### Contents

### 1 Basic 1.1 default code . . . . . . . . . . . . . . . . . 2 math 2.3 MillerRabin other ......... 2.4 Guass 3 flow string 4.1 KMP 4.3 Z-value-palindrome . . . 4.4 Suffix Array(O(NlogN)) . . . . . . . . . . . . 4.6 Aho-Corasick-2016ioicamp . . . . . . . . . . 5.1 Bipartite matching $(O(N^3))$ . . . . . . . . . . 5.3 EdgeBCC 6 data structure 6.1 Treap . . . . . . 6.2 copy on write treap . . . . . . . . . . . . . 6.3 copy on write segment tree . . . . . . . . . . 6.4 Treap+(HOJ 92) . . . . . . . . . . . . . . . . . 6.7 Heavy Light Decomposition . . . . . . . . . 6.8 Disjoint Sets + offline skill . . . . . . . . 7 geometry 7.1 Basic 8 Others

#### Basic 1

#### default code 1.1

```
1
1
   1 #include <bits/stdc++.h>
1
   2 #define PB push_back
   3 #define MP make_pair
   4 #define F first
2
   5 #define S second
   6 #define SZ(x) ((int)(x).size())
     #define ALL(x) (x).begin(),(x).end()
   8 #ifdef _DEBUG_
4
  9
       #define debug(...) printf(__VA_ARGS__)
  10 | #else
4 11
       #define debug(...) (void)0
5 12 #endif
5
  13 using namespace std;
6
  14 typedef long long 11;
7 15
    typedef pair<int,int> PII;
  16 typedef vector<int> VI;
° 17
9 18 int main() {
9 19
       return 0;
  20|}
9
9
10
     1.2
            .vimrc
13
16
   1 color torte
17
   2 syn on
17
   3 set guifont=Consolas:h16: nu sc ai si ts=4
19
        sm sts=4 sw=4
19
19
   5 map <F9> <ESC>:w<CR>:!g++ % -o %< -02 -Wall
20
         -Wno-unused-result -std=c++0x<CR>
20
   6 map <S-F9> <ESC>:w<CR>:!g++ % -o %< -02 -
20
```

```
Wall -Wno-unused-result -D_DEBUG_ -std=c
      ++0x<CR>
7 map <F5> <ESC>:!./%<<CR>
8 map <F6> <ESC>:w<CR>ggvG"+y
9 map <S-F5> <ESC>:!./%< < %<.in<CR>>
10 imap <Home> <ESC>^i
11 com INPUT sp %<.in
```

### math

# 2.1 ext gcd

```
1 / / find one solution (x,y) of ax+by=gcd(
 void ext_gcd(int a,int b,int &g,int &x,int
     &y)
3 {
4
    if(!b){ g=a; x=1; y=0; }
5
    else{ ext_gcd(b, a%b, g, y, x); y -= x*(a
       /b); }
6|}
```

### 2.2 FFT other

```
1 /* FFT code from shik in CodeForces*/
                                                    54|}
 2 /* zj a577*/
                                                    55
 3 #include <bits/stdc++.h>
                                                    56
                                                       char a[N],b[N];
 4 using namespace std;
                                                    57
                                                      int ans[2*N];
 5 const int N=300000;
                                                    58
 6
                                                    59 int main()
 7
  const double PI=acos(-1.0);
                                                    60 {
  struct vir{
                                                    61
                                                         int na,nb,len=1,loglen=0;
 8
                                                         while(scanf("%s%s",a,b)==2)
9
       double re,im;
                                                    62
       vir( double _re=0, double _im=0 ):re(
10
                                                    63
           _re),im(_im){}
                                                    64
                                                           for(int i=2*N-1;i>=0;i--)
                                                    65
                                                             x1[i]=x2[i]=0.0;
11|};
12 vir operator +( vir a, vir b ) { return vir
                                                    66
                                                           for(na=0;a[na];na++);
      (a.re+b.re,a.im+b .im); }
                                                    67
                                                           for(nb=0;b[nb];nb++);
13 vir operator -( vir a, vir b ) { return vir
                                                    68
                                                           for(int i=na-1;i>=0;i--)
      (a.re-b.re,a.im-b .im); }
                                                    69
                                                             x1[i]=(double)(a[na-i-1]-'0');
14 vir operator *( vir a, vir b ) { return vir
                                                    70
                                                           for(int i=nb-1;i>=0;i--)
      (a.re*b.re-a.im*b .im,a.re*b.im+a.im*b.
                                                    71
                                                             x2[i]=(double)(b[nb-i-1]-'0');
                                                           while(len<=2*max(na,nb)+5)</pre>
                                                    72
      re); }
15 vir x1[2*N],x2[2*N];
                                                    73
16
                                                    74
                                                             len*=2;
  int rev( int x, int len ) {
                                                    75
                                                             loglen++;
17
18
                                                    76
       int r=0,i;
19
       for ( i=0; i<len; i++,x>>=1 ) r=(r<<1)</pre>
                                                    77
                                                           fft(x1,len,loglen);
           +(x&1);
                                                    78
                                                           fft(x2,len,loglen);
20
       return r;
                                                    79
                                                           for(int i=0;i<len;i++)</pre>
                                                    80
21
                                                             x1[i]=x1[i]*x2[i];
  }
   void change( vir *x, int len, int loglen )
                                                    81
                                                           dit_fft(x1,len,loglen);
                                                    82
                                                           for(int i=len-1;i>=0;i--)
       for ( int i=0; i<len; i++ )</pre>
                                                    83
23
                                                             ans[i]=(int)round(x1[i].re+0.01);
24
           if ( rev(i,loglen)<i ) swap(x[rev(i 84</pre>
                                                           for(int i=0;i<len;i++)</pre>
               ,loglen)],x[i]);
                                                    85
25
                                                    86
                                                             if(ans[i]>=10)
  void fft( vir *x, int len, int loglen ) {
26
                                                    87
                                                    88
27
       change(x,len,loglen);
                                                               ans[i+1]+=ans[i]/10;
28
       int i,j,s,t=1;
                                                    89
                                                                ans[i]%=10;
29
       for ( i=0; i<loglen; i++,t<<=1 ) {</pre>
                                                    90
30
           for ( s=0; s<len; s+=t+t ) {</pre>
                                                    91
31
                vir a,b,wo(cos(PI/t),sin(PI/t))
                                                   92
                                                           bool zero=0;
                                                    93
                                                           for(int i=len-1;i>=0;i--)
                    ,wn(1,0);
                for ( j=s; j<s+t; j++ ) {</pre>
32
                                                    94
33
                    a=x[j]; b=x[j+t]*wn;
                                                    95
                                                             //printf("%d\n",ans[i]);
                    x[j]=a+b; x[j+t]=a-b;
                                                    96
                                                             if(zero)
34
                                                               printf("%d",ans[i]);
35
                    wn=wn*wo;
                                                    97
                                                             else if(ans[i]>0)
36
                }
                                                    98
37
           }
                                                    99
38
       }
                                                  100
                                                                printf("%d",ans[i]);
39
                                                  101
                                                                zero=1;
40
   void dit_fft( vir *x, int len, int loglen ) 102
                                                  103
                                                  104
41
       int i,j,s,t=len>>1;
                                                           if(!zero)
42
       for ( i=0; i<loglen; i++,t>>=1 ) {
                                                  105
                                                             printf("0");
43
           for ( s=0; s<len; s+=t+t ) {</pre>
                                                  106
                                                           puts("");
44
                vir a,b,wn(1,0),wo(cos(PI/t),-
                                                  107
                                                         }
                                                  108
                    sin(PI/t));
                                                         return 0;
45
                for ( j=s; j<s+t; j++ ) {</pre>
                                                  109 }
46
                    a=x[j]+x[j+t]; b=(x[j]-x[j+
                        t])*wn;
47
                    x[j]=a; x[j+t]=b;
                                                       2.3
                                                             MillerRabin other
48
                    wn=wn*wo;
49
                }
           }
50
                                                     1 /* Miller Rabin code from ioicamp */
51
                                                     2 #include <bits/stdc++.h>
       change(x,len,loglen);
                                                     3 #define PB push back
52
53
       for ( i=0; i<len; i++ ) x[i].re/=len;</pre>
                                                     4 #define MP make_pair
```

for(int i = 0; i < 7; i++) {</pre>

```
5 #define F first
                                                  4
                                                        Frac tmp = mat[i][i]; // Frac -> the
 6 #define S second
                                                            type of data
                                                  5
7 #define SZ(x) ((int)(x).size())
                                                        for(int j = 0; j < 8; j++)</pre>
                                                  6
8 #define ALL(x) (x).begin(),(x).end()
                                                          mat[i][j] = mat[i][j] / tmp;
9 #ifdef _DEBUG_
                                                  7
                                                        for(int j = 0; j < 7; j++) {
    #define debug(...) printf(__VA_ARGS__)
                                                  8
                                                          if(i == j)
                                                  9
11 #else
                                                             continue;
12
                                                 10
                                                          Frac ratio = mat[j][i]; // Frac ->
    #define debug(...) 0
13 #endif
                                                              the type of data
14 using namespace std;
                                                 11
                                                          for(int k = 0; k < 8; k++)
15 typedef long long 11;
                                                 12
                                                            mat[j][k] = mat[j][k] - ratio * mat
16|typedef pair<int,int> PII;
                                                                [i][k];
                                                 13
17 typedef vector<int> VI;
                                                        }
                                                 14
                                                      }
19 | 11 mul(11 a, 11 b, 11 n) {
                                                 15 }
    11 r = 0;
20
    a %= n, b %= n;
21
    while(b) {
22
                                                         flow
23
       if(b\&1) r = (a+r)=n ? a+r-n : a+r);
24
       a = (a+a)=n ? a+a-n : a+a);
25
       b >>= 1;
                                                           dinic
                                                    3.1
26
    }
27
    return r;
                                                  1 #include <bits/stdc++.h>
28 }
                                                  2 #define PB push back
29
30|11 \text{ bigmod(11 a, 11 d, 11 n)}  {
                                                  3 #define MP make_pair
                                                  4 #define F first
31
    if(d==0) return 1LL;
                                                  5 #define S second
32
    if(d==1) return a % n;
     return mul(bigmod(mul(a, a, n), d/2, n),
                                                  6 #define SZ(x) ((int)(x).size())
        d%2?a:1, n);
                                                  7 using namespace std;
34|}
                                                  8 typedef long long 11;
35
                                                  9 typedef pair<int,int> PII;
36 const bool PRIME = 1, COMPOSITE = 0;
                                                 10 typedef vector<int> VI;
37 bool miller_rabin(ll n, ll a) {
                                                 11
                                                 if(__gcd(a, n) == n) return PRIME;
38
39
    if(__gcd(a, n) != 1) return COMPOSITE;
                                                 13 // dinic
40
    11 d = n-1, r = 0, res;
                                                 14 const int MAXV=300;
41
    while(d%2==0) { ++r; d/=2; }
                                                 15 const int MAXE=10000;
                                                 16 const int INF=(int)1e9+10;
42
    res = bigmod(a, d, n);
    if(res == 1 | res == n-1) return PRIME;
43
                                                 17
44
    while(r--) {
                                                 18 struct E{
45
       res = mul(res, res, n);
                                                 19
                                                      int to,co;//capacity
                                                 20
46
       if(res == n-1) return PRIME;
                                                      E(int t=0,int c=0):to(t),co(c){}
47
    }
                                                 21|}eg[2*MAXE];
48
                                                 22
    return COMPOSITE;
49|}
                                                 23 // source:0 sink:n-1
50
                                                 24 struct Flow{
                                                 25
51 bool isprime(ll n) {
                                                      VI e[MAXV];
    if(n==1)
                                                      int ei,v;
52
                                                 26
53
       return COMPOSITE;
                                                 27
                                                      void init(int n) {
54
    11 \text{ as}[7] = \{2, 325, 9375, 28178, 450775,
                                                 28
                                                        v=n;
        9780504, 1795265022};
                                                 29
55
    for(int i=0; i<7; i++)</pre>
                                                 30
                                                        for(int i=0;i<n;i++)</pre>
       if(miller_rabin(n, as[i]) == COMPOSITE) 31
56
                                                          e[i]=VI();
                                                 32
           return COMPOSITE;
57
    return PRIME;
                                                 33
                                                      void add(int a,int b,int c) { //a to b ,
58 }
                                                          maxflow=c
                                                 34
                                                        eg[ei]=E(b,c);
                                                 35
                                                        e[a].PB(ei);
                                                        ei++;
  2.4
       Guass
                                                 37
                                                        eg[ei]=E(a,0);
                                                 38
                                                        e[b].PB(ei);
 1 // be care of the magic number 7 & 8
                                                 39
                                                        ei++;
 2 void guass() {
                                                 40
```

41

1 void KMP\_build(const char \*S,int \*F) {

```
42
     int d[MAXV],qu[MAXV],ql,qr;
                                                        int p=F[0]=-1;
                                                    2
                                                    3
                                                        for(int i=1;S[i];i++) {
43
     bool BFS() {
       memset(d,-1,v*sizeof(int));
                                                    4
44
                                                           while(p!=-1 && S[p+1]!=S[i])
45
                                                    5
       ql=qr=0;
                                                             p=F[p];
46
       qu[qr++]=0;
                                                    6
                                                           if(S[p+1]==S[i])
                                                    7
47
       d[0]=0;
                                                             p++;
                                                    8
48
       while(ql<qr && d[v-1]==-1) {</pre>
                                                           F[i]=p;
                                                    9
49
         int n=qu[q1++];
                                                        }
50
         VI &v=e[n];
                                                   10
51
         for(int i=v.size()-1;i>=0;i--) {
                                                   11
52
           int u=v[i];
                                                   12
                                                      VI KMP_match(const char *S,const int *F,
53
           if(d[eg[u].to]==-1 && eg[u].co>0) {
                                                          const char *T) {
54
                                                   13
                                                        VI ans;
              d[eg[u].to]=d[n]+1;
55
                                                   14
                                                        int p=-1;
              qu[qr++]=eg[u].to;
56
           }
                                                   15
                                                        for(int i=0;T[i];i++) {
57
         }
                                                           while(p!=-1 && S[p+1]!=T[i])
                                                   16
58
                                                   17
                                                             p=F[p];
                                                           if(S[p+1]==T[i])
59
       return d[v-1]!=-1;
                                                   18
60
                                                   19
                                                             p++;
                                                           if(!S[p+1]) {
61
     int ptr[MAXV];
                                                   20
62
     int go(int n,int p) {
                                                   21
                                                             ans.PB(i-p);
                                                   22
63
       if(n==v-1)
                                                             p=F[p];
64
                                                   23
                                                           }
         return p;
                                                   24
                                                        }
65
       VI &u=e[n];
                                                   25
                                                        return ans;
66
       int temp;
67
       for(int i=ptr[n];i<SZ(u);i++)</pre>
                                                   26 }
68
         if(d[n]+1!=d[eg[u[i]].to] || eg[u[i
69
                                                      4.2
                                                             Z-value
             ]].co==0)
70
           continue;
71
         if((temp=go(eg[u[i]].to,min(p,eg[u[i
                                                    1 void Z_build(const char *S,int *Z) {
             ]].co)))==0)
                                                    2
                                                        Z[0]=0;
72
           continue;
                                                    3
                                                        int bst=0;
73
         eg[u[i]].co-=temp;
                                                    4
                                                        for(int i=1;S[i];i++) {
74
         eg[u[i]^1].co+=temp;
                                                    5
                                                           if(Z[bst]+bst<i) Z[i]=0;
75
         ptr[n]=i;
                                                    6
                                                           else Z[i]=min(Z[bst]+bst-i,Z[i-bst]);
76
         return temp;
                                                    7
                                                           while(S[Z[i]]==S[i+Z[i]]) Z[i]++;
77
       }
                                                    8
                                                           if(Z[i]+i>Z[bst]+bst) bst=i;
78
       ptr[n]=SZ(u);
                                                    9
79
       return 0;
                                                   10
80
     }
81
     int max_flow() {
82
       int ans=0,temp;
                                                             Z-value-palindrome
83
       while(BFS()) {
84
         for(int i=0;i<v;i++)</pre>
                                                    1 // AC code of NTUJ1871
85
           ptr[i]=0;
         while((temp=go(0,INF))>0)
                                                      #include <bits/stdc++.h>
86
                                                    3 #define pb push_back
87
           ans+=temp;
88
                                                    4 #define F first
       }
89
       return ans;
                                                    5 #define S second
                                                    6 #define SZ(x) ((int)(x).size())
90
     }
91
  }flow;
                                                    7
                                                      |#define MP make_pair
92
                                                    8 using namespace std;
93
  int main() {
                                                    9 typedef long long 11;
94
                                                   10 typedef pair<int,int> PII;
95
                                                   11
                                                      typedef vector<int> VI;
     return 0;
96|}
                                                   12
                                                   13 char in[100100];
                                                   14 char s[200100];
       string
                                                   15 int z[200100];
                                                   16
                                                   17 int main()
   4.1
         KMP
                                                   18
                                                   19
                                                           while(gets(in))
```

20

{

for(int i=0;i<len;i++)</pre>

```
int len=1;
                                                     19
21
                                                               maxR=max(maxR,R[i]);
            for(int i=0;in[i];i++)
                                                     20
22
                                                             for(int i=0;i<len;i++)</pre>
23
                                                     21
                                                               cnt[R[i]+1]++;
            {
24
                s[len++]='*';
                                                     22
                                                             for(int i=1;i<=maxR;i++)</pre>
25
                s[len++]=in[i];
                                                     23
                                                               cnt[i]+=cnt[i-1];
26
            }
                                                     24
                                                             for(int i=0;i<len;i++)</pre>
27
            s[len]=0;
                                                     25
                                                               SA[cnt[R[i]]++]=i;
28
            z[0]=0;
                                                     26
                                                             for(int i=1;i<len;i*=2)</pre>
29
            z[1]=0;
                                                     27
30
            int bst=1;
                                                     28
                                                               memset(cnt,0,sizeof(int)*(maxR+10));
31
            for(int i=1;i<len;i++)</pre>
                                                     29
                                                               memcpy(tSA,SA,sizeof(int)*(len+10));
                                                     30
32
                                                               memcpy(tR,R,sizeof(int)*(len+i+10));
33
                z[i]=min(bst+z[bst]-i,z[bst+bst
                                                     31
                                                               for(int j=0;j<len;j++)</pre>
                                                                 cnt[R[j]+1]++;
34
                while(s[i+z[i]+1]==s[i-z[i]-1])
                                                     33
                                                               for(int j=1;j<=maxR;j++)</pre>
35
                                                     34
                                                                 cnt[j]+=cnt[j-1];
                     z[i]++;
                                                     35
                                                               for(int j=len-i;j<len;j++)</pre>
36
                if(z[i]+i>bst+z[bst])
                                                                 SA[cnt[R[j]]++]=j;
37
                                                     36
                     bst=i;
38
                                                     37
                                                               for(int j=0;j<len;j++)</pre>
            }
            /*for(int i=1;i<len;i++)
39
                                                     38
40
                putchar(s[i]);
                                                     39
                                                                 int k=tSA[j]-i;
            puts("");
                                                     40
41
                                                                 if(k<0)
42
            for(int i=1;i<len;i++)</pre>
                                                     41
                                                                    continue;
                printf("%d",z[i]);
                                                                 SA[cnt[R[k]]++]=k;
43
                                                     42
            puts(""); */
44
                                                     43
45
            bool yes=0;
                                                     44
                                                               int num=0;
46
            for(int i=3;i<len;i+=2)</pre>
                                                     45
                                                               maxR=0;
                if(z[(i+1)/2]==i/2 && z[(i+len)
47
                                                     46
                                                               R[SA[0]]=num;
                    /2] = (len - i - 1)/2)
                                                     47
                                                               for(int j=1;j<len;j++)</pre>
48
                     yes=1;
                                                     48
49
            if(yes)
                                                     49
                                                                 if(tR[SA[j-1]]<tR[SA[j]] || tR[SA[j</pre>
50
                puts("www");
                                                                     -1]+i]<tR[SA[j]+i])
51
            else
                                                     50
                                                                    num++;
52
                puts("\nu\nu\nu\nu\nu\nu");
                                                     51
                                                                 R[SA[j]]=num;
53
                                                     52
                                                                 maxR=max(maxR,R[SA[j]]);
                                                     53
54
       return 0;
                                                               }
55|}
                                                     54
                                                             }
                                                     55
                                                          }
                                                     56
                                                          void build_H() {
                                                     57
                                                             memset(H,0,sizeof(int)*(len+10));
   4.4
          Suffix Array(O(NlogN))
                                                     58
                                                             for(int i=0;i<len;i++)</pre>
                                                     59
 1 const int SASIZE=100020; // >= (max length
                                                               if(R[i]==0)
                                                     60
       of string + 20)
                                                     61
                                                                 continue;
 2 struct SA{
                                                               int &t=H[R[i]];
                                                     62
 3
     char S[SASIZE]; // put target string into
                                                               if(i>0)
                                                     63
          S[0:(len-1)]
                                                                 t=max(0,H[R[i-1]]-1);
                                                     64
     // you can change the type of S into int
                                                     65
 4
                                                               while(S[i+t]==S[SA[R[i]-1]+t]) t++;
         if required
                                                     66
 5
     // if the string is in int, please avoid
                                                     67
                                                     68|}sa;
         number < 0
 6
     int R[SASIZE*2],SA[SASIZE];
     int tR[SASIZE*2],tSA[SASIZE];
 7
 8
                                  // set len
     int cnt[SASIZE],len;
                                                        4.5
                                                               Aho-Corasick
         before calling build()
 9
     int H[SASIZE];
                                                      1 // AC code of UVa 10679
10
11
     void build_SA() {
                                                      2 #include <cstdio>
12
                                                      3 #include <cstring>
       int maxR=0;
13
       for(int i=0;i<len;i++)</pre>
                                                      4 #include <new>
14
         R[i]=S[i];
15
       for(int i=0;i<=len;i++)</pre>
                                                      6
                                                        struct Trie {
16
         R[len+i]=-1;
                                                      7
                                                          int c;
17
       memset(cnt,0,sizeof(cnt));
                                                      8
                                                          bool fi=0;
```

Trie \*fail, \*ch[52];

```
10
     Trie():c(0){memset(ch,0,sizeof(ch));}
                                                   74
                                                          scanf("%d",&q);
                                                   75
                                                          for(int i=0;i<q;i++) {</pre>
11
   }trie[1000100];
12
                                                   76
                                                             scanf("%s",m);
13
  char m[1010],f[100100];
                                                   77
                                                             insert(m,i);
  Trie *str[1010],*na,*root;
                                                   78
                                                          }
15
                                                   79
                                                          init();
16
  inline int c_i(char a) {
                                                   80
                                                          go(f);
     return (a>='A' && a<='Z') ? a-'A' : a-'a'
17
                                                   81
                                                          for(int i=0;i<q;i++)</pre>
        +26;
                                                   82
                                                             puts(str[i]->fi?"y":"n");
18
  }
                                                   83
19
                                                   84
                                                        return 0;
20 void insert(char *s,int num) {
                                                   85 }
     Trie *at=root;
21
22
     while(*s) {
       if(!at->ch[c_i(*s)])
23
                                                            Aho-Corasick-2016ioicamp
24
         at->ch[c_i(*s)]=new (na++) Trie();
25
       at=at->ch[c_i(*s)],s++;
26
                                                    1 // AC code of 2016ioicamp 54
     }
27
                                                    2 #include <bits/stdc++.h>
     str[num]=at;
28
                                                    3 #define PB push_back
29
                                                    4 #define MP make_pair
  Trie *q[1000100];
                                                    5 #define F first
  int q1,qr;
                                                    6 #define S second
32
                                                    7 #define SZ(x) ((int)(x).size())
  void init() {
33
                                                    8 #define ALL(x) (x).begin(),(x).end()
34
     ql=qr=-1;
                                                    9 #ifdef _DEBUG_
                                                        #define debug(...) printf(__VA_ARGS__)
35
     q[++qr]=root;
                                                   10
     root->fail=NULL;
                                                   11 #else
36
37
     while(ql<qr) {</pre>
                                                   12
                                                        #define debug(...) (void)0
       Trie *n=q[++q1],*f;
38
                                                   13 #endif
39
       for(int i=0;i<52;i++) {</pre>
                                                   14 using namespace std;
40
         if(!n->ch[i])
                                                   15 typedef long long 11;
41
           continue;
                                                   16
                                                      typedef pair<int,int> PII;
42
         f=n->fail;
                                                   17 typedef vector<int> VI;
43
         while(f && !f->ch[i])
                                                   18
44
           f=f->fail;
                                                   19 const int MAXNM=100010;
45
         n->ch[i]->fail=f?f->ch[i]:root;
                                                   20 int pp[MAXNM];
46
         q[++qr]=n->ch[i];
                                                   21
47
                                                   22 const int sizz=100010;
       }
48
                                                   23 int nx[sizz][26], spt;
     }
49
  }
                                                   24 int fl[sizz],efl[sizz],ed[sizz];
50
                                                   25 int len[sizz];
51
  void go(char *s) {
                                                   26 int newnode(int len_=0) {
52
     Trie*p=root;
                                                   27
                                                        for(int i=0;i<26;i++)nx[spt][i]=0;</pre>
     while(*s) {
53
                                                   28
                                                        ed[spt]=0;
54
       while(p && !p->ch[c_i(*s)])
                                                   29
                                                        len[spt]=len_;
55
         p=p->fail;
                                                   30
                                                        return spt++;
       p=p?p->ch[c_i(*s)]:root;
                                                   31 }
56
57
                                                   32 int add(char *s,int p) {
       p->fi=1;
                                                        int l=1;
58
                                                   33
       s++;
59
     }
                                                   34
                                                        for(int i=0;s[i];i++) {
60
                                                   35
                                                          int a=s[i]-'a';
61
                                                   36
                                                          if(nx[p][a]==0) nx[p][a]=newnode(1);
  void AC() {
                                                   37
                                                          p=nx[p][a];
62
     for(int i=qr;i>0;i--)
                                                   38
63
                                                          1++;
64
       q[i]->fail->c+=q[i]->c;
                                                   39
65
  }
                                                   40
                                                        ed[p]=1;
66
                                                   41
                                                        return p;
  int main() {
                                                   42 }
67
68
                                                   43 int q[sizz],qs,qe;
     int T,q;
     scanf("%d",&T);
69
                                                   44 void make_fl(int root) {
                                                        f1[root]=ef1[root]=0;
70
     while(T--) {
                                                   45
71
       na=trie;
                                                   46
                                                        qs=qe=0;
72
       root=new (na++) Trie();
                                                   47
                                                        q[qe++]=root;
73
       scanf("%s",f);
                                                   48
                                                        for(;qs!=qe;) {
```

```
49
        int p=q[qs++];
                                                         5
                                                              graph
        for(int i=0;i<26;i++) {</pre>
 50
 51
           int t=nx[p][i];
                                                                Bipartite matching (O(N^3))
 52
          if(t==0) continue;
 53
           int tmp=fl[p];
 54
           for(;tmp&&nx[tmp][i]==0;) tmp=f1[tmp
                                                       1 // NTUJ1263
                                                       2 #include <bits/stdc++.h>
                                                       3 #define pb push_back
           f1[t]=tmp?nx[tmp][i]:root;
 55
 56
           efl[t]=ed[fl[t]]?fl[t]:efl[fl[t]];
                                                         #define F first
 57
           q[qe++]=t;
                                                         #define S second
 58
        }
                                                       6
                                                         #define SZ(x) ((int)(x).size())
 59
      }
                                                       7
                                                         #define MP make_pair
 60
                                                       8 using namespace std;
    char s[MAXNM];
                                                       9 typedef long long 11;
    char a[MAXNM];
                                                      10 typedef pair<int,int> PII;
                                                      11 typedef vector<int> VI;
 63
    int dp[MAXNM][4];
 64
                                                      12
                                                      13
                                                         bool is(ll x)
 65
                                                      14 {
    void mmax(int &a,int b) {
 66
 67
      a=max(a,b);
                                                      15
                                                           ll l=1,r=2000000,m;
 68
                                                      16
                                                           while(l<=r)</pre>
 69
                                                      17
 70
    void match(int root) {
                                                      18
                                                              m=(1+r)/2;
 71
                                                      19
      int p=root;
                                                              if(m*m==x)
      for(int i=1;s[i];i++) {
 72
                                                      20
                                                                return 1;
 73
        int a=s[i]-'a';
                                                      21
                                                              if(m*m<x)
 74
                                                                l=m+1;
        for(;p&&nx[p][a]==0;p=f1[p]);
                                                      22
 75
                                                      23
        p=p?nx[p][a]:root;
                                                              else
 76
        for(int j=1;j<=3;j++)</pre>
                                                      24
                                                                r=m-1;
                                                      25
 77
           dp[i][j]=dp[i-1][j];
 78
        for(int t=p;t;t=efl[t]) {
                                                      26
                                                           return 0;
 79
           if(!ed[t])
                                                      27
                                                         }
                                                      28
 80
             continue;
                                                         VI odd, even;
 81
           for(int j=1;j<=3;j++)</pre>
                                                      29
             mmax(dp[i][j],dp[i-len[t]][j-1]+(pp
 82
                                                      30 int in[300];
                 [i]-pp[i-len[t]]));
                                                      31 VI e[300];
                                                      32 int match[300];
 83
        }
 84
      }
                                                      33 bool vis[300];
    }
 85
                                                      34
                                                      35
                                                         bool DFS(int x)
 86
 87
    int main() {
                                                      36
 88
      int T;
                                                      37
                                                           vis[x]=1;
 89
      scanf("%d",&T);
                                                      38
                                                           for(int u:e[x])
                                                      39
 90
      while(T--) {
 91
                                                              if(match[u]==-1 || (!vis[match[u]]&&DFS
        int n,m;
                                                      40
        scanf("%d%d",&n,&m);
 92
                                                                 (match[u])))
        scanf("%s",s+1);
 93
                                                      41
                                                              {
        for(int i=1;i<=n;i++)</pre>
                                                                match[u]=x;
 94
                                                      42
 95
           scanf("%d",pp+i);
                                                      43
                                                                match[x]=u;
 96
        for(int i=1;i<=n;i++)</pre>
                                                      44
                                                                return 1;
                                                      45
 97
          pp[i]+=pp[i-1];
                                                              }
 98
        spt=1;
                                                      46
 99
        int root=newnode();
                                                      47
                                                           return 0;
        for(int i=0;i<m;i++) {</pre>
                                                      48
100
           scanf("%s",a);
                                                      49
101
102
           add(a,root);
                                                      50 int main()
103
        }
                                                      51
104
        make_fl(root);
                                                      52
                                                           int N;
                                                           while(scanf("%d",&N)==1)
105
        for(int i=1;i<=n;i++)</pre>
                                                      53
106
           dp[i][1]=dp[i][2]=dp[i][3]=0;
                                                      54
        match(root);
107
                                                      55
                                                             odd.clear();
        printf("%d \n", dp[n][3]);
108
                                                      56
                                                              even.clear();
109
                                                      57
                                                              for(int i=0;i<N;i++)</pre>
110
                                                      58
                                                                e[i].clear();
      return 0;
111|}
                                                      59
                                                              for(int i=0;i<N;i++)</pre>
```

```
60
                                                    34
                                                                         sol.clear();
         scanf("%d",in+i);
                                                    35
61
                                                                         sol.push back(v);
         if(in[i]%2==0)
                                                    36
62
                                                                         return 1;
                                                    37
                                                                     }
63
            even.pb(i);
                                                    38
64
         else
                                                                     return 0;
65
           odd.pb(i);
                                                    39
                                                                }
66
                                                    40
                                                                for(int i=0; i<(V+31)/32; i++) {</pre>
       for(int i:even)
                                                    41
                                                                     for(int a = s[k][i]; a; d++) {
67
68
         for(int j:odd)
                                                    42
                                                                         if(k + (c-d) <= ans) return</pre>
69
            if(is(111*in[i]*in[i]+111*in[j]*in[
               j]) && __gcd(in[i],in[j])==1)
                                                    43
                                                                         int lb = a&(-a), lg = 0;
70
                                                    44
              e[i].pb(j), e[j].pb(i);
                                                                         a ^= lb;
                                                    45
71
       int ans=0;
                                                                         while(lb!=1) {
72
       fill(match, match+N, -1);
                                                    46
                                                                              lb = (unsigned int)(lb)
73
       for(int i=0;i<N;i++)</pre>
                                                                                  >> 1:
74
         if(match[i]==-1)
                                                    47
                                                                              lg ++;
75
                                                    48
         {
76
           fill(vis, vis+N,0);
                                                    49
                                                                         int u = i*32 + lg;
77
                                                    50
           if(DFS(i))
                                                                         if(k + dp[u] <= ans) return</pre>
78
              ans++;
79
         }
                                                    51
                                                                         if(dfs(u, k+1)) {
80
       printf("%d\n",ans);
                                                    52
                                                                              sol.push_back(v);
81
     }
                                                    53
                                                                              return 1;
                                                                         }
82
                                                    54
     return 0;
83 }
                                                    55
                                                                     }
                                                    56
                                                                }
                                                    57
                                                                return 0;
                                                    58
                                                            }
         Max clique(bcw)
   5.2
                                                    59
                                                    60
                                                            int solve() {
 1 class MaxClique {
                                                    61
                                                                for(int i=V-1; i>=0; i--) {
  public:
                                                    62
                                                                     dfs(i, 1);
 3
       static const int MV = 210;
                                                    63
                                                                     dp[i] = ans;
 4
                                                    64
                                                                }
5
                                                    65
       int V;
                                                                return ans;
 6
       int el[MV][MV/30+1];
                                                    66
                                                            }
 7
       int dp[MV];
                                                    67|};
8
       int ans;
9
       int s[MV][MV/30+1];
10
       vector<int> sol;
                                                              EdgeBCC
11
12
       void init(int v) {
13
           V = v; ans = 0;
                                                     1 const int MAXN=1010;
                                                     2 const int MAXM=5010;
14
           FZ(el); FZ(dp);
15
       }
                                                     3 VI e[MAXN];
                                                     4 int low[MAXN], lvl[MAXN], bel[MAXN];
16
17
       /* Zero Base */
                                                     5 bool vis[MAXN];
       void addEdge(int u, int v) {
18
                                                     6
                                                       int cnt;
                                                     7
19
                                                       VI st;
            if(u > v) swap(u, v);
           if(u == v) return;
20
                                                     8
                                                       void DFS(int x,int 1,int p) {
                                                     9
21
            el[u][v/32] |= (1<<(v%32));
                                                         st.PB(x);
22
       }
                                                    10
                                                         vis[x]=1;
23
                                                    11
                                                         low[x]=lvl[x]=l;
       bool dfs(int v, int k) {
                                                         bool top=0;
24
                                                    12
25
            int c = 0, d = 0;
                                                    13
                                                         for(int u:e[x]) {
                                                            if(u==p && !top) {
26
            for(int i=0; i<(V+31)/32; i++) {</pre>
                                                    14
27
                s[k][i] = el[v][i];
                                                    15
                                                              top=1;
28
                if(k != 1) s[k][i] &= s[k-1][i
                                                    16
                                                              continue;
                                                    17
29
                c += __builtin_popcount(s[k][i
                                                    18
                                                            if(!vis[u]) {
                                                    19
                                                              DFS(u,l+1,x);
                    ]);
30
                                                    20
31
            if(c == 0) {
                                                    21
                                                            low[x]=min(low[x],low[u]);
```

23

**if**(x==1 || low[x]==1) {

32

33

 $if(k > ans) {$ 

ans = k;

```
while(st.back()!=x) {
24
25
          bel[st.back()]=cnt;
26
          st.pop_back();
27
28
       bel[st.back()]=cnt;
29
       st.pop_back();
30
       cnt++;
31
     }
32
   }
33
   int main() {
34
     int T;
35
     scanf("%d",&T);
36
     while(T--) {
37
       int N,M,a,b;
38
       scanf("%d%d",&N,&M);
39
       fill(vis, vis+N+1,0);
40
       for(int i=1;i<=N;i++)</pre>
41
          e[i].clear();
42
       while(M--) {
          scanf("%d%d",&a,&b);
43
44
          e[a].PB(b);
45
         e[b].PB(a);
46
       }
47
       cnt=0;
       DFS(1,0,-1);
48
       /****/
49
50
     }
51
     return 0;
52|}
```

### 5.4 VerticeBCC

```
1 const int MAXN=10000;
 2 const int MAXE=100000;
 4 VI e[MAXN+10];
 5 vector<PII> BCC[MAXE];
 6 int bccnt;
   vector<PII> st;
 8 bool vis[MAXN+10];
9
  int low[MAXN+10],level[MAXN+10];
10
  void DFS(int x,int p,int 1) {
11
12
     vis[x]=1;
13
     level[x]=low[x]=1;
14
     for(int u:e[x]) {
15
       if(u==p)
16
         continue;
       if(vis[u]) {
17
18
         if(level[u]<1) {</pre>
19
           st.PB(MP(x,u));
20
           low[x]=min(low[x],level[u]);
         }
21
22
       }
       else {
23
24
         st.PB(MP(x,u));
         DFS(u,x,l+1);
25
26
         if(low[u]>=1) {
27
           PII t=st.back();
28
           st.pop_back();
29
           while(t!=MP(x,u)) {
30
             BCC[bccnt].PB(t);
31
             t=st.back();
```

```
32
              st.pop_back();
33
34
            BCC[bccnt].PB(t);
35
            bccnt++;
36
37
         low[x]=min(low[x],low[u]);
38
       }
39
     }
40
41
42
   int main() {
43
     int T,N,M;
     scanf("%d",&T);
44
45
     while(T--) {
       scanf("%d%d",&N,&M);
46
47
       for(int i=0;i<N;i++)</pre>
48
         e[i].clear();
49
       int cnt=0;
50
       while(1) {
51
         int x,y;
52
          scanf("%d%d",&x,&y);
53
         if(x==-1 \&\& y==-1)
54
            break;
55
         cnt++;
56
         e[x].PB(y);
57
         e[y].PB(x);
58
59
       for(int i=0;i<N;i++) { // no multi-edge</pre>
60
         sort(ALL(e[i]));
61
         e[i].erase(unique(ALL(e[i])),e[i].end
             ());
62
       fill(vis, vis+N,0);
63
64
       while(bccnt)
65
         BCC[--bccnt].clear();
66
       DFS(0,-1,0);
67
68
     }
69
     return 0;
70 }
```

#### 5.5 Them.

```
    Max (vertex) independent set = Max clique on Complement graph
    Min vertex cover = |V| - Max independent set
    On bipartite: Min vertex cover = Max Matching(edge independent)
    Any graph with no isolated vertices: Min edge cover + Max Matching = |V|
```

## 6 data structure

## 6.1 Treap

```
1 #include <cstdlib>
2 #include <cstdio>
3 #include <algorithm>
4
5 using namespace std;
```

**if**(!t) a = b = NULL;

```
69
                                                        else if(sz(t->1) < k) {
                                                   70
   typedef long long 11;
                                                           a = t;
                                                   71
8
                                                           push(a);
9
                                                   72
  const int N = 100000 + 10;
                                                           split(t->r, k - sz(t->l) - 1, a->r, b);
                                                   73
10
                                                           pull(a);
11 struct Treap {
                                                   74
                                                        }
12
     static Treap mem[N], *pmem;
                                                   75
                                                        else {
13
                                                   76
                                                           b = t;
14
     int sz, pri;
                                                   77
                                                           push(b);
15
     ll val, sum, add;
                                                   78
                                                           split(t->1, k, a, b->1);
16
     Treap *1, *r;
                                                   79
                                                           pull(b);
                                                   80
17
18
                                                   81 }
     Treap() {}
19
     Treap(ll val):
                                                   82
       l(NULL), r(NULL), sz(1), pri(rand()),
20
                                                   83 int main() {
           val(_val), sum(_val), add(0) {}
                                                   84
                                                        srand(105105);
21| Treap::mem[N], *Treap::pmem = Treap::mem;
                                                   85
                                                        int n, q;
22
                                                   86
23 Treap* make(ll val) {
                                                   87
                                                        scanf("%d%d", &n, &q);
24
     return new (Treap::pmem++) Treap(val);
                                                   88
25 }
                                                   89
                                                        Treap *t = NULL;
                                                   90
26
                                                        for(int i = 0; i < n; i++) {</pre>
                                                   91
                                                          11 tmp;
27 inline int sz(Treap *t) {
                                                           scanf("%lld", &tmp);
     return t ? t->sz : 0;
                                                   92
28
29|}
                                                   93
                                                           t = merge(t, make(tmp));
30
                                                   94
31 inline ll sum(Treap *t) {
                                                   95
     return t ? t->sum + t->add * sz(t) : 0;
                                                        while(q--) {
32
                                                   96
                                                   97
33|}
                                                           char c;
                                                   98
34
                                                           int 1, r;
35 inline void add(Treap *t, ll x) {
                                                   99
                                                           scanf("\n%c %d %d", &c, &l, &r);
                                                  100
36
    t->add += x;
37 }
                                                  101
                                                           Treap *tl = NULL, *tr = NULL;
                                                           if(c == 'Q') {
38
                                                  102
39 void push(Treap *t) {
                                                  103
                                                             split(t, 1 - 1, tl, t);
40
    t->val += t->add;
                                                  104
                                                             split(t, r - l + 1, t, tr);
     if(t->1) t->1->add += t->add;
                                                  105
                                                             printf("%lld\n", sum(t));
41
     if(t->r) t->r->add += t->add;
                                                  106
                                                             t = merge(tl, merge(t, tr));
42
                                                          }
43
     t->add = 0;
                                                  107
                                                           else {
44|}
                                                  108
45
                                                  109
                                                             11 x;
46 void pull(Treap *t) {
                                                  110
                                                             scanf("%lld", &x);
47
    t\rightarrow sum = sum(t\rightarrow l) + sum(t\rightarrow r) + t\rightarrow val;
                                                  111
                                                             split(t, 1 - 1, tl, t);
48
     t->sz = sz(t->1) + sz(t->r) + 1;
                                                  112
                                                             split(t, r - l + 1, t, tr);
49 }
                                                  113
                                                             add(t, x);
50
                                                  114
                                                             t = merge(tl, merge(t, tr));
51 Treap* merge(Treap *a, Treap *b) {
                                                  115
52
     if(!a | | !b) return a ? a : b;
                                                  116
53
                                                  117
     else if(a->pri > b->pri) {
54
                                                  118
                                                        return 0;
       push(a);
55
                                                  119 }
       a->r = merge(a->r, b);
56
       pull(a);
57
       return a;
58
     }
                                                            copy on write treap
     else {
59
60
       push(b);
61
       b->1 = merge(a, b->1);
                                                    1 #include <cstdlib>
       pull(b);
                                                    2 #include <cstdio>
62
63
       return b;
                                                    3 #include <algorithm>
                                                    4 #include <climits>
64
     }
65 }
                                                    5 #include <cstring>
66
67
  void split(Treap* t, int k, Treap *&a,
                                                    7
                                                      using namespace std;
      Treap *&b) {
```

9 | const int N = 1000000 + 10;

```
10
                                                            if(!a || !b) {
                                                     73
                                                     74
                                                                 Treap* t = a? make(a) : make(b);
11
   struct Treap {
                                                     75
12
       char val;
                                                                 t \rightarrow refs = 0;
13
                                                     76
                                                                 takeRef(t->1);
       int sz, refs;
14
       Treap *1, *r;
                                                     77
                                                                 takeRef(t->r);
15
                                                     78
                                                                 return t;
16
       Treap() {}
                                                     79
                                                            }
17
       Treap(char _val):
                                                     80
18
           val(_val), sz(1), refs(0), l(NULL),
                                                     81
                                                            Treap* t;
                 r(NULL) {}
                                                     82
                                                            if( rnd(a->sz+b->sz) < a->sz) {
19
                                                     83
                                                                 t = make(a);
  };
                                                                 t \rightarrow refs = 0;
20
                                                     84
  Treap* make(Treap* t) {
                                                     85
                                                                 t->r = merge(a->r, b);
21
       return new Treap(*t);
                                                     86
                                                                 takeRef(t->1);
23|}
                                                     87
                                                                 takeRef(t->r);
                                                            }
24
                                                     88
   Treap* make(char _val) {
                                                            else {
25
                                                     89
                                                     90
                                                                 t = make(b);
26
       return new Treap(_val);
27
                                                     91
                                                                 t \rightarrow refs = 0;
28
                                                     92
                                                                 t->1 = merge(a, b->1);
29
   void print_ref(Treap* t) {
                                                     93
                                                                 takeRef(t->1);
30
       if(!t) return ;
                                                     94
                                                                 takeRef(t->r);
       print_ref(t->1);
                                                     95
                                                            }
31
       printf("%d ", t->refs);
32
                                                     96
33
       print ref(t->r);
                                                     97
                                                            pull(t);
34
  }
                                                     98
                                                            return t;
                                                     99
35
  void print(Treap* t) {
                                                    100
36
37
       if(!t) return ;
                                                        void split(Treap* t, int k, Treap* &a,
                                                    101
38
       print(t->1);
                                                            Treap* &b) {
39
       putchar(t->val);
                                                    102
                                                            if(!t) a = b = NULL;
40
       print(t->r);
                                                    103
                                                            else if(sz(t->1) < k) {
41
                                                    104
                                                                 a = make(t);
42
                                                    105
                                                                 a \rightarrow refs = 0;
43
                                                                 split(a->r, k-sz(t->l)-1, a->r, b);
   void takeRef(Treap* t) {
                                                    106
44
       if(t)
              t->refs++;
                                                    107
                                                                 takeRef(a->1);
45 }
                                                    108
                                                                 takeRef(a->r);
46
                                                    109
                                                                 pull(a);
   void dropRef(Treap* t) {
47
                                                    110
                                                            }
                                                            else {
48
       if(t) {
                                                    111
49
            char c = t->val;
                                                    112
                                                                 b = make(t);
            t->refs--;
50
                                                    113
                                                                 b \rightarrow refs = 0;
51
            if(t->refs <= 0) {
                                                    114
                                                                 split(b->1, k, a, b->1);
52
                dropRef(t->1);
                                                    115
                                                                 takeRef(b->1);
53
                dropRef(t->r);
                                                    116
                                                                 takeRef(b->r);
                                                    117
54
                delete t;
                                                                 pull(b);
55
            }
                                                    118
                                                            }
56
       }
                                                    119
57
                                                    120
58
                                                    121
                                                        void print_inorder(Treap* t) {
59
  int sz(Treap* t) {
                                                    122
                                                            if(!t) return ;
60
       return t ? t->sz : 0;
                                                    123
                                                            putchar(t->val);
61
                                                    124
                                                            print_inorder(t->1);
62
                                                            print_inorder(t->r);
                                                    125
  int rnd(int m) {
                                                    126
63
64
       static int x = 851025;
                                                    127
65
       return (x = (x*0xdefaced+1) & INT_MAX)
                                                    128
                                                        char s[N];
           % m;
                                                    129
                                                    130 int main() {
66|}
                                                    131
67
                                                            int m;
                                                            scanf("%d", &m);
68
  void pull(Treap* t) {
                                                    132
69
                                                            scanf("%s", s);
       t->sz = sz(t->1) + sz(t->r) + 1;
                                                    133
70 }
                                                    134
                                                            int n = strlen(s);
                                                    135
71
                                                            int q;
72 Treap* merge(Treap* a, Treap* b) {
                                                    136
                                                            scanf("%d", &q);
```

```
137
138
        Treap* t = NULL;
139
        for(int i = 0; i < n; i++) {</pre>
            Treap *a = t, *b = make(s[i]);
140
141
            t = merge(a, b);
142
            dropRef(a);
143
            dropRef(b);
        }
144
145
        while(q--) {
146
147
            int 1, r, x;
             scanf("%d%d%d", &1, &r, &x);
148
149
150
151
            Treap *a, *b, *c, *d;
            a = b = c = d = NULL;
152
            split(t, l, a, b);
153
            dropRef(a);
154
155
             split(b, r-1, c, d);
156
            dropRef(b);
157
            dropRef(d);
158
             split(t, x, a, b);
159
            dropRef(t);
            Treap* t2 = merge(c, b);
160
161
            dropRef(b);
162
            dropRef(c);
163
            t = merge(a, t2);
164
            dropRef(a);
            dropRef(t2);
165
166
167
            if(t->sz > m) {
                 Treap* t2 = NULL;
168
169
                 split(t, m, t2, a);
170
                 dropRef(a);
171
                 dropRef(t);
172
                 t = t2;
173
            }
174
        }
175
176
        print(t);
177
        putchar('\n');
178
179
        return 0;
180 }
```

# 6.3 copy on write segment tree

```
1 #include <cstdlib>
 2 #include <cstdio>
 3 #include <algorithm>
 4 #include <vector>
5
 6 using namespace std;
7
8 | const int N = 50000 + 10;
9 | const int Q = 10000 + 10;
10
11 struct Seg {
12
     static Seg mem[N*80], *pmem;
13
14
     int val;
15
     Seg *tl, *tr;
16
```

```
17
     Seg():
18
       tl(NULL), tr(NULL), val(0) {}
19
20
     Seg* init(int 1, int r) {
21
       Seg* t = new (pmem++) Seg();
22
       if(1 != r) {
23
         int m = (1+r)/2;
24
         t->tl = init(l, m);
25
         t->tr = init(m+1, r);
26
       }
27
       return t;
28
     }
29
30
     Seg* add(int k, int l, int r) {
31
       Seg* _t = new (pmem++) Seg(*this);
32
       if(l==r) {
         _t->val++;
33
34
         return _t;
35
       }
36
37
       int m = (1+r)/2;
38
       if(k <= m) _t->tl = tl->add(k, l, m);
39
               _t->tr = tr->add(k, m+1, r);
40
41
       _t->val = _t->tl->val + _t->tr->val;
42
       return _t;
43
44|} Seg::mem[N*80], *Seg::pmem = mem;
45
46 int query(Seg* ta, Seg* tb, int k, int l,
      int r) {
47
     if(l == r) return l;
48
49
     int m = (1+r)/2;
50
51
     int a = ta->tl->val;
52
     int b = tb->tl->val;
     if(b-a >= k) return query(ta->tl, tb->tl
53
        , k, l, m);
54
               return query(ta->tr, tb->tr, k
        -(b-a), m+1, r);
55|};
56
57 struct Query {
58
     int op, 1, r, k, c, v;
59
     bool operator<(const Query b) const {</pre>
60
61
       return c < b.c;</pre>
62
     }
63|} qs[Q];
64|int arr[N];
65 Seg *t[N];
66 vector<int> vec2;
67
68 int main() {
     int T;
69
     scanf("%d", &T);
70
71
72
     while(T--) {
73
       int n, q;
       scanf("%d%d", &n, &q);
74
75
76
       for(int i = 1; i <= n; i++) {</pre>
         scanf("%d", arr+i);
77
78
         vec2.push_back(arr[i]);
```

rsum, mx\_sum;

```
79
                                                              Treap *1, *r;
                                                      12
        for(int i = 0; i < q; i++) {</pre>
 80
                                                      13
           scanf("%d", &qs[i].op);
                                                      14
 81
                                                              Treap() {}
           if(qs[i].op == 1) scanf("%d%d%d", &qs 15
 82
                                                              Treap(int _val) :
               [i].1, &qs[i].r, &qs[i].k);
                                                      16
                                                                  pri(rand()), sz(1), val(_val), chg(
                scanf("%d%d", &qs[i].c, &qs[i].
 83
                                                                      INF), rev(0), sum(_val), lsum(
                                                                      _val), rsum(_val), mx_sum(_val),
              v);
                                                                       1(NULL), r(NULL) {}
 84
 85
           if(qs[i].op == 2) vec2.push_back(qs[i 17
                                                         };
                                                      18
 86
        }
                                                      19
                                                         int sz(Treap* t) {return t ? t->sz : 0;}
        sort(vec2.begin(), vec2.end());
                                                         int sum(Treap* t) {
 87
                                                      20
        vec2.resize(unique(vec2.begin(), vec2.
                                                      21
                                                              if(!t)
                                                                      return 0;
 88
            end())-vec2.begin());
                                                      22
                                                              if(t->chg == INF)
                                                                                    return t->sum;
 89
        for(int i = 1; i <= n; i++) arr[i] =</pre>
                                                      23
                                                              else
                                                                       return t->chg*t->sz;
            lower_bound(vec2.begin(), vec2.end() 24
            , arr[i]) - vec2.begin();
                                                      25
                                                         int lsum(Treap* t) {
 90
                                                      26
                                                              if(!t) return -INF;
        int mn = 0, mx = vec2.size()-1;
 91
                                                      27
                                                              if(t->chg != INF)
                                                                                   return max(t->chg,
 92
        for(int i = 0; i <= n; i++) t[i] = NULL</pre>
                                                                  (t->chg)*(t->sz));
                                                      28
                                                              if(t->rev) return t->rsum;
        t[0] = new (Seg::pmem++) Seg();
                                                      29
                                                              return t->lsum;
 93
 94
        t[0] = t[0] - \sinh(mn, mx);
                                                      30 }
 95
        int ptr = 0;
                                                         int rsum(Treap* t) {
                                                      31
        for(int i = 1; i <= n; i++) {</pre>
 96
                                                      32
                                                              if(!t) return -INF;
 97
          t[i] = t[i-1]->add(arr[i], mn, mx);
                                                      33
                                                              if(t->chg != INF)
                                                                                    return max(t->chg,
 98
                                                                 (t->chg)*(t->sz));
 99
                                                              if(t->rev) return t->lsum;
                                                      34
        for(int i = 0; i < q; i++) {</pre>
100
                                                      35
                                                              return t->rsum;
                                                      36
101
          int op = qs[i].op;
102
           if(op == 1) {
                                                      37
                                                         int mx_sum(Treap* t) {
             int 1 = qs[i].1, r = qs[i].r, k =
                                                      38
                                                              if(!t) return -INF;
103
                 qs[i].k;
                                                      39
                                                              if(t->chg != INF)
                                                                                    return max(t->chg,
             printf("%d \mid n", vec2[query(t[1-1], t
104
                                                                  (t->chg)*(t->sz));
                                                      40
                                                              return t->mx_sum;
                 [r], k, mn, mx)]);
105
                                                      41
106
           if(op == 2) {
                                                      42
107
             continue;
                                                      43
                                                         void push(Treap* t) {
                                                      44
                                                              if(t->chg != INF) {
108
           if(op == 3) puts("7122");
109
                                                      45
                                                                  t->val = t->chg;
110
                                                      46
                                                                  t\rightarrow sum = (t\rightarrow sz) * (t\rightarrow chg);
111
                                                      47
                                                                  t->lsum = t->rsum = t->mx_sum = max
112
        vec2.clear();
                                                                      (t->sum, t->val);
113
        Seg::pmem = Seg::mem;
                                                      48
                                                                  if(t->1)
                                                                               t->1->chg = t->chg;
                                                      49
114
      }
                                                                  if(t->r)
                                                                                t->r->chg = t->chg;
115
                                                                  t->chg = INF;
                                                      50
116
                                                      51
      return 0;
117 }
                                                      52
                                                              if(t->rev) {
                                                                  swap(t->1, t->r);
                                                      53
                                                      54
                                                                  if(t->1)
                                                                                t->l->rev ^= 1;
                                                      55
                                                                  if(t->r)
                                                                                t->r->rev ^= 1;
    6.4
           Treap+(HOJ 92)
                                                      56
                                                                  t \rightarrow rev = 0;
                                                      57
                                                              }
  1 #include <cstdlib>
                                                      58
  2 #include <cstdio>
                                                      59
  3 #include <algorithm>
                                                      60
                                                         void pull(Treap* t) {
                                                             t\rightarrow sz = sz(t\rightarrow 1)+sz(t\rightarrow r)+1;
  4 #include <cstring>
                                                      61
  5
                                                      62
                                                             t\rightarrow sum = sum(t\rightarrow 1)+sum(t\rightarrow r)+t\rightarrow val;
                                                              t\rightarrow lsum = max(lsum(t\rightarrow l), sum(t\rightarrow l)+max
  6 using namespace std;
                                                      63
  7
                                                                  (0, lsum(t->r))+t->val);
  8
    const int INF = 103456789;
                                                      64
                                                              t->rsum = max(rsum(t->r), sum(t->r)+max
 9
                                                                 (0, rsum(t->1))+t->val);
 10
    struct Treap {
                                                      65
                                                              t->mx_sum = max(max(mx_sum(t->1)),
                                                                 mx_sum(t->r)), max(0, rsum(t->1))+
 11
        int pri, sz, val, chg, rev, sum, lsum,
                                                                 max(0, lsum(t->r))+t->val);
```

```
66|}
                                                    129
 67
                                                    130
                                                                 if(!strcmp(s, "INSERT")) {
    Treap* merge(Treap* a, Treap* b) {
                                                    131
 68
                                                                     int p, k;
 69
                                                    132
                                                                     scanf("%d%d", &p, &k);
        if(!a || !b)
                          return a ? a : b;
 70
                                                    133
                                                                     for(int i = 0; i < k; i++) {</pre>
        if(a->pri > b->pri) {
 71
                                                    134
            push(a);
                                                                          int x;
                                                                          scanf("%d", &x);
 72
            a->r = merge(a->r, b);
                                                    135
                                                                          t2 = merge(t2, new Treap(x))
 73
            pull(a);
                                                    136
 74
            return a;
                                                                             );
 75
        }
                                                    137
                                                                     }
 76
        else {
                                                    138
                                                                     split(t, p, tl, tr);
                                                    139
 77
            push(b);
                                                                     t = merge(t1, merge(t2, tr));
 78
            b->1 = merge(a, b->1);
                                                    140
                                                                 }
 79
            pull(b);
                                                    141
 80
            return b;
                                                    142
                                                                 if(!strcmp(s, "DELETE")) {
                                                    143
 81
        }
                                                                     int p, k;
                                                                     scanf("%d%d", &p, &k);
 82
    }
                                                    144
 83
                                                    145
                                                                     split(t, p-1, tl, t);
    void split(Treap* t, int k, Treap* &a,
                                                    146
 84
                                                                     split(t, k, t, tr);
       Treap* &b) {
                                                    147
                                                                     del(t);
 85
        if(!t) {
                                                    148
                                                                     t = merge(tl, tr);
                                                    149
                                                                 }
 86
            a = b = NULL;
 87
                                                    150
            return ;
                                                                 if(!strcmp(s, "MAKE-SAME")) {
 88
        }
                                                    151
                                                                     int p, k, 1;
 89
        push(t);
                                                    152
 90
        if(sz(t->1) < k) {
                                                    153
                                                                     scanf("%d%d%d", &p, &k, &1);
 91
                                                    154
            a = t;
                                                                     split(t, p-1, tl, t);
                                                    155
 92
            push(a);
                                                                     split(t, k, t, tr);
                                                                             t->chg = 1;
 93
             split(t->r, k-sz(t->l)-1, a->r, b); 156
                                                                     if(t)
 94
            pull(a);
                                                    157
                                                                     t = merge(tl, merge(t, tr));
 95
        }
                                                    158
        else {
                                                    159
 96
 97
            b = t;
                                                    160
                                                                 if(!strcmp(s, "REVERSE")) {
            push(b);
                                                                     int p, k;
 98
                                                    161
 99
                                                                     scanf("%d%d", &p, &k);
                                                    162
             split(t->1, k, a, b->1);
100
            pull(b);
                                                    163
                                                                     split(t, p-1, tl, t);
101
        }
                                                    164
                                                                     split(t, k, t, tr);
102|}
                                                    165
                                                                     if(t)
                                                                             t->rev ^= 1;
                                                    166
103
                                                                     t = merge(tl, merge(t, tr));
    void del(Treap* t) {
104
                                                    167
        if(!t) return;
105
                                                    168
106
        del(t->1);
                                                    169
                                                                 if(!strcmp(s, "GET-SUM")) {
                                                    170
107
        del(t->r);
                                                                     int p, k;
                                                                     scanf("%d%d", &p, &k);
108
        delete t;
                                                    171
                                                                     split(t, p-1, tl, t);
109 }
                                                    172
                                                                     split(t, k, t, tr);
110
                                                    173
                                                                     printf("%d \setminus n", sum(t));
    int main() {
                                                    174
111
                                                                     t = merge(tl, merge(t, tr));
112
        srand(7122);
                                                    175
                                                    176
113
                                                                 }
114
                                                    177
        int n, m;
        scanf("%d%d", &n, &m);
                                                                 if(!strcmp(s, "MAX-SUM")) {
115
                                                    178
116
                                                    179
                                                                     printf("%d \mid n", mx_sum(t));
117
        Treap* t = NULL;
                                                    180
                                                                 }
        for(int i = 0; i < n; i++) {</pre>
                                                    181
118
                                                            }
119
             int x;
                                                    182
            scanf("%d", &x);
120
                                                    183
                                                            return 0;
                                                    184 }
121
            t = merge(t, new Treap(x));
122
        }
123
124
        while(m--) {
                                                               Leftist Tree
                                                        6.5
            char s[15];
125
             scanf("%s", s);
126
127
                                                      1 #include <bits/stdc++.h>
             Treap *t1 = NULL, *tr = NULL, *t2 =
                                                      2 using namespace std;
128
                 NULL;
                                                      3
```

```
4 struct Left {
                                                    66
                                                                size--;
 5
     Left *1,*r;
                                                    67
                                                                if(size<0)</pre>
 6
     int v,h;
                                                    68
                                                                  bst=bqu=bpq=0;
 7
                                                    69
                                                               if(bst) {
     Left(int v_) : v(v_), h(1), l(0), r(0) {}
 8
                                                    70
                                                                  if(st.top()!=x)
9
                                                    71
                                                                    bst=0;
  int height(Left *p) { return p ? p -> h : 0
10
                                                    72
                                                                  st.pop();
                                                               }
                                                    73
       ; }
                                                               if(bqu) {
11
                                                    74
12
   Left* combine(Left *a,Left *b) {
                                                    75
                                                                  if(qu.front()!=x)
13
     if(!a || !b) return a ? a : b ;
                                                    76
                                                                    bqu=0;
                                                    77
14
     Left *p;
                                                                  qu.pop();
                                                    78
15
     if( a->v > b->v) {
                                                               }
                                                    79
                                                               if(bpq) {
16
       p = a;
17
       p \rightarrow r = combine(p \rightarrow r, b);
                                                    80
                                                               // printf("(%d)\n",top());
                                                                  if(top()!=x)
18
     }
                                                    81
19
     else {
                                                    82
                                                                    bpq=0;
20
                                                    83
       p = b;
                                                                  pop();
21
                                                    84
       p \rightarrow r = combine(p \rightarrow r, a);
                                                               }
                                                             }
22
                                                    85
23
     if( height( p->l ) < height( p->r ) )
                                                    86
                                                           }
24
       swap(p->1, p->r);
                                                    87
                                                           int count=0;
25
     p->h = min( height( p->l ) , height( p->r
                                                           if(bst)
                                                   88
         ) ) + 1;
                                                    89
                                                             count++;
26
     return p;
                                                    90
                                                           if(bqu)
27
  }
                                                    91
                                                             count++;
28 Left *root;
                                                    92
                                                           if(bpq)
29
                                                    93
                                                             count++;
30 void push(int v) {
                                                    94
31
     Left *p = new Left(v);
                                                    95
                                                           if(count>1)
32
     root = combine( root , p );
                                                    96
                                                             puts("not sure");
33 }
                                                    97
                                                           else if(count==0)
34
  int top() { return root? root->v : -1; }
                                                    98
                                                             puts("impossible");
35 void pop() {
                                                    99
                                                           else if(bst)
                                                             puts("stack");
36
     if(!root) return;
                                                  100
37
     Left *a = root->l , *b = root->r ;
                                                  101
                                                           else if(bqu)
38
                                                  102
                                                             puts("queue");
     delete root;
39
     root = combine( a , b );
                                                  103
                                                           else if(bpq)
                                                             puts("priority queue");
40 }
                                                  104
  void clear(Left* &p) {
                                                  105
41
42
     if(!p)
                                                  106
                                                         return 0;
43
       return;
                                                  107 }
44
     if(p->1) clear(p->1);
45
     if(p->r) clear(p->r);
46
     delete p;
                                                       6.6
                                                             Link Cut Tree
47
     p = 0;
48 }
                                                     1 #include <bits/stdc++.h>
49
                                                     2 #define PB push_back
50
  int main() {
51
                                                     3 #define MP make_pair
     int T,n,x,o,size;
                                                    4 #define F first
52
     bool bst,bqu,bpq;
53
     scanf("%d",&T);
                                                     5 #define S second
54
     while(T--) {
                                                    6 #define SZ(x) ((int)(x).size())
                                                    7 #define ALL(x) (x).begin(),(x).end()
55
       bst=bqu=bpq=1;
56
                                                    8 #ifdef DEBUG
       stack<int> st;
57
       queue<int> qu;
                                                    9
                                                         #define debug(...) printf(__VA_ARGS__)
58
       clear(root);
                                                   10 #else
59
                                                    11
                                                         #define debug(...) (void)0
       size=0;
       scanf("%d",&n);
60
                                                    12 #endif
                                                    13 using namespace std;
       while(n--) {
61
         scanf("%d%d",&o,&x);
62
                                                    14 typedef long long 11;
63
         if(o==1)
                                                    15
                                                      typedef pair<int,int> PII;
64
           st.push(x),qu.push(x),push(x),size
                                                    16
                                                      typedef vector<int> VI;
                                                    17
         else if(o==2) {
                                                    18 const int MAXN = 100000 + 10;
65
```

```
19
                                                      int last = 0;
                                                 78
                                                 79
20
  struct SplayTree {
                                                      while(x) {
    int val, mx, ch[2], pa;
                                                 80
21
                                                        splay(x);
22
    bool rev;
                                                 81
                                                        node[x].ch[1] = last;
23
    void init() {
                                                 82
                                                        pull(x);
24
       val = mx = -1;
                                                 83
                                                        last = x;
       rev = false;
25
                                                 84
                                                        x = node[x].pa;
26
       pa = ch[0] = ch[1] = 0;
                                                 85
                                                      }
27
                                                 86
                                                      return last;
28
  } node[MAXN*2];
                                                 87
29
                                                 88
30 inline bool isroot(int x) {
                                                 89
                                                    inline void make_root(int x) {
     return node[node[x].pa].ch[0]!=x && node[
                                                 90
                                                      node[access(x)].rev ^= 1;
31
        node[x].pa].ch[1]!=x;
                                                      splay(x);
32 }
                                                 92 }
33
                                                 93
                                                    inline void link(int x, int y) {
34
  inline void pull(int x) {
                                                 94
35
     node[x].mx = max(node[x].val, max(node[
                                                 95
                                                      make_root(x);
                                                 96
        node[x].ch[0]].mx, node[node[x].ch
                                                      node[x].pa = y;
                                                 97
        [1]].mx));
36 }
                                                 98
37
                                                 99
                                                    inline void cut(int x, int y) {
38
  inline void push(int x) {
                                                100
                                                      make_root(x);
    if(node[x].rev) {
39
                                                101
                                                      access(y);
       node[node[x].ch[0]].rev ^= 1;
40
                                                102
                                                      splay(y);
41
       node[node[x].ch[1]].rev ^= 1;
                                                103
                                                      node[y].ch[0] = 0;
                                                104
42
       swap(node[x].ch[0], node[x].ch[1]);
                                                      node[x].pa = 0;
43
                                                105 }
       node[x].rev ^= 1;
44
                                                106
    }
                                                    inline void cut_parent(int x) {
45|}
                                                107
46
                                                108
                                                      x = access(x);
47
  void push_all(int x) {
                                                109
                                                      splay(x);
48
    if(!isroot(x)) push_all(node[x].pa);
                                                110
                                                      node[node[x].ch[0]].pa = 0;
49
    push(x);
                                                111
                                                      node[x].ch[0] = 0;
50
                                                112
  }
                                                      pull(x);
51
                                                113 }
  inline void rotate(int x) {
                                                114
53
     int y = node[x].pa, z = node[y].pa, d =
                                                115 inline int find_root(int x) {
                                                116
        node[y].ch[1]==x;
                                                      x = access(x);
54
     node[x].pa = z;
                                                117
                                                      while (node[x].ch[0]) x = node[x].ch[0];
55
     if(!isroot(y))
                     node[z].ch[node[z].ch
                                                118
                                                      splay(x);
        [1]==y] = x;
                                                119
                                                      return x;
    node[y].ch[d] = node[x].ch[d^1];
                                                120 }
56
57
    node[node[x].ch[d^1]].pa = y;
                                                121
                                                122 int find_mx(int x) {
58
    node[x].ch[!d] = y;
59
                                                123
                                                      if(node[x].val == node[x].mx) return x;
    node[y].pa = x;
60
                                                124
                                                      return node[node[x].ch[0]].mx==node[x].mx
    pull(y);
61
     pull(x);
                                                           ? find_mx(node[x].ch[0]) : find_mx(
                                                          node[x].ch[1]);
62
                                                125 }
63
64
  void splay(int x) {
                                                126
65
    push_all(x);
                                                127
                                                    inline void change(int x,int b){
66
    while(!isroot(x)) {
                                                128
                                                        splay(x);
67
       int y = node[x].pa;
                                                129
                                                        node[x].data=b;
                                                130
68
       if(!isroot(y)) {
                                                        up(x);
69
         int z = node[y].pa;
                                                131
70
         if((node[z].ch[1]==y) ^ (node[y].ch
                                                132 inline int query_lca(int u,int v){
            [1]==x)) rotate(y);
                                                data 2 2 2 2 2 2 */
71
         else rotate(x);
72
       }
                                                134
                                                      access(u);
73
                                                135
                                                      int lca=access(v);
       rotate(x);
74
     }
                                                136
                                                      splay(u);
75
  }
                                                137
                                                      if(u==lca){
                                                138
                                                         return node[lca].data+node[node[lca].ch
77 inline int access(int x) {
                                                            [1]].sum;
```

}

```
}else{
139
                                                   53|}
140
        return node[lca].data+node[node[lca].ch 54
           [1]].sum+node[u].sum;
                                                   55 int query(int a, int b) {
141
                                                   56
      }
                                                        int res = -1;
142 }
                                                   57
                                                        int ta = link_top[a], tb = link_top[b];
                                                   58
                                                        while(ta != tb) {
                                                   59
                                                          if(dep[ta] < dep[tb]) {</pre>
                                                   60
                                                            swap(a, b);
          Heavy Light Decomposition
                                                   61
                                                            swap(ta, tb);
                                                   62
                                                          }
  1 #include <bits/stdc++.h>
                                                   63
  2 #define PB push_back
                                                   64
                                                          res = max(res, seg->qry(link[ta], link[
  3 #define MP make pair
                                                              a], 1, cnt));
  4 #define F first
                                                   65
                                                          ta = link_top[a=p[ta]];
  5 #define S second
                                                   66
  6 #define SZ(x) ((int)(x).size())
                                                   67
                                                        if(a != b) {
  7 #define ALL(x) (x).begin(),(x).end()
                                                   68
  8 #ifdef _DEBUG_
                                                   69
                                                          if(dep[a] > dep[b]) swap(a, b);
 9
      #define debug(...) printf(__VA_ARGS__)
                                                   70
                                                          a = max_son[a];
 10 #else
                                                   71
                                                          res = max(res, seg->qry(link[a], link[b
 11
      #define debug(...) (void)0
                                                              ], 1, cnt));
                                                   72
 12 #endif
 13 using namespace std;
                                                   73
 14 typedef long long 11;
                                                   74
                                                        return res;
 15 typedef pair<int,int> PII;
                                                   75 }
 16 typedef vector<int> VI;
 17
 18 | const int MAXN = 10000 + 10;
                                                             Disjoint Sets + offline skill
 19
 20 vector<PII> e[MAXN];
 21 int val[MAXN];
                                                    1 #include <bits/stdc++.h>
 22 int sz[MAXN], max_son[MAXN], p[MAXN], dep[
                                                    2 #define PB push_back
                                                    3 #define MP make_pair
 23 int link[MAXN], link_top[MAXN], cnt;
                                                    4 #define F first
                                                    5 #define S second
 24
 25 void find_max_son(int u) {
                                                    6 #define SZ(x) ((int)(x).size())
 26
                                                    7 #define ALL(x) (x).begin(),(x).end()
      sz[u] = 1;
      max\_son[u] = -1;
 27
                                                    8 #ifdef _DEBUG_
      for(int i=0; i<SZ(e[u]); i++) {</pre>
 28
                                                    9
                                                        #define debug(...) printf(__VA_ARGS__)
 29
        PII tmp = e[u][i];
                                                   10 #else
        int v = tmp.F;
                                                        #define debug(...) (void)0
 30
                                                   11
 31
        if(v == p[u]) continue;
                                                   12 #endif
 32
                                                   13 using namespace std;
 33
        p[v] = u;
                                                   14 typedef long long ll;
 34
        dep[v] = dep[u]+1;
                                                   15 typedef pair<int,int> PII;
                                                   16 typedef vector<int> VI;
        val[v] = tmp.S;
 35
        find_max_son(v);
                                                   17
 36
        if(max_son[u]<0 \mid | sz[v]>sz[max_son[u] 18 | const int MAXN = 300000 + 10;
 37
                                                   19
             ]) max_son[u] = v;
 38
                                                   20 bool q[MAXN];
        sz[u] += sz[v];
 39
                                                   21
      }
 40|}
                                                   22 struct DisJointSet {
 41
                                                   23
                                                        int p[MAXN], sz[MAXN], gps;
 42 void build_link(int u, int top) {
                                                   24
                                                        vector<pair<int*, int> > h;
                                                   25
                                                        VI sf;
 43
      link[u] = ++cnt;
 44
                                                   26
      link_top[u] = top;
      if(max_son[u] > 0)
 45
                          build_link(max_son[u
                                                   27
                                                        void init(int n) {
                                                   28
                                                          for(int i=1; i<=n; i++) {</pre>
         ], top);
                                                   29
      for(int i=0; i<SZ(e[u]); i++) {</pre>
 46
                                                            p[i] = i;
 47
                                                   30
                                                            sz[i] = 1;
        PII tmp = e[u][i];
                                                          }
 48
        int v = tmp.F;
                                                   31
 49
        if(v==p[u] || v==max_son[u]) continue;
                                                   32
                                                          gps = n;
 50
                                                   33
 51
        build_link(v, v);
                                                   34
```

35

void assign(int \*k, int v) {

99

int m = (1+r) / 2;

tl->solve(l, m);

```
36
       h.PB(MP(k, *k));
                                                  100
                                                            tr->solve(m+1, r);
37
       *k = v;
                                                  101
38
                                                  102
     }
39
                                                  103
                                                          djs.load();
40
     void save() {
                                                  104
                                                        }
                                                  105|};
41
       sf.PB(SZ(h));
42
                                                  106
43
                                                  107 map<PII, int> prv;
44
     void load() {
                                                  108
45
       int last = sf.back(); sf.pop_back();
                                                  109 int main() {
46
       while(SZ(h) != last) {
                                                  110
                                                        freopen("connect.in", "r", stdin);
47
                                                        freopen("connect.out", "w", stdout);
                                                  111
         auto x = h.back(); h.pop_back();
48
         *x.F = x.S;
                                                  112
49
                                                  113
       }
                                                        int n, k;
                                                        scanf("%d%d \ n", &n, &k);
50
     }
                                                  114
51
                                                  115
                                                        if(!k) return 0;
     int find(int x) {
52
                                                  116
                                                        Seg *seg = new Seg(1, k);
53
       return x==p[x] ? x : find(p[x]);
                                                  117
54
                                                  118
                                                        djs.init(n);
55
                                                  119
                                                        for(int i=1; i<=k; i++) {</pre>
56
     void uni(int x, int y) {
                                                  120
                                                          char op = getchar();
57
       x = find(x), y = find(y);
                                                  121
                                                          if(op == '?') {
58
       if(x == y) return ;
                                                  122
                                                            q[i] = true;
       if(sz[x] < sz[y]) swap(x, y);</pre>
59
                                                  123
                                                            op = getchar();
60
       assign(&sz[x], sz[x]+sz[y]);
                                                  124
                                                          }
       assign(&p[y], x);
                                                          else {
61
                                                  125
                                                  126
62
       assign(&gps, gps-1);
                                                            int u, v;
                                                            scanf("%d%d\n", &u, &v);
                                                  127
63
     }
                                                            if(u > v) swap(u, v);
                                                  128
64|} djs;
                                                            PII eg = MP(u, v);
65
                                                  129
66 struct Seg {
                                                  130
                                                            int p = prv[eg];
                                                  131
67
     vector<PII> es;
                                                            if(p) {
     Seg *tl, *tr;
68
                                                  132
                                                               seg->add(p, i, eg, 1, k);
69
                                                  133
                                                               prv[eg] = 0;
70
                                                  134
     Seg() {}
71
     Seg(int 1, int r) {
                                                  135
                                                            else prv[eg] = i;
72
       if(1 == r) tl = tr = NULL;
                                                  136
                                                          }
73
       else {
                                                  137
74
         int m = (1+r) / 2;
                                                  138
                                                        for(auto p : prv) {
75
         tl = new Seg(1, m);
                                                  139
                                                          if(p.S) {
76
         tr = new Seg(m+1, r);
                                                  140
                                                            seg->add(p.S, k, p.F, 1, k);
77
       }
                                                  141
                                                          }
78
     }
                                                  142
                                                        }
79
                                                  143
     void add(int a, int b, PII e, int 1, int
80
                                                  144
                                                        seg->solve(1, k);
                                                  145
       if(a <= 1 \&\& r <= b) es.PB(e);
                                                  146
81
                                                          return 0;
       else if(b < l || r < a) return ;</pre>
82
                                                  147 }
83
       else {
84
         int m = (1+r) / 2;
         tl->add(a, b, e, l, m);
85
                                                           geometry
86
         tr->add(a, b, e, m+1, r);
87
       }
88
     }
                                                      7.1
                                                             Basic
89
90
     void solve(int 1, int r) {
91
       djs.save();
                                                    1 // correct code of NPSC2013 senior-final pF
92
       for(auto p : es) djs.uni(p.F, p.S);
                                                    2
93
                                                    3 #include <bits/stdc++.h>
94
       if(1 == r) {
                                                    4 #define pb push back
         if(q[1]) printf("%d\n", djs.gps);
95
                                                    5 #define F first
                                                    6 #define S second
96
       }
97
                                                    7
                                                      #define SZ(x) ((int)(x).size())
```

8 #define MP make\_pair

9 using namespace std;

```
10 typedef long long 11;
                                                  64
                                                       return 0;
                                                  65 }
  typedef pair<int,int> PII;
12 typedef vector<int> VI;
13
14 typedef double dou;
                                                            Smallist circle problem
15 struct PT{
16
    dou x,y;
                                                   1 #include <cstdlib>
17
    PT(dou x_{=0.0}, dou y_{=0.0}): x(x_{),y(y_{)} {}
18
    PT operator + (const PT &b) const {
                                                   2
                                                     #include <cstdio>
        return PT(x+b.x,y+b.y); }
                                                     #include <algorithm>
19
    PT operator - (const PT &b) const {
                                                   4
                                                     #include <cmath>
        return PT(x-b.x,y-b.y); }
    PT operator * (const dou &t) const {
                                                   6 using namespace std;
20
        return PT(x*t,y*t); }
                                                   7
    dou operator * (const PT &b) const {
21
                                                   8
                                                     const int N = 1000000 + 10;
        return x*b.x+y*b.y; }
                                                   9
     dou operator % (const PT &b) const {
22
                                                  10
                                                     struct PT {
        return x*b.y-b.x*y; }
                                                       double x, y;
                                                  11
    dou len2() const { return x*x+y*y; }
                                                  12
23
24
    dou len() const { return sqrt(len2()); }
                                                  13
                                                       PT() {}
25|};
                                                  14
                                                       PT(double x, double y):
                                                  15
26
                                                         x(x), y(y) {}
27 const dou INF=1e12;
                                                  16
                                                       PT operator+(const PT &b) const {
                                                  17
28 const dou eps=1e-8;
                                                         return (PT) {x+b.x, y+b.y};
29
  PT inter(const PT &P1,const PT &T1,const PT
                                                  18
       &P2, const PT &T2) // intersection
                                                  19
                                                       PT operator-(const PT &b) const {
30|{
                                                  20
                                                         return (PT) {x-b.x, y-b.y};
    if(fabs(T1%T2)<eps)</pre>
                                                  21
31
       return PT(INF,INF);
                                                  22
                                                       PT operator*(const double b) const {
32
     dou u=((P2-P1)\%T2)/(T1\%T2);
                                                  23
33
                                                         return (PT) {x*b, y*b};
34
     return P1+T1*u;
                                                  24
35 }
                                                  25
                                                       PT operator/(const double b) const {
36
                                                  26
                                                         return (PT) {x/b, y/b};
                                                  27
37
  PT conv[500], cat, to;
                                                  28
                                                       double operator%(const PT &b) const {
38
                                                  29
39 int main()
                                                          return x*b.y - y*b.x;
40 {
                                                  30
41
    int T,N,M;
                                                  31
                                                  32
     scanf("%d",&T);
42
                                                       double len() const {
    while(T--)
                                                  33
43
                                                          return sqrt(x*x + y*y);
44
                                                  34
45
       scanf("%d%d",&N,&M);
                                                  35
                                                       PT T() const {
46
       for(int i=0;i<N;i++)</pre>
                                                  36
                                                         return (PT) {-y, x};
47
         scanf("%lf%lf",&conv[i].x,&conv[i].y)
                                                  37
                                                  38|} p[N];
48
       conv[N]=conv[0];
                                                  39
49
                                                  40
                                                     void update(PT a, PT b, PT c, PT &o, double
       dou ans=0.0;
50
       while(M--)
                                                          &r) {
                                                       if(c.x < 0.0) o = (a+b) / 2.0;
51
                                                  41
       {
                                                       else {
52
         scanf("%lf%lf%lf%lf",&cat.x,&cat.y,&
                                                  42
                                                  43
            to.x,&to.y);
                                                         PT p1 = (a+b)/2.0, p2 = p1 + (b-a).T();
53
         for(int i=0;i<N;i++)</pre>
                                                  44
                                                         PT p3 = (a+c)/2.0, p4 = p3 + (c-a).T();
54
           if(fabs((conv[i]-conv[i+1])%to)>eps
                                                  45
                                                         double a123 = (p2-p1)\%(p3-p1), a124 = (
               )
                                                             p2-p1)%(p4-p1);
55
                                                         if(a123 * a124 > 0.0) a123 = -a123;
                                                  46
               printf("M:%d i=%d\n",M,i);
                                                  47
                                                         else a123 = abs(a123), a124 = abs(a124
56
57
             PT at=inter(conv[i],conv[i]-conv[
                 i+1],cat,to);
                                                  48
                                                         o = (p4*a123 + p3*a124) / (a123 + a124)
             if((conv[i]-at)*(conv[i+1]-at)
58
                 eps && (at-cat)*to>-eps)
                                                  49
59
               ans=max(ans,(cat-at).len());
                                                  50
                                                       r = (a-o).len();
60
           }
                                                  51
61
                                                  52
62
       printf("%.4f \ n",ans);
                                                  53
                                                     int main() {
63
                                                  54
                                                       srand(7122);
     }
```

```
55
                                                             if(b<0)
                                                      8
                                                               a*=-1, b*=-1;
56
     int m, n;
57
     while(scanf("%d%d", &m, &n)) {
                                                      9
58
                                                     10
                                                          Frac(ll a_=0,ll b_=1): a(a_), b(b_) {
       if(!n && !m) return 0;
59
                                                     11
                                                             relax();
60
       for(int i = 0; i < n; i++)</pre>
                                      scanf("%lf%
                                                     12
           lf", &p[i].x, &p[i].y);
                                                     13
                                                          Frac operator + (Frac x) {
                                                     14
61
                                                             relax();
62
       for(int i = 0; i < n; i++)</pre>
                                                     15
                                                             x.relax();
63
         swap(p[i], p[rand() % (i+1)]);
                                                     16
                                                             11 g=__gcd(b,x.b);
64
                                                     17
                                                             11 lcm=b/g*x.b;
65
       PT a = p[0], b = p[1], c(-1.0, -1.0), o 18
                                                             return Frac(a*(lcm/b)+x.a*(lcm/x.b),lcm
            = (a+b) / 2.0;
       double r = (a-o).len();
                                                     19
66
67
       for(int i = 2; i < n; i++) {</pre>
                                                     20
                                                          Frac operator - (Frac x) {
                                                     21
68
         if((p[i]-o).len() <= r) continue;</pre>
                                                             relax();
69
                                                     22
                                                             x.relax();
70
                                                     23
         a = p[i];
                                                             Frac t=x;
71
                                                     24
                                                            t.a*=-1;
         b = p[0];
                                                     25
72
         c = (PT) \{-1.0, -1.0\};
                                                             return *this+t;
73
         update(a, b, c, o, r);
                                                     26
                                                          }
74
         for(int j = 1; j < i; j++) {</pre>
                                                     27
                                                          Frac operator * (Frac x) {
75
            if((p[j]-o).len() <= r) continue;</pre>
                                                     28
                                                             relax();
76
                                                     29
                                                             x.relax();
77
            b = p[j];
                                                     30
                                                             return Frac(a*x.a,b*x.b);
78
            c = (PT) \{-1.0, -1.0\};
                                                     31
79
            update(a, b, c, o, r);
                                                     32
                                                          Frac operator / (Frac x) {
                                                     33
80
                                                             relax();
            for(int k = 0; k < j; k++) {
81
                                                     34
                                                             x.relax();
              if((p[k]-o).len() <= r) continue;</pre>
82
                                                     35
                                                             Frac t=Frac(x.b,x.a);
83
                                                     36
                                                             return (*this)*t;
              c = p[k];
                                                     37
84
                                                          }
85
              update(a, b, c, o, r);
                                                     38|};
86
            }
87
         }
88
       }
89
90
       printf("%.3f \setminus n", r);
91
     }
92 }
```

## 8 Others

#### 8.1 Random

```
1 const int seed=1;
2
3 mt19937 rng(seed);
4 int randint(int lb,int ub) { // [lb, ub]
5 return uniform_int_distribution<int>(lb, ub)(rng);
6 }
```

#### 8.2 Fraction

```
1 struct Frac {
2    ll a,b; // a/b
3    void relax() {
4       ll g=__gcd(a,b);
5       if(g!=0 && g!=1)
6       a/=g, b/=g;
```