0;
61        if (!flag) {
62          subtree(0, u, f);
63          sum = maxn = 0;
64        }
65      }
66
```

4.6 lca

4.6.1 倍增求 lca

```
1   #include <bits/stdc++.h>
2   using namespace std;
3
4   const int N = 1e5 + 1e2;
5
6   vector<int> g[N];
7
8   bool vis[N];
9   int fa[N][22], depth[N];
10
```

```
11      void dfs(int x, int dd)
12      {
13        vis[x] = 1;
14        depth[x] = dd;
15        for (int i = 1; i <= __lg(depth[x]); ++i)
16        {
17          fa[x][i] = fa[fa[x][i - 1]][i - 1];
18        }
19        for (int u : g[x])
20        {
21          if (!vis[u])
22          {
23            fa[u][0] = x;
24            dfs(u, dd + 1);
25          }
26        }
27      }
28
29      int lca(int x, int y)
30      {
31        if (depth[x] < depth[y])
32        {
33          swap(x, y);
34        }
35        while (depth[y] < depth[x])
36        {
37          x = fa[x][__lg(depth[x] - depth[y])];
38        }
39        if (x == y)
40        {
41          return x;
42        }
43        for (int i = __lg(depth[x]); i >= 0; --i)
44        {
45          if (fa[x][i] != fa[y][i])
46          {
47            x = fa[x][i];
48            y = fa[y][i];
```

```
        }
      }
      return fa[x][0];
    }

```

4.6.2 树剖求 lca

```
#include <bits/stdc++.h>
using namespace std;

const int N = 5e5 + 1e2;

vector<int> g[N];
int dep[N], fa[N], sz[N], hson[N];
int top[N], dfn[N], mxdfn[N];
int cnt;

void dfs1(int p, int d = 1)
{
  dep[p] = d;
  int size = 1, ma = 0;
  for (int u : g[p])
  {
    if (dep[u])
    {
      continue;
    }
    dfs1(u, d + 1);
    fa[u] = p;
    size += sz[u];
    if (sz[u] > ma)
    {
      hson[p] = u;
      ma = sz[u];
    }
  }
```

```
30      sz[p] = size;
31    }
32    void dfs2(int p)
33    {
34      dfn[p] = ++cnt;
35      if (hson[p])
36      {
37        top[hson[p]] = top[p];
38        dfs2(hson[p]);
39      }
40      for (int u : g[p])
41      {
42        if (top[u])
43        {
44          continue;
45        }
46        top[u] = u;
47        dfs2(u);
48      }
49      mxdfn[p] = cnt;
50    }
51    int lca(int x, int y)
52    {
53      while (top[x] != top[y])
54      {
55        if (dep[top[x]] > dep[top[y]])
56        {
57          x = fa[top[x]];
58        }
59        else
60        {
61          y = fa[top[y]];
62        }
63      }
64      return dep[x] > dep[y] ? y : x;
65    }
66
```

5 图论

5.1 最短路

5.1.1 dijkstra

```cpp
#include <bits/stdc++.h>
using namespace std;

const int N = 2e5 + 1e2;
typedef pair<int, int> p;

vector<p> g[N];
int dis[N];
priority_queue<p, vector<p>, greater<p>> pq;

void dijkstra(int x)
{
  memset(dis, 127, sizeof dis);
  pq.push({0, x});
  dis[x] = 0;
  while (!pq.empty())
  {
    int d = pq.top().first, u = pq.top().second;
    pq.pop();
    if (d > dis[u])
    {
      continue;
    }
    for (auto p : g[u])
    {
      int v = p.first, w = p.second;
      if (w + dis[u] < dis[v])
      {
        dis[v] = dis[u] + w;
        pq.push({dis[v], v});
      }
    }
```

```
    }
  }

```

5.1.2 spfa

```
#include <bits/stdc++.h>
using namespace std;

const int N = 2e5 + 1e2;
typedef pair<int, int> p;

vector<p> g[N];
int dis[N];
queue<p> q;

void spfa(int x)
{
  memset(dis, 127, sizeof dis);
  q.push({0, x});
  dis[x] = 0;
  while (!q.empty())
  {
    int d = q.front().first, u = q.front().second;
    q.pop();
    if (d > dis[u])
    {
      continue;
    }
    for (auto p : g[u])
    {
      int v = p.first, w = p.second;
      if (w + dis[u] < dis[v])
      {
        dis[v] = dis[u] + w;
        q.push({dis[v], v});
      }
```

```
32              }
33          }
34      }
35
```

6 数学

6.1 线性筛

```cpp
#include <bits/stdc++.h>
using namespace std;

//#define int long long
const int N = 1e8 + 1e2;

bool prime[N];
vector<int> primeset;

void pre(int n) {
  prime[1] = 1;
  for (int i = 2; i <= n; ++i) {
    if (!prime[i]) {
      primeset.push_back(i);
    }
    for (int u : primeset) {
      if (i * u > n) {
        break;
      }
      prime[i * u] = 1;
      if (i % u == 0) {
        break;
      }
    }
  }
}

```

6.2 数论分块

求 $\sum_{i=1}^{n} f(i)\lfloor\frac{n}{i}\rfloor$

```cpp
int l = 1, r = 0;
```

```
    while (l <= n) {
      r = n / (n / l);
      if (r > n) {
        r = n;
      }
      ans += (s(r) - s(l - 1)) * (n / l);
      l = r + 1;
    }

```

6.3 欧拉函数

6.3.1 线性求欧拉函数

```
#include <bits/stdc++.h>
using namespace std;

const int N = 1e7 + 1e2;
bool prime[N];
vector<int> primeset;
int phi[N];

void pre(int n) {
  prime[1] = 1, phi[1] = 1;
  for (int i = 2; i <= n; ++i) {
    if (!prime[i]) {
      primeset.push_back(i);
      phi[i] = i - 1;
    }
    for (int u : primeset) {
      if (i * u > n) {
        break;
      }
      prime[i * u] = 1;
      if (i % u) {
        phi[i * u] = phi[i] * phi[u];
      } else {
```

```
            phi[i * u] = phi[i] * u;
            break;
        }
      }
    }
  }

```

6.3.2 单个数的欧拉函数

```
int phi(int n)
{
  int ans = n;
  for (int i = 2; i * i <= n; i++)
  {
    if (n % i == 0)
    {
      ans = ans / i * (i - 1);
      while (n % i == 0)
      n /= i;
    }
  }
  if (n > 1)
  ans = ans / n * (n - 1);
  return ans % mod;
}

```

6.4 扩展欧拉定理

求 $a^b \bmod p$

```
#include <bits/stdc++.h>
using namespace std;
#define int long long

const int N = 1e8 + 1e2;
```

```
6
7   int qpow(int n, int k, int mod)
8   {
9     int ans = 1;
10    while (k)
11    {
12      if (k % 2)
13      {
14        ans = (ans * n) % mod;
15      }
16      k /= 2;
17      n = (n * n) % mod;
18    }
19    return ans;
20  }
21  int phi(int n)
22  {
23    int ans = n;
24    for (int i = 2; i * i <= n; ++i)
25    {
26      if (n % i == 0)
27      {
28        ans = ans / i * (i - 1);
29      }
30      while (n % i == 0)
31      {
32        n /= i;
33      }
34    }
35    if (n > 1)
36    {
37      ans = ans / n * (n - 1);
38    }
39    return ans;
40  }
41
42  inline int read(int mod)
43  {
```

```
44      int x = 0;
45      bool g = false;
46      char c = getchar();
47      while (c < '0' || c > '9')
48      c = getchar();
49      while (c >= '0' && c <= '9')
50      {
51        x = (x << 3) + (x << 1) + (c ^ '0');
52        if (x >= mod)
53        x %= mod, g = true;
54        c = getchar();
55      }
56      if (g)
57      return (x + mod);
58      else
59      return x;
60    }
61    main()
62    {
63      int a, b, m;
64      cin >> a >> m;
65      int ph = phi(m);
66      int ans = 0;
67      ans = qpow(a, read(ph), m);
68      cout << ans << endl;
69      system("pause");
70    }
71
```

6.5 逆元

6.5.1 费马小定理

要求 p 是质数

```
1   inline ll qpow(ll a, ll n, ll p)// 快速幂
2   {
3     ll ans = 1;
```

```
    while (n)
    {
      if (n & 1)
      ans = ans % p * a % p;
      a = a % p * a % p;
      n >>= 1;
    }
    return ans;
  }
  inline ll inv(ll a, ll p)
  {
    return qpow(a, p - 2, p);
  }

```

6.5.2 扩展欧几里得

```
  ll exgcd(ll a, ll b, ll &x, ll &y)// 拓欧
  {
    if (b == 0)
    {
      x = 1;
      y = 0;
      return a;
    }
    ll d = exgcd(b, a % b, y, x);
    y -= (a / b) * x;
    return d;
  }
  ll inv(ll a, ll p)
  {
    ll x, y;
    if (exgcd(a, p, x, y) != 1) // 无解的情形
    return -1;
    return (x % p + p) % p;
  }

```

6.5.3 线性递推

```
inv[1] = 1;
for (int i = 2; i <= n; ++i) {
  inv[i] = (long long)(p - p / i) * inv[p % i] % p;
}

```

6.6 大步小步算法

解 $a^x \equiv b(\mathrm{mod} m)$, 其中 a, b, m 互质。

```
ll BSGS(ll a, ll b, ll m)
{
  static unordered_map<ll, ll> hs;
  hs.clear();
  ll cur = 1, t = sqrt(m) + 1;
  for (int B = 1; B <= t; ++B)
  {
    (cur *= a) %= m;
    hs[b * cur % m] = B; // 哈希表中存B的值
  }
  ll now = cur; // 此时cur = a^t
  for (int A = 1; A <= t; ++A)
  {
    auto it = hs.find(now);
    if (it != hs.end())
    return A * t - it->second;
    (now *= cur) %= m;
  }
  return -1; // 没有找到，无解
}

```

还有扩展大步小步，可以解 a 和 m 不互质的式子

```
// 修改版的BSGS，额外带一个系数
ll BSGS(ll a, ll b, ll m, ll k = 1)
{
```

```
4       static unordered_map<ll , ll> hs;
5       hs.clear();
6       ll cur = 1, t = sqrt(m) + 1;
7       for (int B = 1; B <= t; ++B)
8       {
9         (cur *= a) %= m;
10        hs[b * cur % m] = B; // 哈希表中存B的值
11      }
12      ll now = cur * k % m;
13      for (int A = 1; A <= t; ++A)
14      {
15        auto it = hs.find(now);
16        if (it != hs.end()) return A * t - it->second;
17        (now *= cur) %= m;
18      }
19      return -INF; // 这里因为要多次加1，要返回更小的负数
20    }
21    ll exBSGS(ll a, ll b, ll m, ll k = 1)
22    {
23      ll A = a %= m, B = b %= m, M = m;
24      if (b == 1) return 0;
25      ll cur = 1 % m;
26      for (int i = 0;; i++)
27      {
28        if (cur == B) return i;
29        cur = cur * A % M;
30        ll d = gcd(a, m);
31        if (b % d) return -INF;
32        if (d == 1) return BSGS(a, b, m, k * a % m) + i + 1;
33        k = k * a / d % m, b /= d, m /= d; // 相当于在递归求解
  exBSGS(a, b / d, m / d, k * a / d % m)
34      }
35    }
36
```

6.7 米勒拉宾素数检验

```
bool is_prime(ll x)
{
  if (x < 3) // 特判1, 2
  return x == 2;
  if (x % 2 == 0) // 特判偶数
  return false;
  ll A[] = {2, 325, 9375, 28178, 450775, 9780504,
1795265022}, d = x - 1, r = 0;
  while (d % 2 == 0) // 算出d, r
  d /= 2, ++r;
  // 或: r = __builtin_ctz(d), d >>= r;
  for (auto a : A)
  {
    ll v = qpow(a, d, x); // a^d
    // 如果a^d 0, 说明是a是x的倍数; 如果a^d 1或-1, 说明这串
数接下来一定都是1, 不用继续计算
    if (v <= 1 || v == x - 1)
    continue;
    for (int i = 0; i < r; ++i)
    {
      v = (__int128)v * v % x; // 同样使用__int128过渡

      if (v == x - 1 && i != r - 1) // 得到-1, 说明接下来都
是1, 可以退出了
      {
        v = 1;
        break;
      }
      // 在中途而非开头得到1, 却没有经过-1, 说明存在其他数字
y -1满足y^2 1, 则x一定不是奇素数
      if (v == 1)
      return false;
    }
    if (v != 1) // 查看是不是以1结尾
    return false;
  }
  return true;
```

```
}

```

6.8 rho

```
ll Pollard_Rho(ll N)
{
  if (N == 4)
  return 2;
  if (is_prime(N))
  return N;
  while (1)
  {
    ll c = randint(1, N - 1);
    auto f = [=](ll x) { return ((lll)x * x + c) % N; };
    ll t = 0, r = 0, p = 1, q;
    do
    {
      for (int i = 0; i < 128; ++i) // 令固定距离C=128
      {
        t = f(t), r = f(f(r));
        if (t == r || (q = (lll)p * abs(t - r) % N) == 0)
// 如果发现环，或者积即将为0，退出
          break;
        p = q;
      }
      ll d = gcd(p, N);
      if (d > 1)
      return d;
    } while (t != r);
  }
}

```

可以求所有素因数或者最大质因数

```
    void find_prime_factor(int n) {
      if (n == 1 || vis.count(n)) {
        return;
      }

      if (is_prime(n)) {
        prime.push_back(n);
        return;
      }

      vis[n] = 1;
      int x = Pollard_Rho(n);
      find_prime_factor(x);
      find_prime_factor(n / x);
    }
    unordered_map<ll , ll> um;
    ll max_prime_factor(ll x)
    {
      if (um.count(x))
      return um[x];
      ll fac = Pollard_Rho(x);
      if (fac == 1)
      um[x] = x;
      else
      um[x] = max(max_prime_factor(fac), max_prime_factor(x / fac));
      return um[x];
    }

```

6.9 康托展开

```
#include <bits/stdc++.h>
using namespace std;

#define int long long
```

```
5   const int mod = 998244353;
6   const int N = 1e6 + 1e2;
7
8   int tree[N], fac[N];
9
10  int lowbit(int x) {
11    return x & -x;
12  }
13  void update(int x, int k) {
14    while (x < N - 100) {
15      tree[x] += k;
16      x += lowbit(x);
17    }
18  }
19  int query(int x) {
20    int ans = 0;
21    while (x) {
22      ans += tree[x];
23      x -= lowbit(x);
24    }
25    return ans;
26  }
27  main() {
28    int n;
29    cin >> n;
30    fac[0] = 1;
31    for (int i = 1; i <= n; ++i) {
32      fac[i] = fac[i - 1] * i % mod;
33      update(i, 1);
34    }
35    int ans = 0;
36    for (int i = 1; i <= n; ++i) {
37      int x;
38      cin >> x;
39      ans = (ans + query(x - 1) * fac[n - i] % mod) % mod;
40      update(x, -1);
41    }
42    cout << ans + 1 << endl;
```

```
43      system("pause");
44    }
45
```

6.10 中国剩余定理 (CRT)

解

$$\begin{cases} x \equiv a_1 (\mathrm{mod} p_1) \\ x \equiv a_2 (\mathrm{mod} p_2) \\ \cdots\cdots \\ x \equiv a_k (\mathrm{mod} p_k) \end{cases}$$

其中 $m_1, m_2, \ldots, m_k$ 是两两互质的整数，中国剩余定理可以给出 x 的最小非负整数解。

```
1   #include <bits/stdc++.h>
2   using namespace std;
3
4   typedef long long LL;
5
6   LL exgcd(LL a, LL b, LL &x, LL &y) {
7     if (b == 0) {
8       x = 1;
9       y = 0;
10      return a;
11    }
12    int d = exgcd(b, a % b, y, x);
13    y -= (a / b) * x;
14    return d;
15  }
16  LL CRT(int k, LL *a, LL *p) {
17    LL mul = 1;
18    for (int i = 0; i < k; ++i) {
19      mul *= p[i];
20    }
21    LL ans = 0;
```

```
    for (int i = 0; i < k; ++i) {
      LL m = mul / p[i], x, y;
      exgcd(m, p[i], x, y);
      ans = (ans + a[i] * m * x % mul) % mul;
    }
    return (ans % mul + mul) % mul;
  }

```

6.11 扩展中国剩余定理 (exCRT)

同中国剩余定理，但无任何限制

(考不到吧)

```
#include <bits/stdc++.h>
using namespace std;

#define int long long
const int N = 1e5 + 1e3;

int mod[N], yu[N];

int lcm(int a, int b) {
  return a / __gcd(a, b) * b;
}

int qmul(int n, int k, int mod) {
  int ans = 0;
  while (k) {
    if (k % 2) {
      ans = (ans + n) % mod;
    }
    k /= 2;
    n = (n + n) % mod;
  }
  return ans % mod;
}
```

```
24
25    int exgcd(int a, int b, int &x, int &y) {
26      if (!b) {
27        x = 1;
28        y = 0;
29        return a;
30      }
31      int d = exgcd(b, a % b, y, x);
32      y -= (a / b) * x;
33      return d;
34    }
35
36    main() {
37      int n;
38      cin >> n;
39      for (int i = 0; i < n; ++i) {
40        cin >> mod[i] >> yu[i];
41      }
42      int ans = yu[0], M = mod[0], t, y;
43      for (int i = 1; i < n; ++i) {
44        int mi = mod[i];
45        int res = ((yu[i] - ans) % mi + mi) % mi;
46        int d = exgcd(M, mi, t, y);
47        if (res % d) {
48          cout << "-1\n";
49          exit(0);
50        }
51        t = qmul(t, res / d, mi);
52        ans += t * M;
53        M = lcm(M, mi);
54        ans = (ans % M + M) % M;
55      }
56      cout << ans << endl;
57      system("pause");
58    }
59
```

6.12 高斯消元

```cpp
#include <bits/stdc++.h>
using namespace std;

vector<valarray<double>> a;
const double eps = 1e-10;
int n;

template <typename it, typename unf, typename bif>
void line_elimin(it &arr, const unf &ck, const bif &op)
{
  for (auto &i : arr)
  {
    if (!ck(i))
    continue;
    for (auto &j : arr)
    if (&i != &j)
    op(i, j);
  }
}
int main()
{
  // int n;
  cin >> n;
  a = vector<valarray<double>>(n, valarray<double>(n + 1));
  for (int i = 0; i < n; ++i)
  {
    for (int j = 0; j < n + 1; ++j)
    {
      cin >> a[i][j];
    }
  }
  auto ck = [](const valarray<double> &x)
  {
    for (int i = 0; i < x.size() - 1; ++i)
    {
      if (abs(x[i]) > eps)
```

```
37            {
38              return 1;
39            }
40          }
41          return 0;
42        };
43        auto op = [](const valarray<double> &a, valarray<double> &
      b)
44        {
45          int i = 0;
46          while (abs(a[i]) < eps)
47          {
48            ++i;
49          }
50          valarray<double> x = b[i] * a / a[i];
51          b -= x;
52        };
53        line_elimin(a, ck, op);
54        for (int i = 0; i < n; ++i)
55        {
56          if (!ck(a[i]))
57          {
58            if (abs(a[i][a[i].size() - 1]) > eps) {
59              puts("-1");
60              return 0;
61            }
62          }
63        }
64        for (int i = 0; i < n; ++i)
65        {
66          if (!ck(a[i]))
67          {
68            if (abs(a[i][a[i].size() - 1]) <= eps)
69            {
70              puts("0");
71              return 0;
72            }
73          }
```

```
74        }
75        for (int i = 0; i < n; ++i)
76        {
77          int j = 0;
78          for (; j < n; ++j)
79          {
80            if (abs(a[j][i]) > eps)
81            {
82              break;
83            }
84          }
85          cout << "x" << i + 1 << "=" << fixed << setprecision(2)
    << a[j][n] / a[j][i] << endl;
86        }
87      }
88
```

6.13 线性基

```
1     bitset<N> B[N];
2     void insert(bitset<N> x)
3     {
4       for (int i = 0; i < N; ++i)
5       if (x[i])
6       x ^= B[i];
7       if (x == 0) return;
8       int msb = N - 1;
9       while (msb && x[msb] == 0) msb--;
10      B[msb] = x;
11      for (int i = msb + 1; i < N; ++i)
12      if (B[i][msb])
13      B[i] ^= x;
14    }
15
```

6.14 杜教筛

```cpp
#define int long long

const int N = 1e6 + 1e2;
using i64 = long long;

int prime[N];
vector<int> primeset;
i64 phi[N], sumphi[N];
i64 mu[N], summu[N];
void init() {
  phi[1] = mu[1] = 1;
  for (int i = 2; i <= N - 100; ++i) {
    if (!prime[i]) {
      phi[i] = i - 1;
      mu[i] = -1;
      primeset.push_back(i);
    }
    for (int u : primeset) {
      if (u * i > N - 100) {
        break;
      }
      prime[i * u] = 1;
      if (i % u == 0) {
        phi[i * u] = phi[i] * u;
        break;
      } else {
        mu[i * u] = mu[i] * mu[u];
        phi[i * u] = phi[i] * phi[u];
      }
    }
  }
  for (int i = 1; i <= N - 100; ++i) {
    //cout << phi[i] << endl;
    summu[i] = summu[i - 1] + mu[i];
    sumphi[i] = sumphi[i - 1] + phi[i];
  }
```

```
37    }
38    namespace SumPhi {
39      unordered_map<int, i64> mp;
40      i64 f(int n) {
41        i64 ans = n * (n + 1) / 2;
42        if (n <= N - 100) {
43          return sumphi[n];
44        }
45        if (mp.count(n)) {
46          return mp[n];
47        }
48        int l = 2, r = n;
49        while (l <= n) {
50          r = n / (n / l);
51          ans -= (r - l + 1) * f(n / l);
52          l = r + 1;
53        }
54        return mp[n] = ans;
55      }
56    }
57    unordered_map<int, i64> mp;
58    i64 f(int n) {
59      i64 ans = 1;
60      if (n <= N - 100) {
61        return summu[n];
62      }
63      if (mp.count(n)) {
64        return mp[n];
65      }
66      int l = 2, r = n;
67      while (l <= n) {
68        r = n / (n / l);
69        ans -= (r - l + 1) * f(n / l);
70        l = r + 1;
71      }
72      return mp[n] = ans;
73    }
74
```

6.15 一些式子

$\sum_{i=1}^{n} i^x = \frac{C(x+1,n+1)}{(n+1)!} + \frac{C(x,n+)}{}$

7 多项式

7.1 fft(快速傅里叶变换)

```
#include <bits/stdc++.h>
using namespace std;

const int N = 1e6 + 1e2;

typedef complex<double> comp;
comp a[N * 3], b[N * 3], tmp[N * 3], ans[N * 3];
int rev[N * 3];
const comp I(0, 1);
const double PI = M_PI;
void fft(comp *f, int len, int op = 1) {
  int bit = __lg(len);
  for (int i = 0; i < len; ++i) {
    rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (bit - 1));
    if (i < rev[i]) {
      swap(f[i], f[rev[i]]);
    }
  }
  for (int l = 1; l < len; l <<= 1) {
    comp step = exp(PI / l * op * I);
    for (int i = 0; i < len; i += 2 * l) {
      comp cur(1, 0);
      for (int k = i; k < i + l; ++k) {
        comp g = f[k], h = f[k + l] * cur;
        f[k] = g + h, f[k + l] = g - h;
        cur *= step;
```

```
        }
      }
    }
  }
  int main() {
    int n, m;
    cin >> n >> m;
    int len = 1 << __lg(n + m + 1) + 1;
    for (int i = 0; i <= n; ++i) {
      cin >> a[i];
    }
    for (int i = 0; i <= m; ++i) {
      cin >> b[i];
    }
    fft(a, len);
    fft(b, len);
    for (int i = 0; i <= len; ++i) {
      ans[i] = a[i] * b[i];
    }
    fft(ans, len, -1);
    for (int i = 0; i <= n + m; ++i) {
      cout << int(ans[i].real() / len + 0.1) << ' ';
    }
    cout << endl;
    system("pause");
  }

```

8 计算几何

8.1 极角排序

```
const double eps = 0;
#define x first
#define y second
typedef pair<double, double> p;
```

```
5   using i64 = long long;
6   vector<p> v;
7   double f(double x) {
8     if (abs(x) <= eps) {
9       return 0;
10    }
11    return x;
12  }
13  i64 cross(p &A, p &B) {
14    return A.x * B.y - A.y * B.x;
15  }
16  i64 dul(p &A, p &B) {
17    return A.x * B.x + A.y * B.y;
18  }
19  int f(p A) {
20    if (A.y > eps) {
21      if (A.x > eps) {
22        return 1;
23      }
24      return 2;
25    } else {
26      if (A.x > eps) {
27        return 4;
28      }
29      return 3;
30    }
31  }
32  bool cmp(p &A, p &B) {
33    return f(A) < f(B) || (f(A) == f(B) && cross(A, B) > 0);
34  }
35  void qsort() {
36    sort(v.begin(), v.end(), cmp);
37  }
38  int len(p &A) {
39    return A.x * A.x + A.y * A.y;
40  }
41  double cos(p &A, p &B) {
42    return dul(A, B) / len(A) / len(B);
```

```
43    }
44
```

8.2 求凸包

```
1     p operator- (p A, p B) {
2       return {A.x - B.x, A.y - B.y};
3     }
4     int cross(p A, p B) {
5       return A.x * B.y - A.y * B.x;
6     }
7     void f(vector<p> v, vector<p> &p) {
8       for (int i = 0; i < v.size(); ++i) {
9         while (p.size() >= 2 && cross(v[i] - p.back(), p[p.size
    () - 2] - p.back()) < 0) {
10          p.pop_back();
11        }
12        p.push_back(v[i]);
13      }
14    }
15
```

8.3 求凸包直径

```
1     const double pi = 3.141592653589793;
2     const double eps=1e-12;
3     int dcmp(double x){
4       return (fabs(x)<=eps)?0:(x<0?-1:1);
5     }
6     struct Point{
7       double x,y;
8       Point(double X=0,double Y=0){x=X,y=Y;}
9     };
10    struct Vector{
11      double x,y;
12      Vector(double X=0,double Y=0){x=X,y=Y;}
```

```
13      };
14      inline Vector operator-(Point x,Point y){// 点-点=向量
15        return Vector(x.x-y.x,x.y-y.y);
16      }
17      inline double cross(Vector x,Vector y){ // 向量叉积
18        return x.x*y.y-x.y*y.x;
19      }
20      inline double operator*(Vector x,Vector y){ // 向量叉积
21        return cross(x,y);
22      }
23      inline double len(Vector x){ // 向量模长
24        return sqrt(x.x*x.x+x.y*x.y);
25      }
26
27      int stk[N];
28      bool used[N];
29      vector<Point> ConvexHull(Point* poly, int n){ // Andrew算法
        求凸包
30        int top=0;
31        sort(poly+1,poly+n+1,[&](Point x,Point y){
32          return (x.x==y.x)?(x.y<y.y):(x.x<y.x);
33        });
34        for (int i = 0; i <= n; ++i) {
35          used[i] = 0;
36        }
37        stk[++top]=1;
38        for(int i=2;i<=n;i++){
39          while(top>1&&dcmp((poly[stk[top]]-poly[stk[top-1]])*(
          poly[i]-poly[stk[top]]))<=0){
40            used[stk[top--]]=0;
41          }
42          used[i]=1;
43          stk[++top]=i;
44        }
45        int tmp=top;
46        for(int i=n-1;i;i--){
47          if(used[i]) continue;
48          while(top>tmp&&dcmp((poly[stk[top]]-poly[stk[top-1]])*(
```

```
poly[i]-poly[stk[top]]))<=0){
49            used[stk[top--]]=0;
50          }
51          used[i]=1;
52          stk[++top]=i;
53        }
54        vector<Point> a;
55        for(int i=1;i<=top;i++){
56          a.push_back(poly[stk[i]]);
57        }
58        return a;
59      }
60
61      struct Line{
62        Point x;Vector y;
63        Line(Point X,Vector Y){x=X,y=Y;}
64        Line(Point X,Point Y){x=X,y=Y-X;}
65      };
66
67      inline double DistanceToLine(Point P,Line x){// 点到直线的距
    离
68        Vector v1=x.y, v2=P-x.x;
69        return fabs(cross(v1,v2))/len(v1);
70      }
71
72      double RoatingCalipers(vector<Point> poly){// 旋转卡壳
73        if(poly.size()==3) return len(poly[1]-poly[0]);
74        int cur=0;
75        double ans=0;
76        for(int i=0;i<poly.size()-1;i++){
77          Line line(poly[i],poly[i+1]);
78          while(DistanceToLine(poly[cur], line) <= DistanceToLine(
    poly[(cur+1)%poly.size()], line)){
79            cur=(cur+1)%poly.size();
80          }
81          ans=max(ans, max(len(poly[i]-poly[cur]), len(poly[i+1]-
    poly[cur])));
82        }
```

```
83      return ans;
84    }
85
```

9 字符串

9.1 KMP

9.1.1 MP

```
1     vector<int> pmt(p.length() + 1, 0);
2     for (int i = 1, j = 0; i < p.length(); ++i) {
3       while (j && p[i] != p[j]) {
4         j = pmt[j - 1];
5       }
6       if (p[i] == p[j]) {
7         ++j;
8       }
9       pmt[i] = j;
10    }
11    for (int i = 0, j = 0; i < s.length(); ++i) {
12      while (j && s[i] != p[j]) {
13        j = pmt[j - 1];
14      }
15      if (s[i] == p[j]) {
16        ++j;
17      }
18      if (j == p.length()) {
19        cout << i - j + 2 << endl;
20        j = pmt[j - 1];
21      }
22    }
23
```

9.1.2 KMP

pmt 含义改变了!

```
vector<int> pmt(p.length() + 1, 0);
for (int i = 1, j = 0; i < p.length(); ++i) {
  while (j && p[i] != p[j]) {
    j = pmt[j - 1];
  }
  if (p[i] == p[j]) {
    ++j;
    if (p[i + 1] == p[j + 1]) {
      pmt[i] = pmt[j];
      continue;
    }
  }
  pmt[i] = j;
}
for (int i = 0, j = 0; i < s.length(); ++i) {
  while (j && s[i] != p[j]) {
    j = pmt[j - 1];
  }
  if (s[i] == p[j]) {
    ++j;
  }
  if (j == p.length()) {
    cout << i - j + 2 << endl;
    j = pmt[j - 1];
  }
}

```

9.2 exKMP

```
vector<int> get_Z(const string &s) {
  vector<int> z(s.length(), 0);
  z[0] = s.length();
  int l = 0, r = 0;
```

```
5       for (int i = 1; i < s.length(); ++i) {
6         if (i > r || i + z[i - l] > r) {
7           z[i] = max(0, r - i + 1);
8           while (i + z[i] < s.length() && s[z[i]] == s[i + z[i
    ]]) {
9             z[i]++;
10          }
11          l = i, r = i + z[i] - 1;
12          continue;
13        }
14        z[i] = z[i - l];
15      }
16      return z;
17    }
18
```