Platelet

Team Reference Material

凌皓煜 | 陈彤 | 顾逸

Contents

Chapter 1

Graph Theory

Chapter 2

Math

Chapter 3

Geometry

Chapter 4

String

Chapter 5

Data Structure

5.1 莫队

```
//
//  Title: Modui
//  Date: 26.02.2016
//  Test:BZOJ-2038
//  Complexity:O(n^3/2)
//
/*
        莫队算法——将所有询问储存起来，然后分块暴力处理。
        时间复杂度为 O（n× 根号 n）。
*/
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <cmath>

#ifdef WIN32
        #define LL "%I64d"
#else
        #define LL "%lld"
#endif

#ifdef CT
        #define debug(...) printf(__VA_ARGS__)
#else
        #define debug(...)
#endif

#define R register
#define getc() (S==T&&(T=(S=B)+fread(B,1,1<<15,stdin),S==T)?EOF:*S++)
```

```
#define gmax(_a, _b) ((_a) > (_b) ? (_a) : (_b))
#define gmin(_a, _b) ((_a) < (_b) ? (_a) : (_b))
#define cmax(_a, _b) (_a < (_b) ? _a = (_b) : 0)
#define cmin(_a, _b) (_a > (_b) ? _a = (_b) : 0)
char B[1<<15],*S=B,*T=B;
inline int FastIn()
{
        R char ch;R int cnt=0;R bool minus=0;
        while (ch=getc(),(ch < '0' || ch > '9') && ch != '-') ;
        ch == '-' ?minus=1:cnt=ch-'0';
        while (ch=getc(),ch >= '0' && ch <= '9') cnt = cnt * 10 + ch - '0';
        return minus?-cnt:cnt;
}
#define maxn 50010
int col[maxn],num[maxn],size,pos[maxn];
long long up[maxn],dw[maxn],ans;
struct Query{
        int l,r,id;
}q[maxn];
inline bool cmp(const Query &i,const Query &j){
        return pos[i.l]!=pos[j.l] ? (i.l<j.l) : (pos[i.l]&1 ? i.r<j.r : i.r>j.r);
}
inline long long gcd(R long long a,R long long b){
        R long long tmp;
        while (b){
                tmp=b;
                b=a%b;
                a=tmp;
        }
        return a;
}
inline void update(R int x,R int d){
        ans-=num[col[x]]*num[col[x]];
        num[col[x]]+=d;
        ans+=num[col[x]]*num[col[x]];
}
int main()
{
        R int n=FastIn(),m=FastIn();size=(int)sqrt(n*1.0);
        for (R int i=1;i<=n;i++) col[i]=FastIn(),pos[i]=(i-1)/size+1;
        for (R int i=1;i<=m;i++){
                q[i].l=FastIn();q[i].r=FastIn();q[i].id=i;
        }
        std::sort(q+1,q+m+1,cmp);
        R int l=1,r=0;
        for (R int i=1;i<=m;i++){
```

```
                R int id_now=q[i].id;
                if (q[i].l==q[i].r){
                        up[id_now]=0;dw[id_now]=1;continue;
                }
                for (;r<q[i].r;r++) update(r+1,1);
                for (;r>q[i].r;r--) update(r,-1);
                for (;l<q[i].l;l++) update(l,-1);
                for (;l>q[i].l;l--) update(l-1,1);
                R long long aa,bb,cc;
                aa=ans-q[i].r+q[i].l-1;
                bb=(long long)(q[i].r-q[i].l+1)*(q[i].r-q[i].l);
                cc=gcd(aa,bb);aa/=cc;bb/=cc;
                up[id_now]=aa;dw[id_now]=bb;
        }
        for (R int i=1;i<=m;i++) printf("%lld/%lld\n",up[i],dw[i] );
        return 0;
}
```

5.2 ST 表

```
#include <cstdio>

#define dmax(_a, _b) ((_a) > (_b) ? (_a) : (_b))

#define maxn 200010
int a[maxn], f[20][maxn], n;
int Log[maxn];

void build()
{
        for (int i = 1; i <= n; ++i) f[0][i] = a[i];

        int lim = Log[n];
        for (int j = 1; j <= lim; ++j)
        {
                int *fj = f[j], *fj1 = f[j - 1];
                for (int i = 1; i <= n - (1 << j) + 1; ++i)
                        fj[i] = dmax(fj1[i], fj1[i + (1 << (j - 1))]);
        }
}
int Query(int l, int r)
{
        int k = Log[r - l + 1];
        return dmax(f[k][l], f[k][r - (1 << k) + 1]);
}
int main()
```

```
{
        scanf("%d", &n);
        Log[0] = -1;
        for (int i = 1; i <= n; ++i)
        {
                scanf("%d", &a[i]);
                Log[i] = Log[i >> 1] + 1;
        }
        build();
        int q;
        scanf("%d", &q);
        for (; q; --q)
        {
                int l, r; scanf("%d%d", &l, &r);
                printf("%d\n", Query(l, r) );
        }
}
```

5.3 可并堆

```
struct Node {
        Node *ch[2];
        ll val; int size;
        inline void update()
        {
                size = ch[0] -> size + ch[1] -> size + 1;
        }
} mem[maxn], *rt[maxn];
Node *merge(Node *a, Node *b)
{
        if (a == mem) return b;
        if (b == mem) return a;
        if (a -> val < b -> val) std::swap(a, b);
        std::swap(a -> ch[0], a -> ch[1]);
        a -> ch[1] = merge(a -> ch[1], b);
        a -> update();
        return a;
}
```

5.4 线段树

5.4.1 ZKW 线段树

```
//
//  Title:ZKW Segment Tree
```

```
//  Date:19.11.2015
//  Complexity:
//    Build Tree:O(N)
//    Query:O(logN)
//    Change:O(logN)

#include<cstdio>
#include<cmath>
#define maxn 100000
#define R register
int T[1<<18|1],n,m,M;

inline int FastIn()
{
        R char ch=getchar();R int cnt=0;R bool minus=0;
        while ((ch<'0'||ch>'9')&&ch!='-') ch=getchar();
        if (ch=='-') minus=1,ch=getchar();
        while (ch>='0'&&ch<='9') cnt=cnt*10+ch-'0',ch=getchar();
        return minus?-cnt:cnt;
}

inline void Build_Tree()
{
        for (R int i=M-1;i>=1;i--)
          T[i]=T[2*i]+T[2*i+1];
}

inline int Query(int s,int t)
{
    R int Ans;
        for (Ans=0,s=s+M-1,t=t+M+1;s^t^1;s>>=1,t>>=1)
        {
                if (~s&1) Ans+=T[s^1];
                if (t&1) Ans+=T[t^1];
        }
        return Ans;
}

inline void Change(int x,int NewValue)
{
        R int i=M+x;
        for (T[i]=NewValue,i>>=1;i;i>>=1)
          T[i]=T[2*i]+T[2*i+1];
}

int main()
```

```
{
        n=FastIn();m=FastIn();
        for (M=1;M<=n;M<<=1);
        for (R int i=0;i<n;i++)
          T[M+i]=FastIn();
        Build_Tree();
        for (R int i=1;i<=m;i++)
        {
                R char cmd=getchar();
                if (cmd=='Q')
                {
                        R int a=FastIn()-1,b=FastIn()-1;
                        printf("%d\n",Query(a,b));
                }
                if (cmd=='M')
                {
                        R int a=FastIn()-1,b=FastIn();
                        Change(a,b);
                }
        }
        return 0;
}
```

5.4.2 主席树

```
//
//  Title: Functional Segment Tree
//  Date:16.12.2015
//  Complexity:O((n+m)logn)
//  Test:YZOJ-1991
#include<cstdio>
#include<algorithm>
#define maxt 2000010
#define maxn 100010
#define R register
inline int FastIn(){
        R char ch=getchar();R int cnt=0;
        while (ch<'0'||ch>'9') ch=getchar();
        while (ch>='0'&&ch<='9') cnt=cnt*10+ch-'0',ch=getchar();
        return cnt;
}

int ls[maxt],
    rs[maxt],
        count[maxt],
        root[maxn],
```

```
22          tot;
23
24  int num[maxn],rank[maxn],n,m,r[maxn];
25
26  bool cmp(const int &i,const int &j){
27          return num[i]<num[j];
28  }
29
30  inline void Insert(int last,int left,int right,int pre)
31  {
32          count[++tot]=count[last]+1;
33          if (left==right) return;
34          R int mid=(left+right)>>1;
35          if (pre>mid){
36                  rs[tot]=tot+1;
37                  Insert(rs[last],mid+1,right,pre);
38          }
39          else{
40                  ls[tot]=tot+1;
41                  rs[tot]=rs[last];
42                  Insert(ls[last],left,mid,pre);
43          }
44  }
45
46  inline int Query(int a,int b,int k)
47  {
48          R int l=1,r=n,mid,f1=a,f2=b,cnt,kk=k;
49          while (l<r){
50                  mid=(l+r)>>1;cnt=count[ls[f2]]-count[ls[f1]];
51                  if (cnt>=kk) f1=ls[f1],f2=ls[f2],r=mid;
52                  else f1=rs[f1],f2=rs[f2],l=mid+1,kk-=cnt;
53          }
54          return l;
55  }
56
57  int main()
58  {
59          n=FastIn();m=FastIn();R int i,a,b,k;
60          for (i=1;i<=n;i++) num[i]=FastIn(),rank[i]=i;
61          std::sort(rank+1,rank+n+1,cmp);
62          std::sort(num+1,num+n+1);
63          for (i=1;i<=n;i++) r[rank[i]]=i;
64          for (i=1;i<=n;i++) {
65                  root[i]=tot+1;
66                  Insert(root[i-1],1,n,r[i]);
67          }
```

```
68          for (i=1;i<=m;i++){
69                  a=FastIn();b=FastIn();k=FastIn();
70                  printf("%d\n",num[Query(root[a-1],root[b],k)]);
71          }
72          return 0;
73  }
```

5.5 平衡树

5.5.1 Splay

```
1   //
2   // Title : Splay Tree
3   // Date : 11.01.2016
4   // Complexity : O(nlogn) (期望)
5   // Test : BZOJ-1251
6   /*
7   */
8   #include <cstdio>
9   #include <cstring>
10  #include <algorithm>
11  #include <cmath>
12
13  #ifdef WIN32
14          #define LL "%I64d"
15  #else
16          #define LL "%lld"
17  #endif
18
19  #ifdef CT
20          #define debug(...) printf(__VA_ARGS__)
21  #else
22          #define debug(...)
23  #endif
24
25  #define R register
26  #define getc() (S==T&&(T=(S=B)+fread(B,1,1<<15,stdin),S==T)?EOF:*S++)
27  #define gmax(_a, _b) ((_a) > (_b) ? (_a) : (_b))
28  #define gmin(_a, _b) ((_a) < (_b) ? (_a) : (_b))
29  #define cmax(_a, _b) (_a < (_b) ? _a = (_b) : 0)
30  #define cmin(_a, _b) (_a > (_b) ? _a = (_b) : 0)
31  char B[1<<15],*S=B,*T=B;
32  inline int FastIn()
33  {
34          R char ch;R int cnt=0;R bool minus=0;
35          while (ch=getc(),(ch < '0' || ch > '9') && ch != '-') ;
```

```
        ch == '-' ?minus=1:cnt=ch-'0';
        while (ch=getc(),ch >= '0' && ch <= '9') cnt = cnt * 10 + ch - '0';
        return minus?-cnt:cnt;
}
#define maxn 50010
int n,Q,root;
int fa[maxn],ch[maxn][2],id[maxn],size[maxn];
int tag[maxn],mx[maxn],num[maxn];
bool rev[maxn];
inline void update(int x){
        R int ls=ch[x][0],rs=ch[x][1];
        mx[x]=num[x];
        cmax(mx[x],mx[ls]);cmax(mx[x],mx[rs]);
        size[x]=size[ls]+size[rs]+1;
}//更新
void build(int l,int r,int rt){
        if (l>r) return ;
        R int mid=l+r>>1;
        fa[mid]=rt;
        if (mid<rt) ch[rt][0]=mid;
        else ch[rt][1]=mid;
        build(l,mid-1,mid);
        build(mid+1,r,mid);
        update(mid);
}//建树
inline void pushdown(int x){
        R int ls=ch[x][0],rs=ch[x][1];
        if (tag[x]){
                R int lazy=tag[x];
                if (ls) tag[ls]+=lazy,num[ls]+=lazy,mx[ls]+=lazy;
                if (rs) tag[rs]+=lazy,num[rs]+=lazy,mx[rs]+=lazy;
                tag[x]=0;
        }
        if (rev[x]){
                if (ls) rev[ls]^=1;
                if (rs) rev[rs]^=1;
                ch[x][1]=ls;ch[x][0]=rs;
                rev[x]=0;
        }
}//具体下传的过程
inline void rotate(int x){//把 x 向上旋转到 x 的父亲
        R int f=fa[x],gf=fa[f],d=(ch[f][1]==x);//f 表示 x 的父亲，gf 是祖父，d 是 x 在其父亲的位置
        if (f==root) root=x,ch[0][0]=x;
        (ch[f][d]=ch[x][d^1])>0 ? fa[ch[f][d]]=f : 0;//把 x 的儿子中与 d 相反的节点来代替 x 的位置
        (fa[x]=gf)>0 ? ch[gf][ch[gf][1]==f]=x : 0;//把 x 代替 f 的位置
        fa[ch[x][d^1]=f]=x;//把 f 接到 x 的下面
```

```
82          update(f);//更新 f 节点
83  }
84  inline void splay(int x,int rt){//把 x 旋转到 rt
85          while (fa[x]!=rt){
86                  R int f=fa[x],gf=fa[f];
87                  if (gf!=rt) rotate((ch[gf][1]==f)^(ch[f][1]==x) ? x :f);//如果祖孙三代是相同方向
88                  rotate(x);
89          }
90          update(x);
91  }
92  int find(int x,int rank){
93          if (tag[x]||rev[x]) pushdown(x);
94          R int ls=ch[x][0],rs=ch[x][1],lsize=size[ls];
95          if (lsize+1==rank) return x;
96          if (lsize>=rank) return find(ls,rank);
97          else return find(rs,rank-lsize-1);
98  }//找第 k 小
99  inline int prepare(int l,int r){
100         R int x=find(root,l-1);
101         splay(x,0);
102         x=find(root,r+1);
103         splay(x,root);
104         return ch[x][0];
105 }//把 l-1 旋到根，r+1 旋到右儿子，然后返回 r+1 的左儿子，返回一个包含 [l，r] 的节点
106 inline void add(int l,int r,int w){
107         R int x=prepare(l,r);
108         tag[x]+=w,num[x]+=w,mx[x]+=w;
109 }//区间加
110 inline void rever(int l,int r){
111         R int x=prepare(l,r);
112         rev[x]^=1;
113 }//区间翻转
114 inline void query(int l,int r){
115         R int x=prepare(l,r);
116         printf("%d\n",mx[x] );
117 }//区间查询最大值
118 inline int split(R int k){
119         R int ls;
120         if (k<size[root])
121         {
122                 R int kth=find(root,k+1);
123                 splay(kth);ls=ch[kth][0];
124                 fa[ls]=0;ch[kth][0]=0;
125                 size[kth]-=size[ls];
126         }
127         else{
```

```
128                 ls=root;root=0;
129         }
130         return ls;
131 }//删除数列
132 inline void merge(R int nwrt){
133         if (!root) {root=nwrt;return;}
134         R int nw=find(root,1);
135         splay(nw);fa[nwrt]=nw;ch[nw][0]=nwrt;
136         size[nw]+=size[nwrt];
137 }//合并数列
138 int main()
139 {
140         n=FastIn()+2;Q=FastIn();R int i,l,r,v,cmd;mx[0]=-23333333;
141         build(1,n,0);root=(1+n)>>1;
142         for (;Q--;){
143                 cmd=FastIn();l=FastIn()+1;r=FastIn()+1;
144                 if (cmd==1) v=FastIn(),add(l,r,v);
145                 else if (cmd==2) rever(l,r);
146                 else query(l,r);
147         }
148         return 0;
149 }
```

5.5.2 非旋转 Treap

```
1  //
2  //  Title : Treap (unrotated)
3  //  Date : 13.04.2016
4  //  Test : BZOJ-3224
5  //  Complexity : O(nlogn)(期望)
6  //
7  /*
8          对于序列上的一些操作的问题——
9          解决办法: 平衡树 Treap
10 */
11 #include <cstdio>
12 #include <cstring>
13 #include <algorithm>
14 #include <cmath>
15
16 #ifdef WIN32
17         #define LL "%I64d"
18 #else
19         #define LL "%lld"
20 #endif
21
```

```
#ifdef CT
        #define debug(...) printf(__VA_ARGS__)
        #define setfile()
#else
        #define debug(...)
        #define filename ""
        #define setfile() freopen(filename".in", "r", stdin); freopen(filename".out", "w", stdo
#endif

#define R register
#define getc() (S == T && (T = (S = B) + fread(B, 1, 1 << 15, stdin), S == T) ? EOF : *S++)
#define dmax(_a, _b) ((_a) > (_b) ? (_a) : (_b))
#define dmin(_a, _b) ((_a) < (_b) ? (_a) : (_b))
#define cmax(_a, _b) (_a < (_b) ? _a = (_b) : 0)
#define cmin(_a, _b) (_a > (_b) ? _a = (_b) : 0)
char B[1 << 15], *S = B, *T = B;
inline int FastIn()
{
        R char ch; R int cnt = 0; R bool minus = 0;
        while (ch = getc(), (ch < '0' || ch > '9') && ch != '-') ;
        ch == '-' ? minus = 1 : cnt = ch - '0';
        while (ch = getc(), ch >= '0' && ch <= '9') cnt = cnt * 10 + ch - '0';
        return minus ? -cnt : cnt;
}
const int Ta = 1 << 16 | 3, Tb = 33333331;
int Tc;
inline int randint() {return Tc = Ta * Tc + Tb;}
struct Treap
{
        int data, key, size;
        Treap *ls, *rs;
        Treap(int _val):data(_val), key(randint()), ls(NULL), rs(NULL), size(1){}
        inline void update()
        {
                size = (ls ? ls -> size : 0) + (rs ? rs -> size : 0) + 1;
        }
}*root;
inline int Size(Treap *x)
{
        return x ? x -> size : 0;
}
//为了防止访问到空节点，定义一个函数来访问 size
struct Pair
{
        Treap *fir, *sec;
};
```

```
Treap *Merge(Treap *a, Treap *b)
{
        if (!a) return b;
        if (!b) return a;
        if (a -> key < b -> key)
        {
                a -> rs = Merge(a -> rs, b);
                a -> update();
                return a;
        }
        else
        {
                b -> ls = Merge(a, b -> ls);
                b -> update();
                return b;
        }
}
//按照 a, b 的顺序来合并两棵 Treap
Pair Split(Treap *x, int k)
{
        if (!x) return (Pair){NULL, NULL};
        Pair y; y.fir = NULL; y.sec = NULL;
        if (Size(x -> ls) >= k)
        {
                y = Split(x -> ls, k);
                x -> ls = y.sec;
                x -> update();
                y.sec = x;
        }
        else
        {
                y = Split(x -> rs, k - Size(x -> ls) - 1);
                x -> rs = y.fir;
                x -> update();
                y.fir = x;
        }
        return y;
}
//将前 k 个的点分离出来
inline int Find(R int k)
{
        Pair x = Split(root, k - 1);
        Pair y = Split(x.sec, 1);
        Treap *ans = y.fir;
        root = Merge(Merge(x.fir, ans), y.sec);
        return ans -> data;
```

```
}
//找到第 k 小的 data 值
int Get(Treap *x, R int val)
{
        if (!x) return 0;
        return val < x -> data ? Get(x -> ls, val) : Get(x -> rs, val) + Size(x -> ls) + 1;
}
//找到 val 的排名
inline void Insert(R int val)
{
        R int k = Get(root, val);
        Pair x = Split(root, k);
        Treap *pre = new Treap(val);
        root = Merge(Merge(x.fir, pre), x.sec);
}
//插入
inline void Delete(R int val)
{
        R int k = Get(root, val);
        Pair x = Split(root, k - 1);
        Pair y = Split(x.sec, 1);
        root = Merge(x.fir, y.sec);
}
//单点删除
inline int upper(R int val)
{
        R int ans = 1e9;
        Treap *tmp = root;
        while (tmp)
        {
                if (tmp -> data > val)
                {
                        cmin(ans, tmp -> data);
                        tmp = tmp -> ls;
                }
                else
                        tmp = tmp -> rs;
        }
        return ans;
}
inline int lower(R int val)
{
        R int ans = -1e9;
        Treap *tmp = root;
        while (tmp)
        {
```

```
160                 if (tmp -> data < val)
161                 {
162                         cmax(ans, tmp -> data);
163                         tmp = tmp -> rs;
164                 }
165                 else tmp = tmp -> ls;
166         }
167         return ans;
168 }
169 void print(Treap *x)
170 {
171         if (!x) return;
172         print(x -> ls);
173         printf("%d ",x -> data );
174         print(x -> rs);
175 }
176 int main()
177 {
178         root = NULL;
179         for (R int Q = FastIn(); Q; --Q)
180         {
181                 R int opt = FastIn(), x = FastIn();
182                 if (opt == 1) Insert(x);
183                 else if (opt == 2) Delete(x);
184                 else if (opt == 3)
185                 {
186                         R int ans = Get(root, x);
187                         while (ans > 1 && Find(ans - 1) == x) ans--;
188                         printf("%d\n", ans );
189                 }
190                 else if (opt == 4) printf("%d\n", Find(x) );
191                 else if (opt == 5) printf("%d\n",lower(x) );
192                 else printf("%d\n",upper(x) );
193         }
194         return 0;
195 }
196 /*
197 input:
198 10
199 1 106465
200 4 1
201 1 317721
202 1 460929
203 1 644985
204 1 84185
205 1 89851
```

```
6 81968
1 492737
5 493598

output:
106465
84185
492737

input2:
5
1 1
1 1
1 1
1 2
3 1
output2:
1
*/
```

5.5.3 可持久化平衡树

```
//
//  Title: Functional Treap
//  Date: 16.04.2016
//  Test:YZOJ-1620
//  Complexity:O(nlogn)(期望)
//
/*
    可持久化 Treap:
        用来解决超级编辑器等问题。
        优势: 好写好调好理解的平衡树
        缺点: 写不好看的话常数大。(相较于 SBT 来说, 甚至有可能会比 splay 慢), 需手写 rand
*/
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <cmath>

#ifdef WIN32
        #define LL "%I64d"
#else
        #define LL "%lld"
#endif

#ifdef CT
```

```
25          #define debug(...) printf(__VA_ARGS__)
26          #define setfile()
27  #else
28          #define debug(...)
29          #define filename ""
30          #define setfile() freopen(filename".in", 'r', stdin); freopen(filename".out", 'w', stdout)
31  #endif
32
33  #define R register
34  //#define getc() (S==T&&(T=(S=B)+fread(B,1,1<<15,stdin),S==T)?EOF:*S++)
35  #define getc() getchar()
36  #define dmax(_a, _b) ((_a) > (_b) ? (_a) : (_b))
37  #define dmin(_a, _b) ((_a) < (_b) ? (_a) : (_b))
38  #define cmax(_a, _b) (_a < (_b) ? _a = (_b) : 0)
39  #define cmin(_a, _b) (_a > (_b) ? _a = (_b) : 0)
40  #define cabs(_x) ((_x)<0?(-_x):(_x))
41  char B[1<<15],*S=B,*T=B;
42  inline int FastIn()
43  {
44          R char ch;R int cnt=0;R bool minus=0;
45          while (ch=getc(),(ch < '0' || ch > '9') && ch != '-') ;
46          ch == '-' ?minus=1:cnt=ch-'0';
47          while (ch=getc(),ch >= '0' && ch <= '9') cnt = cnt * 10 + ch - '0';
48          return minus?-cnt:cnt;
49  }
50  #define maxn 100010
51  char str[maxn];
52  struct Treap
53  {
54          char data;
55          int size;
56          Treap *ls, *rs;
57          Treap(char _ch): data(_ch), size(1), ls(NULL), rs(NULL){}
58          inline void update()
59          {
60                  size = (ls ? ls -> size : 0) + (rs ? rs -> size : 0) + 1;
61          }
62  }*root[maxn];
63  inline int Size(Treap *x)
64  {
65          return x ? x -> size : 0;
66  }
67  struct Pair
68  {
69          Treap *fir, *sec;
70  };
```

```
71  inline Treap *copy(Treap *x)
72  {
73          if (!x) return NULL;
74          Treap *nw = new Treap(x -> data);
75          nw -> ls = x -> ls;
76          nw -> rs = x -> rs;
77          nw -> size = x -> size;
78          return nw;
79  }
80  Pair Split(Treap *x, int k)
81  {
82          if (!x) return (Pair){NULL, NULL};
83          Pair y; y.fir = NULL; y.sec = NULL;
84          Treap *nw = copy(x);
85          if (Size(nw -> ls) >= k)
86          {
87                  y = Split(nw -> ls, k);
88                  nw -> ls = y.sec;
89                  nw -> update();
90                  y.sec = nw;
91          }
92          else
93          {
94                  y = Split(nw -> rs, k - Size(nw -> ls) - 1);
95                  nw -> rs = y.fir;
96                  nw -> update();
97                  y.fir = nw;
98          }
99          return y;
100 }
101 const int Ta = 1 << 16 | 3, Tb = 33333331;
102 unsigned int Tc;
103 inline unsigned int randint(){return Tc = Ta * Tc + Tb;}
104 Treap *Merge(Treap *a, Treap *b)
105 {
106         Treap *nw;
107         if (!a) return nw = copy(b);
108         if (!b) return nw = copy(a);
109         if (randint() % (Size(a) + Size(b)) < Size(a))
110         {
111                 nw = copy(a);
112                 nw -> rs = Merge(nw -> rs, b);
113         }
114         else
115         {
116                 nw = copy(b);
```

```
117                 nw -> ls = Merge(a, nw -> ls);
118         }
119         nw -> update();
120         return nw;
121 }
122 Treap *Build(int l, int r)
123 {
124         if (l > r) return NULL;
125         R int mid = l + r >> 1;
126         Treap *nw = new Treap(str[mid]);
127         nw -> ls = Build(l, mid - 1);
128         nw -> rs = Build(mid + 1, r);
129         nw -> update();
130         return nw;
131 }
132 int now;
133 inline void Insert(R int k, R char ch)
134 {
135         Pair x = Split(root[now], k);
136         Treap *nw = new Treap(ch);
137         root[++now] = Merge(Merge(x.fir, nw), x.sec);
138 }
139 inline void Del(R int l, R int r)
140 {
141         Pair x = Split(root[now], l - 1);
142         Pair y = Split(x.sec, r - l + 1);
143         root[++now] = Merge(x.fir, y.sec);
144 }
145 inline void Copy(R int l, R int r, R int ll)
146 {
147         Pair x = Split(root[now], l - 1);
148         Pair y = Split(x.sec, r - l + 1);
149         Pair z = Split(root[now], ll);
150         Treap *ans = y.fir;
151         root[++now] = Merge(Merge(z.fir, ans), z.sec);
152 }
153 inline void Print(Treap *x, R int l, R int r)
154 {
155         if (!x) return ;
156         if (l > r) return;
157         R int mid = Size(x -> ls) + 1;
158         if (r < mid)
159         {
160                 Print(x -> ls, l, r);
161                 return ;
162         }
```

```
163         if (l > mid)
164         {
165                 Print(x -> rs, l - mid, r - mid);
166                 return ;
167         }
168         Print(x -> ls, l, mid - 1);
169         printf("%c",x -> data );
170         Print(x -> rs, 1, r - mid);
171 }
172 inline void Printtree(Treap *x)
173 {
174         if (!x) return;
175         Printtree(x -> ls);
176         printf("%c",x -> data );
177         Printtree(x -> rs);
178 }
179 int main()
180 {
181 //        setfile();
182         R int n = FastIn();
183         gets(str + 1);
184         R int len = strlen(str + 1);
185         root[0] = Build(1, len);
186         while (1)
187         {
188                 R char opt = getc();
189                 while (opt < 'A' || opt > 'Z')
190                 {
191                         if (opt == EOF) return 0;
192                         opt = getc();
193                 }
194                 if (opt == 'I')
195                 {
196                         R int x = FastIn();
197                         R char ch = getc();
198                         Insert(x, ch);
199                 }
200                 else if (opt == 'D')
201                 {
202                         R int l = FastIn(), r = FastIn();
203                         Del(l, r);
204                 }
205                 else if (opt == 'C')
206                 {
207                         R int x = FastIn(), y = FastIn(), z = FastIn();
208                         Copy(x, y, z);
```

```
209                 }
210                 else if (opt == 'P')
211                 {
212                         R int x = FastIn(), y = FastIn(), z = FastIn();
213 //                        printf("%d %d %d\n",x, y, z );
214                         Print(root[now - x], y, z);
215                         puts("");
216                 }
217 //                Printtree(root[now]);
218 //                puts("");
219         }
220         return 0;
221 }
```

5.6 CDQ 分治

```
1  //
2  //  Title : cdq 分治
3  //  Date : 18.04.2016
4  //  Test : BZOJ-1176
5  //  Complexity : O(nlog^2n)
6  //
7  /*
8          对于三维偏序等问题——
9          解决办法：离线询问，分治降维，剩下一维用随便什么树乱搞。这样就不用写树套树啦!
10 */
11 #include <cstdio>
12 #include <cstring>
13 #include <algorithm>
14 #include <cmath>
15
16 #ifdef WIN32
17         #define LL "%I64d"
18 #else
19         #define LL "%lld"
20 #endif
21
22 #ifdef CT
23         #define debug(...) printf(__VA_ARGS__)
24         #define setfile()
25 #else
26         #define debug(...)
27         #define filename ""
28         #define setfile() freopen(filename".in", "r", stdin); freopen(filename".out", "w", stdout);
29 #endif
30
```

```
#define R register
#define getc() (S == T && (T = (S = B) + fread(B, 1, 1 << 15, stdin), S == T) ? EOF : *S++)
#define dmax(_a, _b) ((_a) > (_b) ? (_a) : (_b))
#define dmin(_a, _b) ((_a) < (_b) ? (_a) : (_b))
#define cmax(_a, _b) (_a < (_b) ? _a = (_b) : 0)
#define cmin(_a, _b) (_a > (_b) ? _a = (_b) : 0)
char B[1 << 15], *S = B, *T = B;
inline int FastIn()
{
        R char ch; R int cnt = 0; R bool minus = 0;
        while (ch = getc(), (ch < '0' || ch > '9') && ch != '-') ;
        ch == '-' ? minus = 1 : cnt = ch - '0';
        while (ch = getc(), ch >= '0' && ch <= '9') cnt = cnt * 10 + ch - '0';
        return minus ? -cnt : cnt;
}
#define maxn 200010
#define maxm 2000010
struct event
{
        int x, y, pos, opet, ans;
        inline bool operator < (const event &that) const {return pos < that.pos ;}
}t[maxn], q[maxn];
#define lowbit(_x) ((_x) & -(_x))
int bit[maxm], last[maxm], s, w, cnt, now;
inline void add(R int x, R int val)
{
        for (; x <= w; x += lowbit(x))
        {
                if (last[x] != now)
                        bit[x] = 0;
                bit[x] += val;
                last[x] = now;
        }
}
inline int query(R int x)
{
        R int ans = 0;
        for (; x ; x -= lowbit(x))
        {
                if (last[x] == now)
                        ans += bit[x];
        }
        return ans;
}
void cdq(R int left, R int right)
{
```

```
77          if (left == right) return ;
78          R int mid = left + right >> 1;
79          cdq(left, mid); cdq(mid + 1, right);
80          //分成若干个子问题
81          ++now;
82          for (R int i = left, j = mid + 1; j <= right; ++j)
83          {
84                  for (; i <= mid && q[i].x <= q[j].x; ++i)
85                          if (!q[i].opet)
86                                  add(q[i].y, q[i].ans);
87                  //考虑前面的修改操作对后面的询问的影响
88                  if (q[j].opet)
89                          q[j].ans += query(q[j].y);
90          }
91          R int i, j, k = 0;
92          //以下相当于归并排序
93          for (i = left, j = mid + 1; i <= mid && j <= right; )
94          {
95                  if (q[i].x <= q[j].x)
96                          t[k++] = q[i++];
97                  else
98                          t[k++] = q[j++];
99          }
100         for (; i <= mid; )
101                 t[k++] = q[i++];
102         for (; j <= right; )
103                 t[k++] = q[j++];
104         for (R int i = 0; i < k; ++i)
105                 q[left + i] = t[i];
106 }
107 int main()
108 {
109 //        setfile();
110         s = FastIn();
111         w = FastIn();
112         while (1)
113         {
114                 R int opt = FastIn();
115                 if (opt == 1)
116                 {
117                         R int x = FastIn(), y = FastIn(), a = FastIn();
118                         q[++cnt] = (event){x, y, cnt, 0, a};
119                 }
120                 if (opt == 2)
121                 {
122                         R int x = FastIn() - 1, y = FastIn() - 1, a = FastIn(), b = FastIn();
```

```
123                         q[++cnt] = (event) {x, y, cnt, 1, x * y * s};
124                         q[++cnt] = (event) {a, b, cnt, 2, a * b * s};
125                         q[++cnt] = (event) {x, b, cnt, 2, x * b * s};
126                         q[++cnt] = (event) {a, y, cnt, 2, a * y * s};
127                 }
128                 if (opt == 3) break;
129         }
130         cdq(1, cnt);
131         std::sort(q + 1, q + cnt + 1);
132         for (R int i = 1; i <= cnt; ++i)
133                 if (q[i].opet == 1)
134                         printf("%d\n",q[i].ans + q[i + 1].ans - q[i + 2].ans - q[i + 3].ans ), 
135         return 0;
136 }
```

Chapter 6

Others

6.1 vimrc

```
se et ts=4 sw=4 sts=4 nu sc sm lbr is hls mouse=a
sy on
ino <tab> <c-n>
ino <s-tab> <tab>
au winnew * winc L

nm <f6> ggVG"+y
nm <f7> :w<cr>:make<cr>
nm <f8> :!@@<cr>
nm <f9> :!@@ < in<cr>
nm <s-f9> :!(time @@ < in &>> out) &>> out<cr>:sp out<cr>

au filetype cpp cm @@ ./a.out | se cin fdm=syntax mp=g++\ %\ -std=c++11\ -Wall\ -Wextra\ -O2

map <c-p> :ha<cr>
se pheader=%n\ %f

au filetype java cm @@ java %< | se cin fdm=syntax mp=javac\ %
au filetype python cm @@ python % | se si fdm=indent
au bufenter *.kt setf kotlin
au filetype kotlin cm @@ kotlin _%<Kt | se si mp=kotlinc\ %
```

6.2 Java Template

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.math.BigDecimal;
```

```java
import java.math.BigInteger;
import java.math.RoundingMode;
import java.util.ArrayDeque;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Comparator;
import java.util.Deque;
import java.util.LinkedList;
import java.util.List;
import java.util.Scanner;
import java.util.StringTokenizer;

public class Template {
    // Input
    private static BufferedReader reader;
    private static StringTokenizer tokenizer;

    private static String next() {
        try {
            while (tokenizer == null || !tokenizer.hasMoreTokens())
                tokenizer = new StringTokenizer(reader.readLine());
        } catch (IOException e) {
            // do nothing
        }
        return tokenizer.nextToken();
    }

    private static int nextInt() {
        return Integer.parseInt(next());
    }

    private static double nextDouble() {
        return Double.parseDouble(next());
    }

    private static BigInteger nextBigInteger() {
        return new BigInteger(next());
    }

    public static void main(String[] args) {
        reader = new BufferedReader(new InputStreamReader(System.in));
        Scanner scanner = new Scanner(System.in);
        while (scanner.hasNext())
            scanner.next();
    }

```

```java
    // BigInteger & BigDecimal
    private static void bigDecimal() {
        BigDecimal a = BigDecimal.valueOf(1.0);
        BigDecimal b = a.setScale(50, RoundingMode.HALF_EVEN);
        BigDecimal c = b.abs();
        // if scale omitted, b.scale is used
        BigDecimal d = c.divide(b, 50, RoundingMode.HALF_EVEN);
        // since Java 9
        BigDecimal e = d.sqrt(new MathContext(50, RoundingMode.HALF_EVEN));
        BigDecimal x = new BigDecimal(BigInteger.ZERO);
        BigInteger y = BigDecimal.ZERO.toBigInteger(); // RoundingMode.DOWN
        y = BigDecimal.ZERO.setScale(0, RoundingMode.HALF_EVEN).unscaledValue();
    }

    // sqrt for Java 8
    private static BigDecimal sqrt(BigDecimal a, int scale, RoundingMode mode) {
        if (a.equals(BigDecimal.ZERO))
            return BigDecimal.ZERO;
        a = a.setScale(scale, mode);
        BigDecimal ans = a;
        BigDecimal TWO = BigDecimal.valueOf(2L);
        for (int i = 1; i <= scale; i++)
            ans = ans.add(a.divide(ans, scale, mode)).divide(TWO, scale, mode);
        return ans;
    }

    private static BigInteger sqrt(BigInteger a) {
        BigInteger about = BigInteger.ZERO.setBit(a.bitLength() / 2);
        return sqrt(new BigDecimal(a.toString()), new BigDecimal(about.toString())).setScale(0, Roundi
    }

    private static BigDecimal sqrt(BigDecimal a, BigDecimal initial) {
        if (a.equals(BigDecimal.ZERO))
            return BigDecimal.ZERO;
        a = a.setScale(50, RoundingMode.HALF_EVEN);
        BigDecimal ans = initial;
        for (int i = 1; i <= 10; i++)
            ans = ans.add(a.divide(ans, RoundingMode.HALF_EVEN)).divide(BigDecimal.valueOf(2), Roundin
        return ans;
    }

    // ArrayList
    private static void arrayList() {
        List<Integer> list = new ArrayList<>();
        // Generic array is banned
        List[] lists = new List[100];
```

```java
        lists[0] = new ArrayList<Integer>();
        // for List<Integer>, remove(Integer) stands for element, while remove(int) stands for
        list.remove(list.get(1));
        list.remove(list.size() - 1);
        list.clear();
    }

    // Queue
    private static void queue() {
        LinkedList<Integer> queue = new LinkedList<>();
        // return the value without popping
        queue.peek();
        // pop and return the value
        queue.poll();
        Deque<Integer> deque = new ArrayDeque<>();
        deque.peekFirst();
        deque.peekLast();
        deque.pollFirst();
    }

    // Others
    private static void others() {
        Arrays.sort(new int[10]);
        Arrays.sort(new Integer[10], (a, b) -> {
            if (a.equals(b)) return 0;
            if (a > b) return -1;
            return 1;
        });
        Arrays.sort(new Integer[10], Comparator.comparingInt((a) -> (int) a).reversed());
        long a = 1_000_000_000_000_000_000L;
        int b = Integer.MAX_VALUE;
        int c = 'a';
    }
}
```

6.3 Big Fraction

```kotlin
fun gcd(a: Long, b: Long): Long = if (b == 0L) a else gcd(b, a % b)

class Fraction(val a: BigInteger, val b: BigInteger) {
    constructor(a: Long, b: Long) : this(BigInteger.valueOf(a / gcd(a, b)), BigInteger.valueOf(

    operator fun plus(o: Fraction): Fraction {
        var gcd = b.gcd(o.b)
        val tempProduct = (b / gcd) * (o.b / gcd)
        var ansA = a * (o.b / gcd) + o.a * (b / gcd)
```

```
        val gcd2 = ansA.gcd(gcd)
        ansA /= gcd2
        gcd /= gcd2
        return Fraction(ansA, gcd * tempProduct)
    }

    operator fun minus(o: Fraction): Fraction {
        var gcd = b.gcd(o.b)
        val tempProduct = (b / gcd) * (o.b / gcd)
        var ansA = a * (o.b / gcd) - o.a * (b / gcd)
        val gcd2 = ansA.gcd(gcd)
        ansA /= gcd2
        gcd /= gcd2
        return Fraction(ansA, gcd * tempProduct)
    }

    operator fun times(o: Fraction): Fraction {
        val gcd1 = a.gcd(o.b)
        val gcd2 = b.gcd(o.a)
        return Fraction((a / gcd1) * (o.a / gcd2), (b / gcd2) * (o.b / gcd1))
    }
}
```

6.4 模拟退火

```
#include <cstdio>
#include <cmath>
#include <cstdlib>
#include <ctime>

#define R register
#define cmax(_a, _b) (_a < (_b) ? _a = (_b) : 0)
#define maxn 10010
struct Poi {
        double x, y, m;
}p[maxn];
double ans_x, ans_y, fans;
int n;
inline double rand01() {return rand() / 2147483647.0;}
inline double randp() {return (rand() & 1 ? 1 : -1) * rand01();}
inline double sqr(R double x) {return x * x;}
inline double f(R double x, R double y)
{
        R double maxx = 0;
        for (R int i = 1; i <= n; ++i)
                maxx += sqrt(sqr(x - p[i].x) + sqr(y - p[i].y)) * p[i].m;
```

```
        if (maxx < fans) {fans = maxx; ans_x = x; ans_y = y;}
        return maxx;
}
int main()
{
        srand(time(NULL) + clock());
        scanf("%d", &n);
        R double x = 0, y = 0, tot = 0;
        for (R int i = 1; i <= n; ++i)
                scanf("%lf%lf%lf", &p[i].x, &p[i].y, &p[i].m), x += p[i].x * p[i].m, y += p[i].y
        fans = 1e30; x /= tot; y /= tot;
        R double fnow = f(x, y);
        for (R double T = 1e4; T > 1e-4; T *= 0.997)
        {
                R double nx = x + randp() * T, ny = y + randp() * T, fnext = f(nx, ny);
                R double delta = fnext - fnow;
                if (delta < 1e-9 || exp(-delta / T) > rand01())
                {
                        x = nx; y = ny; fnow = fnext;
                }
        }
        printf("%.3lf %.3lf\n", ans_x, ans_y);
        return 0;
}
```

6.5 三分

```
#define maxn 200010
#define inf 1e9
int a[maxn], n;
inline double check(R double x)
{
        R double tmp, tmp1 = 0, tmp2 = 0, maxx = -inf, minn = -inf;
        for (R int i = 1; i <= n; ++i)
        {
                tmp = (double) a[i] - x;

                tmp1 += tmp;
                cmax(maxx, tmp1);
                tmp1 < 0 ? tmp1 = 0 : 0;

                tmp2 -= tmp;
                cmax(minn, tmp2);
                tmp2 < 0 ? tmp2 = 0 : 0;
        }
        return dmax(maxx, minn);
```

```
}
int main()
{
        n = F();
        for (R int i = 1; i <= n; ++i) a[i] = F();
        R double l = -1e4, r = 1e4;
        for (R int i = 1; i <= 100; ++i)
        {
                R double ll = (l + r) * 0.5;
                R double rr = (ll + r) * 0.5;
                if (check(ll) < check(rr)) r = rr;
                else l = ll;
        }
        printf("%.6lf\n", check((l + r) * 0.5) );
        return 0;
}
```

6.6 博弈论模型

- Wythoff's game
  给定两堆石子，每次可以从任意一堆中取至少一个石子，或从两堆中取相同的至少一个石子，取走最后石子的胜
  先手胜当且仅当石子数满足：
  $\lfloor (b-a) \times \phi \rfloor = a, (a \leq b, \phi = \frac{\sqrt{5}+1}{2})$
  先手胜对应的石子数构成两个序列：
  Lower Wythoff sequence: $a_n = \lfloor n \times \phi \rfloor$
  Upper Wythoff sequence: $b_n = \lfloor n \times \phi^2 \rfloor$

- Fibonacci nim
  给定一堆石子，第一次可以取至少一个、少于石子总数数量的石子，之后每次可以取至少一个、不超过上次取石子数量两倍的石子，取走最后石子的胜
  先手胜当且仅当石子数为斐波那契数