# Algorithm Code Book

Tanvir Hasan Anick

July 8, 2015

# Contents

1	Dat	a Structure	3
	1.1	Trie	3
		1.1.1 Static Trie	3
	1.2	RMQ	4
		1.2.1 Bit	4
		1.2.2 Square Root Decomposition	5
		1.2.3 MO's Algorithm	6
		1.2.4 Segment Tree	8
		1.2.5 Sliding Window RMQ	17
		1.2.6 Sparse Table	18
	1.3	Heavy Light Decomposition	18
	1.4	Ternary Bit Mask	25
2	Gra	ph Theory	27
	2.1	DFS	27
		2.1.1 Bicoloring	27
		2.1.2 Cycle Finding	27
	2.2	Topological Sort	27
3	Flo	w networks/ matching	29
	3.1	Max Flow	29
4	Dyı	namic programming	34
	4.1	Edit Distance	34
5	Stri	ngs	39
	5.1	KMP	39
	5.2	Aho Corasick	40
		5.2.1 Aho Corasick with Dynamic Trie	40
		5.2.2 Aho Corasick with Static Trie	42
	5.3	Manacher's Algorithm	45
6	Cor	nputational geometry	46

7	Math	47		
	7.1 Reduce Ratio	47		
8	Number Theory			
	8.1 NCR	48		
	8.1.1 Lucas Theorem	48		

## **Data Structure**

#### 1.1 Trie

#### 1.1.1 Static Trie

```
1 #define Max 10005
1 int getId(char c)
       return c>='a'?c-'a':c-'A'+26;
6 struct Trie
7 {
       struct Tree
9
           int Next[52];
10
           bool word;
11
           void clear()
12
13
                word=false;
14
                memset(Next, -1, sizeof(Next));
16
       T[Max];
17
       int ptr;
18
       void clear()
19
20
           ptr = 1;
21
22
           T[0].clear();
           memset(T[0].Next,0,sizeof(T[0].Next));
23
24
       void Insert(const char *str)
25
26
27
           int p=0;
           for(int i=0;str[i];i++)
28
                int id=getId(str[i]);
30
                if(T[p].Next[id] \le 0)
31
32
                    T[p].Next[id]=ptr;
33
```

```
T[ptr++].clear();
34
35
                  p=T[p]. Next[id];
36
37
             \hat{T}[p]. word=true;
38
39
        bool Search(const char *str)
40
41
42
             int p=0;
             for(int i=0;str[i];i++)
43
44
                  int id=getId(str[i]);
45
                  if (T[p]. Next[id]>0)
46
47
                       p\!\!=\!\!T[\,p\,] . Next [\,i\,d\,]\,;
48
49
                  else return false;
50
             }
51
52
             return T[p].word;
53
54 };
55 Trie A;
```

### 1.2 RMQ

#### 1.2.1 Bit

#### 1D Bit

```
1 #define MaxVal 100000
1 int Bit [MaxVal];
3 /**find sum from 1 to idx**/
4 int read(int idx)
5 {
       int sum = 0;
       while (idx > 0)
7
8
           sum += Bit[idx];
9
10
           idx = (idx \& -idx);
11
       return sum;
12
13 }
/**update value ind to MaxVal**/
void update(int idx ,int val)
16 {
       while (idx <= MaxVal)</pre>
17
18
           Bit [idx] += val;
19
20
           idx += (idx \& -idx);
21
22 }
^{24} /**Find the value of idx**/
25 int readSingle(int idx)
26 {
int sum = Bit[idx]; /// sum will be decreased
```

```
if (idx > 0) /// special case
28
29
           int z = idx - (idx \& -idx); /// make z first
30
           \mathrm{id}x--; /// \mathrm{id}x is no important any more, so instead y, you
31
       can use idx
           while (idx != z) /// at some iteration idx (y) will become
           {
               sum -= Bit[idx];/// substruct Bit frequency which is
34
       between y and "the same path"
               idx = (idx \& -idx);
35
36
37
38
       return sum;
39 }
```

#### 2D Bit

```
void updatey(int x , int y , int val)
2
       while (y \le \max_{y})
3
4
5
           tree[x][y] += val;
           y += (y \& -y);
6
8 }
9 void update(int x , int y , int val)
10 {
       while (x \le \max_{x} x)
11
           updatey(x , y , val); // this function should update array
13
       tree [x]
14
           x += (x \& -x);
15
16 }
```

#### 1.2.2 Square Root Decomposition

```
1 #include < bits / stdc++.h>
2 using namespace std;
s const int sz = 100005;
4 const int inf=(1<<28);
5 template < typename t> t MIN3(t a, t b, t c)
6 {
7
      return min(a, min(b,c));
8 }
9 int BLOCK[400];
10 int arr[sz];
int getId(int indx, int blockSZ)
12 {
      return indx/blockSZ;
13
14 }
void init (int sz)
16 {
      for (int i=0; i \le z; i++)BLOCK[i]=inf;
17
18 }
void update(int val, int indx, int blockSZ)
```

```
20 {
21
       int id=getId(indx, blockSZ);
       BLOCK[id]=min(BLOCK[id], val);
22
23 }
int query(int L, int R, int blockSZ)
25 {
       int lid=getId(L, blockSZ);
26
       int rid=getId(R, blockSZ);
27
       if (lid=rid)
28
29
30
            int ret=inf;
            for(int i=L; i<=R; i++)ret=min(ret, arr[i]);</pre>
31
            return ret;
32
33
       int m1=inf, m2=inf, m3=inf;
34
       for (int i=L; i < (lid+1)*blockSZ; i++)m1=min(m1, arr[i]);
35
       for(int i=lid+1; i<rid; i++)m2=min(m2,BLOCK[i]);</pre>
36
       for (int i=rid*blockSZ; i \le R; i++)m3=min(m3, arr[i]);
37
       return MIN3(m1,m2,m3);
38
39 }
40
  int main()
41
       int N,Q;
42
       scanf ("%d %d",&N,&Q);
43
       int blockSZ=sqrt(N);
44
45
       init (blockSZ);
       for(int i=0; i<N; i++)
46
47
48
            int x;
            scanf("%d",&x);
49
50
            arr[i]=x;
            update(x,i,blockSZ);
51
52
       while (Q--)
53
54
55
            int x, y;
            scanf("%d %d",&x,&y);
56
            printf("%d\n", query(x,y, blockSZ));
57
58
59
       return 0;
60 }
```

#### 1.2.3 MO's Algorithm

```
MO's Algorithm
problem: http://www.spoj.com/problems/DQUERY

MOs algorithm is just an order in which we process the queries

We were given M queries, we will re-order the queries in a particular order and then process them.
Clearly, this is an off-line algorithm. Each query has L and R, we will call them opening and closing.

Let us divide the given input array into Sqrt(N) blocks.
Each block will be N / Sqrt(N) = Sqrt(N) size.
Each opening has to fall in one of these blocks.
```

```
Each closing has to fall in one of these blocks.
12
        All the queries are first ordered in ascending order of their
13
        block number (block number is the block in which its opening
        falls).
        Ties are ordered in ascending order of their R value.
14
15
16 **/
^{17} #include < bits / stdc++.h>
using namespace std;
19 #define Mx 30005
20 #define MxNum 1000005
21 int BlockSize;
22 int Answer;
int Freq[MxNum], Num[Mx];
   struct info
24
25 {
        int L,R,qno;
26
27
        info\left(\begin{array}{cccc} int & L=0, int & R=0, int & qno=0 \end{array}\right) : L(L) \; , R(R) \; , qno\left(\begin{array}{ccc} qno \end{array}\right) \left\{\right\};
        bool operator < (const info &a) const
28
29
             if (L/BlockSize!=a.L/BlockSize)return L/BlockSize<a.L/
30
        BlockSize;
31
            return R<a.R;
32
   } Query [200005];
33
   int StoreAnswer [200005];
34
   void Add(int indx)
35
36
37
        Freq[Num[indx]]++;
38
        if (Freq [Num[indx]]==1) Answer++;
39 }
   void Remove(int indx)
40
41 {
        Freq[Num[indx]] - -;
42
43
        if(Freq[Num[indx]]==0)Answer--;
44 }
45
   int main()
46
47
        int N;
        scanf ("%d",&N);
48
        BlockSize=sqrt(N);
49
50
        for (int i=0; i < N; i++)
             scanf("%d",&Num[i]);
52
53
54
        int Q;
        scanf("%d",&Q);
55
        for(int i=0; i<Q; i++)
56
57
        {
             int x, y;
58
             scanf("%d %d",&x,&y);
59
60
             Query [ i ]= info (x-1,y-1,i);
61
62
        sort (Query , Query+Q) ;
        int currentL=0, currentR=0;
63
        Answer=0;
64
```

```
for (int i=0; i<Q; i++)
65
66
            int L=Query[i].L;
67
            int R=Query[i].R;
68
            while (currentL<L)
69
70
71
                 Remove(currentL);
                 currentL++;
72
            }
73
            while (currentL>L)
74
75
                 Add(currentL-1);
76
                 currentL --;
77
78
            while (current R <=R)
79
80
81
                 Add(currentR);
                 currentR++;
82
83
            while (current R>R+1)
84
85
                 Remove(currentR - 1);
86
                 currentR--;
87
88
            StoreAnswer [Query [i].qno]=Answer;
89
90
        for (int i=0; i<Q; i++)
91
92
            printf("%d\n", StoreAnswer[i]);
93
94
95
        return 0;
96 }
```

#### 1.2.4 Segment Tree

#### Lazy Propagration1

```
_{2} **You are given an array of N elements, which are initially all 0.
      After **that you will be given C commands. They are
  **0 p q v - you have to add v to all numbers in the range **of p to
       q (inclusive), where p and q are two indexes of the array.
  **1 p q - output a line containing a single integer which is the
      sum of all **the array elements between p and q (inclusive)
5 */
6 \#include < bits / stdc++.h>
7 using namespace std;
s typedef long long LLD;
9 LLD tree [3*100005];
10 LLD lazy [3*100005];
  void update(int left,int right,int index,int x,int y,int value)
11
12
       if(x \le left \& xy \ge right)
13
14
           tree[index] += (LLD)(right - left + 1) * value;
15
16
           lazy [index]+=value;
          return;
17
```

```
18
19
       int mid=(left+right)/2;
       if (lazy [index]!=0)
20
21
            tree[2*index]+=(LLD)(mid-left+1)*lazy[index];
22
            tree [2*index+1]+=(LLD) (right-mid)*lazy [index];
23
            lazy [2*index]+=lazy [index];
24
            lazy[2*index+1]+=lazy[index];
25
26
            lazy[index]=0;
27
       if (x<=mid)
28
29
            update(left, mid, 2*index, x, y, value);
30
31
       if (y>mid)
33
34
            update(mid+1,right,2*index+1,x,y,value);
35
36
       tree[index] = tree[2*index] + tree[2*index+1];
37
38
  LLD query (int left, int right, int index, int x, int y)
39
       LLD a1=0, a2=0;
40
       if(x \le left \&\&y \ge right)
41
42
43
            return tree[index];
44
       int mid=(left+right)/2;
45
       if (lazy[index]!=0)
46
47
48
            tree[2*index] += (LLD)(mid-left+1)*lazy[index];
            tree[2*index+1]+=(LLD)(right-mid)*lazy[index];
49
50
            lazy [2*index]+=lazy [index];
            lazy[2*index+1]+=lazy[index];
            lazy[index]=0;
52
53
       if (x<=mid)
54
55
            a1=query(left,mid,2*index,x,y);
56
57
       if (y>mid)
58
59
            a2=query(mid+1,right,2*index+1,x,y);
60
61
       return (a1+a2);
62
63
  int main()
64
65
       int test , t;
66
       scanf("%d",&test);
67
       for(t=1;t<=test;t++)
68
       {
69
70
            memset(tree, 0, size of(tree));
            memset(lazy, 0, sizeof*lazy);
71
72
            int s,q;
            scanf("%d %d",&s,&q);
73
74
            while (q--)
```

```
75
                  int x,y,v,dec;
scanf("%d",&dec);
76
77
                   if (dec)
78
79
                        scanf("%d %d",&x,&y);
80
81
                       LLD ans=query (0, s-1, 1, x-1, y-1);
                        printf("%lld\n",ans);
82
                   }
83
84
                   else
85
                   {
                        scanf("%d %d %d",&x,&y,&v);
86
                        update(0, s-1, 1, x-1, y-1, v);
87
88
             }
89
90
91
        return 0;
92 }
```

#### Lazy Propagration2

```
_{2} **You have an array with n elements which is indexed from 0 to n -
       1. **Initially all elements are zero. Now you have to deal with
       two types of **operations
  **1. Increase the numbers between indices i and j (inclusive) by 1.
      This **is **represented by the command '0 i j'.
  **2. Answer how many numbers between indices i and j (inclusive) are
        **divisible by 3. This is represented by the command '1 i j'.
5 */
6 #include < bits / stdc++.h>
7 using namespace std;
8 #define Max 100010
9 int Tree [8 * Max] [4];
int lazy[8*Max];
11 int temp[4];
void build (int left, int right, int indx)
13
14
       if (left=right)
15
           Tree [indx][0]=1;
16
           Tree [indx][1] = Tree [indx][2] = lazy[indx] = 0;
17
           return;
18
19
       int mid=(left+right)/2;
20
       build(left, mid, 2*indx);
21
       build(mid+1,right,2*indx+1);
22
       for (int i=0; i<3; i++)
23
24
       {
           Tree[indx][i]=Tree[2*indx][i]+Tree[2*indx+1][i];
25
26
27 }
  void update(int left, int right, int indx, int x, int y, int add)
28
29
       if (lazy[indx])
30
31
       {
           int lazy_val=lazy[indx];
32
           lazy[2*indx]=(lazy[2*indx]+lazy_val)\%3;
```

```
lazy [2*indx+1]=(lazy [2*indx+1]+lazy_val)\%3;
34
35
            for (int i=0; i<3; i++)temp [(lazy_val+i)%3]=Tree [indx][i];
            for (int i=0; i <3; i++) Tree [indx][i]=temp[i];
36
            lazy[indx]=0;
37
38
       if (left >y | | right <x) return;</pre>
39
40
        if(x \le left \&\&right \le y)
41
            for (int i = 0; i < 3; i++)
42
43
            {
                 temp[(i+add)%3]=Tree[indx][i];
44
45
            for (int i=0; i<3; i++)Tree [indx][i]=temp [i];
46
47
            lazy [2*indx] = (lazy [2*indx] + add) \%3;
            lazy [2*indx+1]=(lazy [2*indx+1]+add)\%3;
48
49
       int mid=(left+right)/2;
51
52
       update(left, mid,2*indx,x,y,add);
       update(mid+1, right, 2*indx+1, x, y, add);
54
       for (int i=0; i < 3; i++)
            Tree[indx][i]=Tree[2*indx][i]+Tree[2*indx+1][i];
56
57
58 }
  int query(int left, int right, int indx, int x, int y)
59
60
        if (lazy [indx])
61
62
            int lazy_val=lazy[indx];
            lazy[2*indx] = (lazy[2*indx] + lazy_val)\%3;
64
            lazy [2*indx+1]=(lazy [2*indx+1]+lazy_val)%3;
65
            for (int i=0; i <3; i++)temp [(lazy_val+i)%3]=Tree [indx][i];
66
            for (int i=0; i <3; i++) Tree [indx][i]=temp[i];
67
            lazy[indx]=0;
68
69
       if(left>y||right< x)return 0;
70
71
        if(x \le left \& right \le y) return Tree[indx][0];
       int mid = (left + right)/2;
72
73
       return query (left, mid, 2*indx, x, y)+query (mid+1, right, 2*indx+1, x,
       y);
74 }
75
  int main()
76 {
77
        int x,y;
       int test;
78
       scanf("%d",&test);
79
80
       for (int t=1;t \le test;t++)
81
            memset(lazy,0,sizeof(lazy));
82
            int N,Q;
83
            scanf ("%d %d",&N,&Q);
84
            build (0, N-1, 1);
85
            printf("Case \%d: \n",t);
86
            for (int i=0; i<Q; i++)
87
88
                 int d;
89
```

#### Segment Tree Variant 1

```
**Give a array Of N numbers. Finding Maximum cumulative number
       frequency in **the range.
   **input:
 4 **10 4
5 **1 1 1 3 3 3 3 2 2 2
6 **1 5
7 **1 6
 8 **1 7
9 **Output:
10 **3
11 **3
12 **4
13 **2
14 */
#include < bits / stdc++.h>
using namespace std;
17 typedef long long LLD;
18 #define MAX 50005
19 struct info
20 {
        int Lcnt, Rcnt, Max, Lnum, Rnum;
21
        info(int Lcnt=0,int Rcnt=0,int Max=0,int Lnum=0,int Rnum=0):
22
       \texttt{Lcnt}\left(\texttt{Lcnt}\right), \texttt{Rcnt}\left(\texttt{Rcnt}\right), \texttt{Max}(\texttt{Max}), \texttt{Lnum}(\texttt{Lnum}), \texttt{Rnum}(\texttt{Rnum}) \left\{\right\};
23 };
24
  info Tree [3*MAX];
25 int arr [MAX];
info marge(const info &L, const info &R)
27 {
        info ret;
28
29
        if (L.Rnum=R.Lnum)
30
             ret.Max=max(L.Rcnt+R.Lcnt, max(L.Max,R.Max));
31
        else ret.Max=max(L.Max,R.Max);
33
34
        ret .Lnum=L.Lnum;
        ret.Rnum=R.Rnum;
35
        if (L.Lnum=R.Lnum) ret.Lcnt=L.Lcnt+R.Lcnt;
36
37
        else ret.Lcnt=L.Lcnt;
        if (L.Rnum=R.Rnum) ret.Rcnt=L.Rcnt+R.Rcnt;
38
39
        else ret.Rcnt=R.Rcnt;
        return ret;
40
41 }
void build (int L, int R, int indx)
43 {
```

```
if (L=R)
44
45
        {
            Tree [indx] = info (1,1,1, arr [L], arr [R]);
46
47
48
        int mid=(L+R)>>1;
49
        build(L, mid, 2*indx);
50
        build (mid+1,R,2*indx+1);
51
       Tree [indx] = marge (Tree [2*indx], Tree [2*indx+1]);
52
53
  info query (int L, int R, int indx, int x, int y)
54
55
        if (L>=x&&R<=y) return Tree[indx];</pre>
56
57
        int mid=(L+R)>>1;
        info c1, c2;
58
        if(x \le mid) c1 = query(L, mid, 2 * indx, x, y);
59
        if(y>mid)c2=query(mid+1,R,2*indx+1,x,y);
60
        return marge(c1,c2);
61
62 }
  int main()
63
64
        int test;
65
        scanf("%d",&test);
66
        for (int t=1; t \le t est; t++)
67
68
69
            int N,C,Q;
            scanf ("%d %d %d",&N,&C,&Q);
70
            for (int i=0; i< N; i++)
71
72
73
                 int x;
                 scanf("%d",&arr[i+1]);
74
75
            build (1,N,1);
76
            printf("Case %d:\n",t);
77
            while (Q--)
78
79
            {
                 int x,y;
80
                 scanf("%d %d",&x,&y);
81
                 printf("%d\n", query(1,N,1,x,y).Max);
82
83
84
85
        return 0;
86
```

#### Segment Tree Variant 2

```
11 #define MN 50005
   struct info
12
13
       LLD prefixSum;
14
       LLD suffixSum;
15
       LLD Total;
16
       LLD TotalMax;
17
       info(int pre=-Inf,int suff=-Inf,int total=-Inf,int totalmax=-
18
       Inf): prefixSum(pre), suffixSum(suff), Total(total), TotalMax(
       totalmax) {};
19
   info marge(const info &a, const info &b)
20
21
22
       info ret;
       ret. Total=a. Total+b. Total;
23
       ret.prefixSum=max(a.prefixSum,a.Total+b.prefixSum);
24
25
       ret.suffixSum=max(a.suffixSum+b.Total,b.suffixSum);
       ret . TotalMax=MAX3(a. TotalMax, b. TotalMax, a. suffixSum+b. prefixSum
26
       return ret;
27
28
29 LLD arr [MN];
30 info Tree [3*MN];
31
  void build (int L, int R, int indx)
32
33
       i f (L—R)
34
            Tree[indx]=info(arr[L], arr[L], arr[L], arr[L]);
35
36
            return;
37
       int mid=(L+R)>>1;
38
       build (L, mid, 2*indx);
39
       build(mid+1,R,2*indx+1);
40
       Tree[indx] = marge(Tree[2*indx], Tree[2*indx+1]);
41
42 }
43
   void update(int L, int R, int indx, int x, LLD val)
44
45
       i f (L≔R)
46
       {
47
            Tree [indx] = info (val, val, val, val);
48
            return;
49
       int mid=(L+R)>>1;
50
       if(x \le mid)update(L, mid, 2*indx, x, val);
       else update(mid+1,R,2*indx+1,x,val);
52
       Tree [indx] = marge (Tree [2*indx], Tree [2*indx+1]);
53
54 }
  info query (int L, int R, int indx, int x, int y)
55
56
57
       if (L=x and y=R) return Tree [indx];
       int mid=(L+R)>>1;
58
       if (y \le mid) return query (L, mid, 2 * indx, x, y);
59
       else if (x>mid) return query (mid+1,R,2*indx+1,x,y);
60
       return marge(query(L, mid, 2*indx, x, mid), query(mid+1,R, 2*indx+1,
61
       mid+1,y));
62 }
63 int main()
```

```
64 {
65
       #ifdef _ANICK_
        // f_i nput;
66
       #endif // _ANICK_
67
        int N;
68
        scanf("%d",&N);
69
        for (int i=1; i <= N; i++) scanf ("%lld", & arr [i]);
70
        build (1,N,1);
71
72
        int Q;
        scanf("%d",&Q);
73
        while (Q--)
74
75
        {
            int t, x, y;
76
            scanf ("%d %d %d",&t,&x,&y);
77
            if (t) printf ("%lld \n", query (1, N, 1, x, y). TotalMax);
78
            else update(1,N,1,x,y);
79
80
       return 0;
81
```

#### Segment Tree Variant 3

```
**Given a bracket sequence.
_{\rm 3} ** On a bracket word one can do the following operations:
4 **replacement -- changes the i-th bracket into the opposite one
5 **check -- if the word is a correct bracket expression
6 **/
7 #include < bits / stdc++.h>
s using namespace std;
9 typedef long long LLD;
#define MAX 50005
11 struct info
12 {
13
       int sum, sub;
       info(int sum=0,int sub=0):sum(sum),sub(sub){};
14
15 };
info Tree [4*MAX];
  char inp [MAX];
info marge(const info &L, const info &R)
19 {
20
       info ret;
       ret.sum= L.sum+R.sum;
21
22
       ret.sub=L.sub;
       ret.sub=min(ret.sub,L.sum+R.sub);
23
       return ret;
24
25 }
void build (int L, int R, int indx)
  {
27
       i f (L≔R)
28
29
30
           int x;
           if (inp [L] == '(')x=1;
31
32
           else x=-1;
           Tree[indx]=info(x,x);
33
34
           return;
35
      int mid=(L+R)>>1;
```

```
build(L, mid, 2*indx);
37
38
        build(mid+1,R,2*indx+1);
        Tree [indx] = marge (Tree [2*indx], Tree [2*indx+1]);
39
40 }
   void update(int L, int R, int indx, int x)
41
42
43
        if (L=R)
44
45
             int x;
             if (inp [L]=='(')x=1;
46
             else x=-1;
47
             Tree\left[\,in\,d\,x\,\right]\!=\!i\,n\,fo\left(\,x\,,x\,\right)\,;
48
             return;
49
50
        int mid=(L+R)>>1;
        if(x \le mid)update(L, mid, 2 * indx, x);
52
53
        else update (mid+1,R,2*indx+1,x);
        Tree [indx] = marge (Tree [2*indx], Tree [2*indx+1]);
54
55
   info query(int L, int R, int indx, int x, int y)
56
57
        if (L=x&R=y)return Tree[indx];
58
59
        int mid=(L+R)>>1;
60
        if(y \le mid) return query(L, mid, 2 * indx, x, y);
        else if (x>mid) return query (mid+1,R,2*indx+1,x,y);
61
62
        else return marge(query(L, mid, 2*indx, x, mid), query(mid+1,R,2*
        indx+1, mid+1, y));
63 }
   int main()
64
65
66
        int N, t=1;
        while (scanf("%d",&N)==1)
67
        {
68
             scanf("%s", inp);
69
             build (0, N-1, 1);
70
71
             int Q;
             printf("Test %d:\n",t++);
72
             scanf("%d",&Q);
73
             while (Q--)
74
75
76
                  int x;
                  scanf("%d",&x);
77
78
                  if(x)
                  {
79
                        \begin{array}{l} \mbox{if (inp [x-1]=='(')inp [x-1]=')';} \\ \mbox{else inp [x-1]='(';} \end{array} 
80
81
                       update(0, N-1, 1, x-1);
82
83
                  }
                  else
84
85
                  {
                        info y=query(0,N-1,1,0,N-1);
86
                        if(y.sum=0\&\&y.sub>=0)printf("YES\n");
87
88
                        else printf("NO\n");
89
90
91
        return 0;
92
```

93 }

#### 1.2.5 Sliding Window RMQ

```
1 /**
       every K size window RMQ
2
       Calculate in O(N+K) time
3
5 #include < bits / stdc++.h>
6 using namespace std;
7 vector<int>SlidingRMQ(int *A, int N, int k)
8
9
       /** Create a Double Ended Queue, Qi that will store indexes of
      array elements
           The queue will store indexes of useful elements in every
      window and it will
           maintain decreasing order of values from front to rear in
       Qi, i.e.
           arr [Qi.front []] to arr [Qi.rear()] are sorted in increasing
       order
       **/
13
       vector < int > MinWindow;
14
       deque < int > Q;
15
       int i;
16
       /* Process first k (or first window) elements of array */
17
       for (i=0; i < k; i++)
18
19
       {
           /// For very element, the previous largest elements are
20
       useless so
           /// remove them from Qi
21
           while (!Q.empty() \text{ and } A[i] \le A[Q.back()])Q.pop_back();
22
23
           Q. push_back(i);
24
       ^{\prime}/// Process rest of the elements, i.e., from arr[k] to arr[n-1]
25
       while (i <N)
26
27
           /// The element at the front of the queue is the smallest
28
       element of
           /// previous window, so insert it result
29
           MinWindow.push_back(A[Q.front()]);
30
31
           /// Remove the elements which are out of this window
32
           while (!Q.empty() and Q.front()<=i-k)Q.pop_front();
33
34
           /// Remove all elements larger than the currently
35
           /// being added element (remove useless elements)
36
           while (!Q.empty() \text{ and } A[i] \le A[Q.back()])Q.pop\_back();
37
38
           /// Add current element at the rear of Qi
39
           Q. push_back(i);
40
41
42
       /// insert the minimum element of last window
43
      MinWindow.push_back(A[Q.front()]);
44
       return MinWindow;
45
46 }
47 int main()
```

```
 \begin{cases} & \text{int A}[] = \{100, 10, -1, 2, -3, -4, 10, 1, 100, 20\}; \\ & \text{vector} < \text{int} > \text{a} = \text{SlidingRMQ}(A, 10, 2); \\ & \text{for (int i} = 0; \text{i} < \text{a. size ()}; \text{i} + +) \text{cout} << \text{a [i]} << \text{"";} \\ & \text{return 0;} \\ & \text{53} \end{cases}
```

#### 1.2.6 Sparse Table

```
1
                            Compute sparse table in O(NlogN)
   2
                             query in O(1)
   3
                             Ref link: https://www.topcoder.com/community/data-science/data-
                             \verb|science-tutorials/range-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-minimum-query-and-lowest-common-mi
                             ancestor/
  6 #include < bits / stdc++.h>
   7 using namespace std;
   8 \text{ #define Max } 10000005
  9 int rmq[24][Max];
 10 int A[Max];
void Compute_ST(int N)
12 {
                              for (int i = 0; i < N; ++i)rmq[0][i] = i;
 13
                              for (int k = 1; (1 << k) < N; ++k)
 14
                                               for (int i = 0; i + (1 << k) <= N; i++)
 16
 17
                                               {
                                                                \begin{array}{ll} \mbox{int} & x \, = \, rmq \, [\, k \, - \, \, 1\,] \, [\, i \, ]\,; \\ \mbox{int} & y \, = \, rmq \, [\, k \, - \, \, 1\,] \, [\, i \, + \, (\, 1 \, << \, k \, - \, \, 1) \,]\,; \end{array}
 18
 19
                                                                rmq[k][i] = A[x] \le A[y] ? x : y;
20
 21
22
23 }
24
25 int RMQ(int i, int j)
26 {
                              int k = log 2 (j-i);
27
                              \begin{array}{ll} \text{int} & x = rmq[k][i]; \\ \text{int} & y = rmq[k][j-(1 << k) + 1]; \\ \end{array} 
28
29
                              \mathbf{return} \ A[x] <= A[y] \ ? \ x : y;
30
31 }
32
 33
           int main()
34 {
35
                              return 0;
 36
 37 }
```

### 1.3 Heavy Light Decomposition

```
1 /*
2 Tanvir Hasan Anick
3 University of Asia pacific
4 */
5 /**Header file**/
```

```
6 #include < cstdio >
7 #include < iomanip>
8 #include < cstring >
9 #include < cmath >
10 #include < cstdlib >
11 #include < cctype >
12 #include < algorithm >
13 #include < string >
14 #include < vector >
15 #include < queue >
16 #include <map>
17 #include < set >
18 #include < sstream >
19 #include < stack >
20 #include <list >
21 #include <iostream >
22 #include < assert . h>
/**Define file I/O **/
#define f_input freopen("input.txt","r", stdin)
  #define f_output freopen("output.txt","w", stdout)
28 /**Define memory set function**/
<sup>29</sup> #define mem(x,y) memset(x,y, sizeof(x))
30 #define CLEAR(x) memset(x,0,sizeof(x))
32 /**Define function and object**/
33 #define pb push_back
34 #define Sort(v) sort(v.begin(), v.end())
#define RSort(v) sort(v.rbegin(),v.rend())
36 #define CSort(v,C) sort(v.begin(),v.end(),C)
37 #define all(v) (v).begin(),(v).end()
38 #define sqr(x) ((x)*(x))
39 #define find_dist(a,b) sqrt(sqr(a.x-b.x)+sqr(a.y-b.y))
40
/**Define constant value**/
#define ERR 1e-9
43 #define pi (2*acos(0))
44 #define PI 3.141592653589793
46 /**Define input**/
47 #define scanint(a) scanf("%d",&a)
48 #define scanLLD(a) scanf("%lld",&a)
49 #define scanstr(s) scanf("%s",s)
#define scanline(1) scanf(" \%[^{n}]",1);
/**Define Bitwise operation**/
^{53} #define check(n, pos) (n & (1<<(pos)))
^{54} #define biton(n, pos) (n | (1<<(pos)))
55 #define bitoff(n, pos) (n & (1 << (pos)))
/**Define color**/
58 #define WHITE 0
59 #define GREY 1
60 #define BLACK 2
62 /**Sync off with stdio**/
```

```
#define __ cin.sync_with_stdio(false);
              cin.tie();
64
65
  using namespace std;
66
67
  /**Typedef**/
68
  typedef vector <int > vint;
70 typedef vector < vint > vint2D;
71 typedef vector<string> vstr;
72 typedef vector < char > vchar;
73 typedef vector < vchar >vchar2D;
74 typedef queue<int> Qi;
75 typedef queue < Qi > Qii;
76 typedef map<int, int> Mii;
77 typedef map<string , int> Msi;
78 typedef map<int, string > Mis;
79 typedef stack<int> stk;
so typedef pair<int, int> pp;
81 typedef pair<int, pp > ppp;
82 typedef long long int LLD;
   const int inf=0x7FFFFFFF;
**Template & structure**/
  namespace my{
  struct point_int(int x,y; point_int(){} point_int(int a,int b){x=a,y=
       b; }}; ///Point for x,y (int) coordinate in 2D space
   struct \ point\_double \{double \ x,y; point\_double () \{\} point\_double (double \} \}
       a, double b) \{x=a, y=b;\}\}; ///Point for x,y (double) coordinate in
        2D space
89 struct Node{int v,w; Node() {} bool operator < (const Node &a) const{
       return w>a.w;} Node(int _v,int _w) {v=_v,w=_w;}};///Node for
       Diikstra
90 template < class T>T gcd (T a,T b) {return b == 0 ? a : gcd (b, a % b);}
91 template \langle typename \ T \rangle T \ lcm(T \ a, \ T \ b) \{ return \ a \ / \ gcd(a,b) * b; \}
  template < class T>T big_mod(T n,T p,T m) { if (p==0)return (T) 1;T x=
       big_{-mod}(n, p/2, m); x=(x*x)\%m; if (p&1)x=(x*n)\%m; return x;
   template < class T>T multiplication (T n,T p,T m) { if (p==0)return (T) 0;
      T x=multiplication(n,p/2,m); x=(x+x)\%m; if(p&1)x=(x+n)\%m; return x
       ;}
  template < class T>T my_pow(T n,T p) { if (p==0)return 1;T x=my_pow(n,p
94
       /2); x=(x*x); if (p\&1)x=(x*n); return x; ///n to the power p
95 template <class T> double getdist(T a, T b) {return sqrt((a.x - b.x)
        *(a.x - b.x) + (a.y - b.y) * (a.y - b.y); /// distance
       between a & b
   template <class T> T extract(string s, T ret) {stringstream ss(s);
       ss >> ret; return ret;}/// extract words or numbers from a line
  template <class T> string tostring (T n) {stringstream ss; ss << n;
       return ss.str();}/// convert a number to string
   template < class T> inline T Mod(T n,T m) {return (n/m+m)/m;} ///For
98
       Positive Negative No.
   template < class T> T MIN3(T a,T b,T c) {return min(a,min(b,c));} ///
        minimum of 3 number
  template < class T> T MAX3(T a, T b, T c) {return max(a, max(b, c));} ///
       maximum of 3 number
  template <class T> void print_vector(T &v){int sz=v.size(); if(sz)
       cout << v[0]; for(int i = 1; i < sz; i++)cout << '' '<< v[i]; cout <<
       endl; \} /// prints all elements in a vector
```

```
bool isVowel(char ch){ ch=toupper(ch); if(ch='A'||ch='U'||ch='I'
        ||ch='O'||ch='E') return true; return false;}
   bool isConsonant(char ch){if (isalpha(ch) && !isVowel(ch)) return
103
        true; return false;}}
   /**Shortcut input function **/
104
   int read_int(){int n; scanf("%d",&n); return n;}
int read_LLD(){LLD n; scanf("%lld",&n); return n;}
105
   inline int buffer_input() { char inp[1000]; scanstr(inp); return
107
        atoi(inp); }
108
109
   /**Direction**/
   \frac{1}{1}/\frac{1}{1} int col[8] = {0, 1, 1, 1, 0, -1, -1, -1}; int row[8] = {1, 1, 0,
        -1, -1, -1, 0, 1; ///8 Direction
   ///int \ col[4] = \{1, 0, -1, 0\}; int \ row[4] = \{0, 1, 0, -1\}; ///4
        Direction
   ///int dx[]={2,1,-1,-2,-2,-1,1,2};int dy
        []=\{1,2,2,1,-1,-2,-2,-1\};//Knight Direction
   ///int dx[]=\{-1,-1,+0,+1,+1,+0\};int dy[]=\{-1,+1,+2,+1,-1,-2\}; ///
       Hexagonal Direction
115
    '******Ajaira Jinish Sesh
116
   const int Max=10000;
118
   struct info
119
120
121
        int v, cost;
        info(int v=0,int cost=0):v(v),cost(cost)\{\};
123 };
   vector <pp>edges;
124
   \verb|vector| < \verb|info| > Graph[Max + 5];
125
   int Tree[5*Max+5], BaseArray[Max+5], SubTreeSize[Max+5];
126
   int ChainHead [Max+5], ChainNum [Max+5], PosInBaseArray [Max+5], ChainNo;
127
   int Level [Max+5], Parent [Max+5], SparseTable [Max+5][16];
128
   int ptr;
129
   void init (int N)
130
131
        for (int i = 0; i \le N; i++)
133
        {
            Graph [i]. clear(), ChainHead [i]=-1;
134
            for (int j=0; j <=15; j++) Sparse Table [i] [j]=-1;
135
136
        edges.clear();
        ptr=ChainNo=0;
138
139
   }
   void buildSegmentTree(int l, int r, int indx)
140
141
        if ( l==r )
142
143
        {
            Tree[indx]=BaseArray[l];
144
            return;
145
146
        int mid=(l+r)>>1;
147
148
        int lindx = indx << 1;
        int rindx=lindx | 1;
149
        buildSegmentTree(l, mid, lindx);
```

```
buildSegmentTree(mid+1,r,rindx);
152
        Tree [indx]=max(Tree [lindx], Tree [rindx]);
153
   void updateSegmentTree(int l, int r, int indx, int update_indx, int
154
        value)
155
156
        if ( l==r )
157
            Tree [indx]=value;
158
159
             return;
        int mid=(l+r)>>1;
161
        int lindx=indx <<1;</pre>
163
        int rindx=lindx | 1;
        if (update_indx <= mid) updateSegmentTree(l, mid, lindx, update_indx,</pre>
        value);
165
        else updateSegmentTree(mid+1,r,rindx,update_indx,value);
        Tree [indx] = max(Tree [lindx], Tree [rindx]);
167
       querySegmentTree(int l, int r, int indx, int x, int y)
   int
168
169
        if(l>y||r<x)return 0;
        if (x<=l&&y>=r) return Tree[indx];
        int mid=(l+r)>>1;
172
        int lindx=indx << 1;
173
174
        int rindx=lindx | 1;
        int c1=0, c2=0;
        if (x<=mid) c1=querySegmentTree(l, mid, lindx, x, y);</pre>
176
        if (y>mid) c2=querySegmentTree (mid+1,r,rindx,x,y);
177
        return max(c1,c2);
178
179
   void dfs (int from, int u, int depth)
180
181
        Level[u]=depth;
182
        Parent [u]=from;
183
184
        SubTreeSize[u]=1;
        int sz=Graph[u].size();
185
186
        for (int i=0; i < sz; i++)
187
             int v=Graph[u][i].v;
188
             if (v==from)continue;
189
             dfs(u,v,depth+1);
190
191
            SubTreeSize [u]+=SubTreeSize [v];
193
   void sparseTable(int N)
195
        for (int i=0; i \le N; i++) Sparse Table [i][0] = Parent[i];
196
        for (int j=1;(1<< j)<=N; j++)
197
198
             for(int i=0; i=N; i++)
199
             {
200
                 if(SparseTable[i][j-1]!=-1)
201
202
203
                      int a=SparseTable[i][j-1];
                      SparseTable[i][j]=SparseTable[a][j-1];
204
205
```

```
206
207
208
   int LCA(int p, int q)
209
210
        if(Level[p] < Level[q])swap(p,q);
211
        int Log=log 2 (Level[p]) + 1;
212
       for (int i=Log; i>=0; i--)
213
214
            if (( Level[p]-(1<<i))>=Level[q]) p=SparseTable[p][i];
215
216
        if(p=q)return p;
217
       for (int i=Log; i>=0; i--)
218
219
            if (SparseTable [p][i]!=-1&&SparseTable [p][i]!=SparseTable [q
       ][i])
                p=SparseTable[p][i],q=SparseTable[q][i];
222
223
       return Parent[p];
225
226
227
228
    * Actual HL-Decomposition part
      Initially all entries of chainHead [] are set to -1.
229
      So when ever a new chain is started, chain head is correctly
       assigned.
     As we add a new node to chain, we will note its position in the
231
       baseArray.
    * In the first for loop we find the child node which has maximum
       sub-tree size.
    * The following if condition is failed for leaf nodes.
233
    * When the if condition passes, we expand the chain to special
234
       child.
    * In the second for loop we recursively call the function on all
       normal nodes.
    * chainNo++ ensures that we are creating a new chain for each
       normal child.
237
    **/
   void heavyLightDecompositon(int from, int curNode, int cost)
238
239
        if (ChainHead [ChainNo]==-1)ChainHead [ChainNo]=curNode; ///
240
       Assign chain head
       ChainNum [curNode]=ChainNo;
241
       PosInBaseArray[curNode]=ptr; /// Position of this node in
242
       baseArray which we will use in Segtree
       BaseArray [ptr++]=cost;
243
244
       int sc=-1, nextCost;
       int sz=Graph[curNode].size();
245
        for(int i=0; i < sz; i++) /// Loop to find special child
246
247
            int v=Graph[curNode][i].v;
248
249
            if (v=from) continue;
            if(sc == -1 || SubTreeSize[sc] < SubTreeSize[v])
251
            {
                nextCost=Graph[curNode][i].cost;
```

```
254
255
                    if (sc!=-1)heavyLightDecompositon(curNode, sc, nextCost); ///
256
                   Expand the chain
                    for(int i=0; i < sz; i++)
257
258
259
                               int v=Graph[curNode][i].v;
                               int cost=Graph[curNode][i].cost;
260
                               if (v==from | | sc==v) continue;
261
                              ChainNo++;
262
                               heavyLightDecompositon(curNode, v, cost);
263
264
265 }
        void updateTree(int ith, int val)
266
267
                   pp a=edges[ith];
268
269
                    int u=a.first ,v=a.second;
                    int indx=PosInBaseArray[u];
270
271
                    if (Level[u]<Level[v]) indx=PosInBaseArray[v];
                    updateSegmentTree(0,ptr-1,1,indx,val);
272
273
274
275
           * query_up:
276
           * It takes two nodes u and v, condition is that v is an ancestor
                   of u
           * We query the chain in which u is present till chain head, then
                  move to next chain up
            * We do that way till u and v are in the same chain, we query for
                   that part of chain and break
279
         int queryUp(int u, int v)
280
281
                    if(u=v)return 0;
282
                    int uchain, vchain=ChainNum[v], ans=-1;
283
                    while (true)
284
285
                               uchain=ChainNum[u];
286
287
                               if (uchain=vchain)
288
                                                                                     /// Both u and v are in the same chain,
                                          i f ( u==v )
                      so we need to query from u to v, update answer and break.
                                                                                     /// We break because we came from u up
                                                     break;
290
                    till v, we are done
                                         ans\!\!=\!\!max(\,ans\,,querySegmentTree\,(\,0\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBaseArray\,[\,v\,,ptr\,-1\,,1\,,PosInBase
291
                   ]+1,PosInBaseArray[u]));
                                         break;
293
                               int uchainhead=ChainHead[uchain];
294
                              ans=max(ans, querySegmentTree(0, ptr-1,1, PosInBaseArray[
295
                    uchainhead], PosInBaseArray[u]));
                                                                /// Above is call to segment tree query
296
                    function. We do from chainHead of u till u. That is the whole
                   chain from
                              u=Parent [uchainhead];
297
298
                    return ans;
300
```

```
int queryTree(int u,int v)
302
        int lca=LCA(u,v);
303
        return max(queryUp(u,lca),queryUp(v,lca));
304
305 }
   int main()
306
307
        #ifdef _ANICK_
308
309
        //f_input;
        #endif // _ANICK_
310
311
        int test;
312
        cin>>test;
313
        while (test --)
314
315
             int N;
316
317
             cin >> N;
             init (N);
318
319
             for (int i=0; i< N-1; i++)
320
321
                  int u, v, c;
                  \verb|cin>> \verb|u>> \verb|v>> c;
322
323
324
                  Graph[u].pb(info(v,c));
                  Graph[v].pb(info(u,c));
325
326
                  edges.pb(pp(u,v));
327
             dfs(-1,0,0);
328
             sparseTable(N);
329
             heavyLightDecompositon(-1,0,-1);
330
331
             buildSegmentTree (0, ptr -1, 1);
332
             string ch;
             int x,y;
333
             while(true)
334
335
336
                  cin>>ch;
                  if (ch[0]== 'D') break;
337
338
                  cin>>x>>y;
                  if (ch[0] == Q') printf (\%d n'), queryTree (x-1,y-1);
339
                  else if (ch[0] == 'C') updateTree (x-1,y);
340
341
342
        return 0;
343
344 }
```

### 1.4 Ternary Bit Mask

```
int more_bit[10];
int get_bit(int mask , int pos)

{
    return (mask / more_bit[pos]) % 3;

}
int set_bit(int mask, int pos , int bit)

{
    int tmp = (mask / more_bit[pos]) % 3;
    mask -= tmp * more_bit[pos];
}
```

```
mask += bit * more_bit[pos];
return mask;

void init(void)

more_bit[0] = 3;
for(int i = 1; i < 10; i++) more_bit[i] = 3 * more_bit[i - 1];

}</pre>
```

# Graph Theory

#### 2.1 DFS

#### 2.1.1 Bicoloring

```
/// color will be initial with -1
int color [20005];
bool dfs(int u, int c)

{
    if (color [u]==c) return true;
    if (color [u]==(1-c)) return false;
    color [u]=c;
    bool ret=true;
    for (auto v: graph [u]) ret&=dfs(v,1-c);
    return ret;
}
```

#### 2.1.2 Cycle Finding

```
1 int color [20005];
2 bool dfs(int u)
3 {
       {\tt color}\,[\,u]\!\!=\!\!\!G\!R\!E\!Y;
       bool no_cycle=true;
        for (auto v:graph[u])
             if ( color [ v]==WHITE)
9
                 no\_cycle=dfs(v);
10
11
            else if(color[v]==GREY)return false;
12
13
       color[u]=BLACK;
14
       return no_cycle;
15
16 }
```

### 2.2 Topological Sort

```
_{1} #include < bits / stdc++.h>
using namespace std;
3 #define WHITE 0
4 #define GREY 1
5 #define BLACK 2
^{6}\ \ vector{<} int{>}\ graph [100005];
vector < int > ans;
8 int visit [100005];
9 bool dfs(int u)
10 {
       visit [u]=GREY;
11
       bool no_cycle=true;
12
       int sz=graph[u].size();
13
       for (int i=0; i < sz; i++)
14
15
            int v=graph[u][i];
16
            if (visit [v]==WHITE)
17
            {
18
19
                no\_cycle=dfs(v);
20
21
            else if (visit [v]==GREY) return false;
22
       visit [u]=BLACK;
23
24
       ans.push_back(u);
       return no_cycle;
25
26
bool topsort (int N)
28 {
       ans.clear();
29
       memset(visit, false, sizeof(visit));
30
31
       int no_cycle=true;
       for (int i=0; i< N; i++)
32
33
       {
            if(visit[i]==WHITE)no\_cycle\&=dfs(i);
34
35
36
       return no_cycle;
37 }
38
  int main()
39
40
       #ifdef _ANICK_
41
       //f_input;
42
       #endif // _ANICK_
       return 0;
43
44 }
```

# Flow networks/ matching

#### 3.1 Max Flow

```
Tanvir Hasan Anick
2
       University of Asia pacific
4 */
5 /**Header file**/
6 #include < cstdio >
7 #include < iomanip>
8 #include < cstring >
9 #include < cmath >
10 #include < cstdlib >
11 #include < cctype >
12 #include < algorithm >
13 #include < string >
14 #include < vector >
15 #include <queue>
16 #include <map>
17 #include < set >
18 #include < sstream >
19 #include < stack >
20 #include < list >
21 #include <iostream >
22 #include < assert . h>
/**Define file I/O **/
#define f_input freopen("input.txt","r", stdin)
#define f_output freopen("output.txt","w",stdout)
/**Define memory set function **/
29 #define mem(x,y) memset(x,y, size of(x))
30 #define CLEAR(x) memset(x,0,sizeof(x))
32 /**Define function and object**/
33 #define pb push_back
34 #define Sort(v) sort(v.begin(), v.end())
35 #define RSort(v) sort(v.rbegin(),v.rend())
#define CSort(v,C) sort(v.begin(),v.end(),C)
```

```
\#define \ all(v)(v).begin(),(v).end()
  #define sqr(x) ((x)*(x))
\#define\ find_dist(a,b)\ sqrt(sqr(a.x-b.x)+sqr(a.y-b.y))
/**Define constant value**/
42 #define ERR 1e-9
43 #define pi (2*acos(0))
44 #define PI 3.141592653589793
/**Define input**/
47 #define scanint(a) scanf("%d",&a)
48 #define scanLLD(a) scanf("%lld",&a)
#define scanstr(s) scanf("%s",s)
50 #define scanline(1) scanf(" \%[^{n}]",1);
52 /**Define Bitwise operation**/
^{53} #define check(n, pos) (n & (1<<(pos)))
#define biton(n, pos) (n | (1 << (pos)))
55 #define bitoff(n, pos) (n & (1 << (pos)))
  /**Define color**/
#define WHITE 0
59 #define GREY 1
60 #define BLACK 2
61
  /**Sync off with stdio**/
63 #define __ cin.sync_with_stdio(false);\
64
             cin.tie();
/**Debug tools**/
66 #define what_is(x) cerr << (#x) << " is " << x << endl
67 using namespace std;
/**Typedef**/
70 typedef vector<int> vint;
71 typedef vector< vint > vint2D;
72 typedef vector < string > vstr;
73 typedef vector < char > vchar;
74 typedef vector< vchar >vchar2D;
75 typedef queue<int> Qi;
76 typedef queue< Qi > Qii;
77 typedef map<int , int> Mii;
78 typedef map<string , int> Msi;
79 typedef map<int, string> Mis;
so typedef stack<int> stk;
81 typedef pair<int, int> pp;
typedef pair<int, pp > ppp;
83 typedef long long int LLD;
84 const int inf=0x7FFFFFF;
**Template & structure**/
  struct point_int{int x,y; point_int(){} point_int(int a,int b){x=a,y=
      b; }}; ///Point for x,y (int) coordinate in 2D space
  struct point_double { double x,y; point_double() {} } point_double ( double
      a, double b) \{x=a, y=b;\}\}; ///Point for x,y (double) coordinate in
       2D space
89 struct Node{int v,w; Node() {} bool operator < (const Node &a) const{
   return w>a.w;} Node(int _v,int _w) {v=_v,w=_w;}}; ///Node for
```

```
Dijkstra
 90 namespace my{
91 template < class T>T gcd(T a, T b) {return b == 0 ? a : gcd(b, a % b);}
 92 template < typename T>T lcm(T a, T b) {return a / gcd(a,b) * b;}
        template < class T>T big_mod(T n,T p,T m){if(p==0)return (T)1;T x=
                     big_{-mod(n,p/2,m)}; x=(x*x)\%m; if(p\&1)x=(x*n)\%m; return x;
          \begin{array}{ll} \textbf{template} < \textbf{class} & \textbf{T} > \textbf{T} & \textbf{multiplication} & (\textbf{T} & \textbf{n}, \textbf{T} & \textbf{p}, \textbf{T} & \textbf{m}) \\ \textbf{if} & (\textbf{p} == \textbf{0}) & \textbf{return} & (\textbf{T}) & \textbf{0}; \end{array} 
                    T x=multiplication(n,p/2,m); x=(x+x)\%m; if(p&1)x=(x+n)\%m; return x
                     ;}
         template < class T>T my_pow(T n,T p){if(p==0)return 1;T x=my_pow(n,p
        /2);x=(x*x); if (p\&1)x=(x*n); return x; ///n to the power p template <class T> double getdist(T a, T b){return sqrt((a.x - b.x))}
                      *(a.x - b.x) + (a.y - b.y) *(a.y - b.y); /// distance
                     between a & b
        template <class T> T extract(string s, T ret) {stringstream ss(s);
                     ss >> ret; return ret;}/// extract words or numbers from a line
         template < class T> string tostring (T n) {stringstream ss; ss << n;
                     return ss.str();}/// convert a number to string
         template < class T > inline T Mod(T n,T m) {return (n/m+m)/m;} ///For
                     Positive Negative No.
         template < class T> T MIN3(T a, T b, T c) {return min(a, min(b, c));} ///
                        minimum of 3 number
         template < class T> T MAX3(T a,T b,T c) {return max(a,max(b,c));} ///
                     maximum of 3 number
         template <class T> void print_vector(T &v){int sz=v.size(); if(sz)
102
                     cout << v[0]; for(int i = 1; i < sz; i++)cout << ' '<< v[i]; cout <<
                     endl;}/// prints all elements in a vector
bool isVowel(char ch) { ch=toupper(ch); if(ch='A'||ch='U'||ch='I'
                     ||ch='O'||ch='E') return true; return false;}
         bool isConsonant(char ch){if (isalpha(ch) && !isVowel(ch)) return
104
                      true; return false;}}
          /**Shortcut input function **/
105
        int read_int(){int n; scanf("%d",&n); return n;}
106
        int read_LLD(){LLD n; scanf("%lld",&n); return n;}
107
         inline int buffer_input() { char inp[1000]; scanstr(inp); return
108
                     atoi(inp); }
110
         /**Direction**/
         ///int \ col[8] = \{0, 1, 1, 1, 0, -1, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1, -1\}; int \ row[8] = \{1, 1, 0, -1\}; int \ row
111
                      -1, -1, -1, 0, 1; ///8 Direction
         ///int col[4] = \{1, 0, -1, 0\}; int row[4] = \{0, 1, 0, -1\}; ///4
                     Direction
         ///int dx[] = \{2,1,-1,-2,-2,-1,1,2\}; int dy
                     [] = \{1, 2, 2, 1, -1, -2, -2, -1\}; /// Knight Direction
         ///int dx[] = \{-1, -1, +0, +1, +1, +0\}; int dy[] = \{-1, +1, +2, +1, -1, -2\}; ///int dx[] = \{-1, -1, +0, +1, +1, +0\}; int dx[] = \{-1, -1, +0, +1, +1, +0\}; int dx[] = \{-1, +1, +2, +1, -1, -2\}; ///int dx[] = \{-1, +1, +2, +1, -1, -2\}; ///int dx[] = \{-1, +1, +2, +1, -1, -2\}; ///int dx[] = \{-1, +1, +2, +1, -1, -2\}; ///int dx[] = \{-1, +1, +2, +1, -1, -2\}; ///int dx[] = \{-1, +1, +2, +1, -1, -2\}; ///int dx[] = \{-1, +1, +2, +1, -1, -2\}; ///int dx[] = \{-1, +1, +2, +1, -1, -2\}; ///int dx[] = \{-1, +1, +2, +1, -1, -2\}; ///int dx[] = \{-1, +1, +2, +1, -1, -2\}; ///int dx[] = \{-1, +1, +2, +1, -1, -2\}; ///int dx[] = \{-1, +1, +2, +1, -1, -2\}; ///int dx[] = \{-1, +1, +2, +1, -1, -2\}; ///int dx[] = \{-1, +1, +2, +1, -1, -2\}; ///int[] = \{-1, +1, +2, +1, -2\}; ///int[] = \{-1, +1, +2, +1, -2\}; ///int[] = \{-1, +1, +2\}; //int[] = \{-1, +2\}; //int[] = 
114
                     Hexagonal Direction
                                                                                ****** Ajaira Jinish Sesh
117
        #define MN 1000
118
vint2D graph;
120 int Cost [MN] [MN];
int parent [MN+5];
        int flow;
void init (int N)
124 {
```

```
graph=vint2D(N);
125
126
        mem(Cost, 0);
127
   void AddEdge(int u,int v,int cost)
128
129
        graph [u].pb(v);
130
        graph [v].pb(u);
Cost [u][v]+=cost;
131
        Cost[v][u]+=cost;
133
134
   bool augmenting_path (int source, int sink)
135
136
        mem(parent, -1);
137
138
        queue<int>Q;
        Q. push (source);
139
        while (!Q. empty())
140
141
             int u=Q.front();
142
143
             Q. pop();
             int sz=graph[u].size();
144
145
             for (int i=0; i < sz; i++)
146
                  int v=graph[u][i];
147
                  if(parent[v]==-1 and Cost[u][v]>0)
148
149
150
                       parent [v]=u;
                      Q.push(v);
151
                       if (v==sink)return true;
152
153
154
155
        return false;
156
157 }
   void path(int v, int source)
158
159
160
        int u=parent[v];
        flow=min(flow, Cost[u][v]);
161
162
        if (source!=u) path(u, source);
        \mathrm{Cost}\,[\,u\,]\,[\,v] -= flow\;;
163
164
        Cost[v][u]+=flow;
165
        return;
166
   int max_flow(int source, int sink)
167
168
        int ret = 0;
169
        while (augmenting_path (source, sink))
171
172
             flow=inf;
             path(sink, source);
174
             ret+=flow;
        return ret;
176
177 }
178
   int main()
179
   {
        #ifdef _ANICK_
180
181
        //f_input;
```

```
#endif // _ANICK_
182
              int test;
scanf("%d",&test);
183
184
185
               while (test --)
186
                       187
189
                       \begin{array}{ll} \textbf{int} & \textbf{superSource} \!=\! 0, \\ \textbf{SuperSikn} \!\!=\! \!\! P \!\!+\! S \!+\! 1; \end{array}
190
                       \begin{array}{ll} \textbf{for} (\, \textbf{int} \quad i = 1; i < = P; \, i + +) \\ \textbf{AddEdge} (\, \textbf{superSource} \,\,, \, i \,\,, 1) \,\,; \end{array}
191
                       \begin{array}{ll} & \text{for (int } i=1; i <= S; i++) \\ \text{AddEdge(P+1, SuperSikn, C);} \\ & \text{for (int } i=0; i < M; i++) \\ \end{array}
192
193
194
                                int x,y;
scanf("%d %d",&x,&y);
195
196
197
                                \operatorname{AddEdge}\left(\left.x\right.,P\!\!+\!\!y\left.,\left(1\!<\!<\!30\right)\right.\right);
198
                       printf("%d\n", max\_flow(superSource, SuperSikn));
199
200
               return 0;
201
202 }
```

# Dynamic programming

#### 4.1 Edit Distance

```
Tanvir Hasan Anick
2
       University of Asia pacific
4 */
5 /**Header file**/
6 #include < cstdio >
7 #include < iomanip>
8 #include < cstring >
9 #include < cmath >
10 #include < cstdlib >
#include < cctype >
12 #include < algorithm >
13 #include < string >
#include < vector >
15 #include <queue>
16 #include <map>
17 #include < set >
18 #include < sstream >
19 #include < stack >
20 #include <list >
21 #include <iostream >
22 #include < assert . h>
/**Define file I/O **/
25 #define f_input freopen("input.txt","r", stdin)
#define f_output freopen("output.txt","w",stdout)
/**Define memory set function **/
\# define mem(x,y) memset(x,y,sizeof(x))
30 #define CLEAR(x) memset(x,0,sizeof(x))
32 /**Define function and object**/
33 #define pb push_back
34 #define Sort(v) sort(v.begin(), v.end())
\#define RSort(v) sort(v.rbegin(),v.rend())
#define CSort(v,C) sort(v.begin(),v.end(),C)
```

```
\#define \ all(v)(v).begin(),(v).end()
  #define sqr(x) ((x)*(x))
\#define\ find_dist(a,b)\ sqrt(sqr(a.x-b.x)+sqr(a.y-b.y))
/**Define constant value**/
42 #define ERR 1e-9
43 #define pi (2*acos(0))
44 #define PI 3.141592653589793
/**Define input**/
47 #define scanint(a) scanf("%d",&a)
48 #define scanLLD(a) scanf("%lld",&a)
#define scanstr(s) scanf("%s",s)
50 #define scanline(1) scanf(" \%[^{n}]",1);
52 /**Define Bitwise operation**/
^{53} #define check(n, pos) (n & (1<<(pos)))
#define biton(n, pos) (n | (1 << (pos)))
55 #define bitoff(n, pos) (n & (1 << (pos)))
  /**Define color**/
#define WHITE 0
59 #define GREY 1
60 #define BLACK 2
61
  /**Sync off with stdio**/
63 #define __ cin.sync_with_stdio(false);\
64
             cin.tie();
/**Debug tools**/
66 #define what_is(x) cerr << (#x) << " is " << x << endl
67 using namespace std;
69 /**Typedef**/
70 typedef vector<int> vint;
71 typedef vector< vint > vint2D;
72 typedef vector < string > vstr;
73 typedef vector < char > vchar;
74 typedef vector< vchar >vchar2D;
75 typedef queue<int> Qi;
76 typedef queue< Qi > Qii;
77 typedef map<int , int> Mii;
78 typedef map<string , int> Msi;
79 typedef map<int, string> Mis;
so typedef stack<int> stk;
81 typedef pair<int, int> pp;
typedef pair<int, pp > ppp;
83 typedef long long int LLD;
84 const int inf=0x7FFFFFF;
**Template & structure**/
  struct point_int{int x,y; point_int(){} point_int(int a,int b){x=a,y=
      b; }}; ///Point for x,y (int) coordinate in 2D space
  struct point_double { double x,y; point_double() {} } point_double ( double
      a, double b) \{x=a, y=b;\}\}; ///Point for x,y (double) coordinate in
       2D space
89 struct Node{int v,w; Node() {} bool operator < (const Node &a) const{
   return w>a.w;} Node(int _v,int _w) {v=_v,w=_w;}}; ///Node for
```

```
Dijkstra
90 namespace my{
91 template < class T>T gcd(T a, T b) {return b == 0 ? a : gcd(b, a % b);}
92 template < typename T>T lcm(T a, T b) {return a / gcd(a,b) * b;}
   template < class T>T big_mod(T n,T p,T m) { if (p==0)return (T) 1;T x=
        big_{-mod(n,p/2,m)}; x=(x*x)\%m; if(p\&1)x=(x*n)\%m; return x;
    \begin{array}{ll} \textbf{template} < \textbf{class} & \textbf{T} > \textbf{T} & \textbf{multiplication} & (\textbf{T} & \textbf{n}, \textbf{T} & \textbf{p}, \textbf{T} & \textbf{m}) \\ \textbf{if} & (\textbf{p} == \textbf{0}) & \textbf{return} & (\textbf{T}) & \textbf{0} \\ \end{array} 
       T x=multiplication(n,p/2,m); x=(x+x)\%m; if(p&1)x=(x+n)\%m; return x
        ;}
   template < class T>T my_pow(T n,T p){if(p==0)return 1;T x=my_pow(n,p
   /2);x=(x*x); if (p\&1)x=(x*n); return x; ///n to the power p template <class T> double getdist(T a, T b){return sqrt((a.x - b.x))}
        *(a.x - b.x) + (a.y - b.y) *(a.y - b.y); /// distance
        between a & b
   template <class T> T extract(string s, T ret) {stringstream ss(s);
        ss >> ret; return ret;}/// extract words or numbers from a line
   template < class T> string tostring (T n) {stringstream ss; ss << n;
        return ss.str();}/// convert a number to string
   template < class T > inline T Mod(T n,T m) {return (n/m+m)/m;} ///For
        Positive Negative No.
   template < class T> T MIN3(T a, T b, T c) {return min(a, min(b, c));} ///
         minimum of 3 number
   template < class T> T MAX3(T a,T b,T c) {return max(a,max(b,c));} ///
       maximum of 3 number
   template <class T> void print_vector(T &v){int sz=v.size(); if(sz)
102
        cout << v[0]; for(int i = 1; i < sz; i++)cout << '' '<< v[i]; cout << '''
        \n;}/// prints all elements in a vector
   bool is Vowel (char ch) { ch=toupper (ch); if (ch='A' || ch='U' || ch='I'
103
        ||ch='O'||ch='E') return true; return false;}
   bool isConsonant(char ch){if (isalpha(ch) && !isVowel(ch)) return
104
        true; return false;}}
   /**Shortcut input function **/
105
   int read_int(){int n; scanf("%d",&n); return n;}
106
   int read_LLD(){LLD n; scanf("%lld",&n); return n;}
107
   inline int buffer_input() { char inp[1000]; scanstr(inp); return
108
        atoi(inp); }
   /**Direction**/
   ///int col[8] = {0, 1, 1, 1, 0, -1, -1, -1}; int row[8] = {1, 1, 0,
111
        -1, -1, -1, 0, 1; ///8 Direction
   ///int col[4] = \{1, 0, -1, 0\}; int row[4] = \{0, 1, 0, -1\}; ///4
        Direction
   ///int dx[] = \{2,1,-1,-2,-2,-1,1,2\}; int dy
        [] = \{1, 2, 2, 1, -1, -2, -2, -1\}; /// Knight Direction
   ///int dx[]=\{-1,-1,+0,+1,+1,+0\};int dy[]=\{-1,+1,+2,+1,-1,-2\}; ///
114
        Hexagonal Direction
                               ****** Ajaira Jinish Sesh
117
   int dp[88][88];
118
int N,M, step;
char S1[88], S2[88];
   int solve(int i, int j)
121
122
   {
        if (i=N and j=M) return 0;
123
        if (i=N) return M-j;
124
```

```
if (j=M) return N-i;
125
126
         int &ret=dp[i][j];
         if (ret!=-1)return ret;
127
         ret = (1 < <28);
128
         if(S1[i]==S2[j])ret=solve(i+1,j+1);
129
         else
130
131
         {
              ret=min(ret, solve(i, j+1)+1);
133
              ret=min(ret, solve(i+1,j)+1);
              ret=min(ret, solve(i+1, j+1)+1);
135
136
         return ret;
   }
137
    void pathPrint(int i, int j, int del, int ins, int st)
138
139
         if(i=N&&j=M) return ;
140
141
         if (i=N)
142
143
               for (int k=j; k<M; k++, i++)
              {
144
145
                    printf("%d Insert %d,%c\n",st++,i-del+1+ins,S2[k]);
              }
146
              return ;
147
148
         if ( j =M)
149
150
              for (; i < N; i++)</pre>
151
152
              {
                    printf("%d Delete %d\n", st++,i-del+1+ins);
153
                   del++;
155
              return ;
156
157
         int ret = solve(i,j);
158
         int tmp;
159
160
         if (S1[i]==S2[j])
161
162
              tmp=solve(i+1,j+1);
              if (ret=tmp)
163
164
                    pathPrint(i+1,j+1,del,ins,st);
165
                    return ;
167
168
         tmp=solve(i,j+1)+1;
169
         if (tmp=ret)
171
               printf("%d Insert %d,%c\n",st,i-del+1+ins,S2[j]);
172
              \mathtt{pathPrint}\left(\mathtt{i}\ ,\mathtt{j+1},\mathtt{del}\ ,\mathtt{ins+1},\mathtt{st+1}\right);
173
174
              return ;
         tmp=solve(i+1,j)+1;
177
         if (tmp==ret)
178
               printf("%d Delete %d\n", st, i-del+1+ins);
179
              \mathtt{pathPrint} \left( \, i + 1, j \,\, , \, d \, e \, l + 1, i \, n \, s \,\, , \, s \, t + 1 \right);
180
181
              return ;
```

```
182
183
         tmp = solve(i+1,j+1)+1;
         if (tmp==ret)
184
185
               printf("\%d \ Replace \ \%d,\%c\n",st\,i-del+1+ins\,S2[\,j\,]);
186
               pathPrint\left(\,i+1,j+1,del\,\,,ins\,\,,st+1\right);
187
188
               return ;
         }
189
         return ;
190
191 }
192
    int main()
193
194
         \#ifdef _ANICK_
         //f_input;
195
         #endif // ANICK_bool New=false;
196
197
          while (gets (S1))
198
199
200
               gets(S2);
               if(New)printf("\n");
201
202
              New=true;
              N=strlen(S1);
203
204
              M=strlen(S2);
205
              mem(dp, -1);
              step=solve(0,0);
printf("%d\n", step);
pathPrint(0,0,0,0,1);
206
207
208
209
         return 0;
210
211 }
```

# Strings

### 5.1 KMP

#### Tutorial

```
_{1} #include < bits / stdc++.h>
2 using namespace std;
3 char TXT[10000000], ptr[10000000];
4 vector<int> compute_prefix(const char *p)
6
       int = strlen(p+1);
       vector < int > prefix (m+1);
       prefix[1]=0;
       int k=0;
9
10
       for (int i=2; i \le m; i++)
            while (k>0 \text{ and } p[k+1]!=p[i]) k=prefix[k];
12
            if(p[k+1]==p[i])k=k+1;
13
14
            prefix[i]=k;
15
       return prefix;
16
17 }
vector < int > KMP_match(const char *txt, const char *ptrn)
19
       int n=strlen(txt+1);
20
       int m=strlen(ptrn+1);
21
       vector<int> Prefix=compute_prefix(ptrn);
22
       vector < int > Match_position;
23
       int q=0;
       for(int i=1; i <= n; i++)
25
26
            while (q>0 \text{ and } ptrn[q+1]!=txt[i]) q=Prefix[q];
27
            if(ptrn[q+1]==txt[i])q=q+1;
28
            if(q=m)
30
31
                Match_position.push_back(i-m);
                q=Prefix[q];
32
            }
33
```

```
return Match_position;
35
36 }
37 int main()
38 {
            scanf("%s %s", TXT+1, ptr+1);
39
            vector < int > Match_position = KMP_match(TXT, ptr);
40
            for (int i=0; i < Match_position.size(); i++)</pre>
41
42
                    \begin{array}{ll} \textbf{if} \ (!\,i\,) \ printf \ ("\%d"\,, Match\_position \ [\,i\,]) \ ; \\ \textbf{else} \ \ printf \ ("\%d"\,, Match\_position \ [\,i\,]) \ ; \\ \end{array} 
44
45
            return 0;
46
47 }
```

### 5.2 Aho Corasick

### 5.2.1 Aho Corasick with Dynamic Trie

```
1 #include < bits / stdc++.h>
2 using namespace std;
з #define Max 26
4 int getID (char c)
5 {
       return c>='a'?c-'a':c-'A';
7 }
s char inp[1000005];
9 char text[1000005];
int ans [5000];
map<string, int>Map;
vector < int > v;
13 struct Trie
14 {
       Trie *next[26], * fail;
15
16
       int stringMap;
       Trie()
17
18
       {
           stringMap = 0;
19
           for (int i=0; i<Max; i++)next[i]=NULL;
20
           fail=NULL;
21
22
23
24 Trie *root;
  void Insert(const char *str, int M)
25
26 {
       Trie *p=root;
27
       for (int i=0; str[i]; i++)
28
       {
29
           int id=getID(str[i]);
           if (p->next[id]==NULL)p->next[id]=new Trie();
31
           p=p->next[id];
32
33
       p->stringMap=M;
34
35 }
36 void computeFailure()
37 {
```

```
Trie *u, * prefix;
38
39
        queue<Trie*>Q;
       Q. push (root);
40
        while (!Q. empty())
41
42
            u=Q. front(); ///Take a new node
43
44
            Q. pop();
            for(int i=0; i<Max; i++)
45
46
                  if (u->next[i]!=NULL) ///select fail position of ith
47
        node of parent u
                      prefix=u->fail; /// Going to u node fail position/
49
        prefix position
                      while (prefix!=NULL)
51
                      {
                           if(prefix->next[i]!=NULL) ///if match found
                           {
53
54
                                u->next[i]->fail=prefix->next[i];
                                break;
55
56
                           prefix=prefix->fail; /// match not found, going
         to upper child prefix position
58
                      if (prefix=NULL)u->next[i]->fail=root;
59
                      Q.push(u->next[i]);
60
61
            }
62
63
64 }
65
   void AhoCorasick(const char *str)
66
        Trie *p=root;
67
        int cnt = 0;
68
        for (int i=0; str[i]; i++)
69
70
             int id=getID(str[i]);
71
72
             while (p->next[id]==NULL\&&p!=root)p=p->fail, cnt++;
             if (p->next[id]!=NULL)p=p->next[id];
73
74
            Trie *tp=p;
             while (tp!=root)
75
76
77
                  \begin{array}{l} \textbf{if} \ (\, tp -\!\!> \!\! string Map > \!\! 0) \, ans \, [\, tp -\!\!> \!\! string Map \,] + +; \end{array}
78
79
                 tp=tp->fail;
80
81
82 }
  void Delete (Trie *u)
83
84
        if (u=NULL) return;
85
        for (int i=0; i < Max; i++) Delete (u->next[i]);
86
87
        delete u;
88 }
89
90 int main()
91 {
```

```
int test;
92
93
         scanf("%d",&test);
         for (int t=1;t<=test;t++)
94
95
             Map.clear();
96
             v.clear();
97
             memset(ans,0,sizeof(ans));
98
             root=new Trie();
99
100
             int N;
             \operatorname{scanf}("%d",\&N);
             scanf("%s", text);
int cnt=1;
103
             for(int i=0; i<\!\!N; i++)
104
105
             {
                  scanf("%s", inp);
106
                   if (Map. find (inp) = Map. end ()) Map [inp] = cnt ++;
107
                  Insert(inp,Map[inp])
108
                  v.push_back(Map[inp]);
109
110
             }
             computeFailure();
112
             AhoCorasick (text);
             printf("Case %d:\n",t);
113
             for (int i=0; i< N; i++)
114
115
                  printf("\%d \ n", ans[v[i]]);
116
117
             Delete (root);
118
         }
119
         return 0;
120
121 }
```

#### 5.2.2 Aho Corasick with Static Trie

```
1 #include < bits / stdc++.h>
2 using namespace std;
3 #define root 0
4 #define NuLL −1
5 #define Max 248878
6 #define MC 26
7 int ans[10000];
8 char text[1000005];
9 char inp[100000];
10 map<string , int >Map;
vector < int > v;
int getID(const char c)
13 {
       return c>='a'?c-'a':c-'A';
14
15 }
  struct Trie
16
17
       struct node
18
19
           int Next[26], fail;
20
21
           int stringMap;
           void clear()
22
23
           {
               memset(Next, -1, sizeof(Next));
24
```

```
fail=-1;
25
26
                   stringMap = 0;
27
        T[Max];
28
        int ptr;
29
        void clear()
30
31
              ptr=1;
32
33
             T[0].clear();
34
        void Insert(char *str, int M)
35
36
              int p=0;
37
              for (int i=0; str[i]; i++)
38
39
                   int id=getID(str[i]);
40
41
                   if(T[p].Next[id]==-1)
42
                        T[p].Next[id]=ptr;
43
                        T[ptr++].clear();
44
45
                   int q=p;
46
                   p\!\!=\!\!T[\,p\,] . Next [\,i\,d\,\,]\,;
47
48
                   if(p<0)
49
                        while(1);
50
52
             T[p].stringMap=M;
54
55
        void ComputeFailure()
56
57
              queue<int>Q;
             Q. push (root);
58
59
              int u, prefix;
60
              int cnt=0, cnt2=0;
              while (!Q.empty())
61
62
                   u=Q. front();
63
                  \begin{array}{l} Q.\:pop\:(\:)\:;\\ \hline \text{for}\:(\:int\:\:\:i=0;i<\!\!M\!C;\:i++) \end{array}
64
65
66
                   {
                        if (T[u]. Next[i]!=NuLL)
67
                        {
68
                              int now=T[u].Next[i];
69
                              prefix=T[u].fail;
70
                              while (prefix!=NuLL)
71
72
                              {
                                   cnt2++;
73
                                   if (T[prefix]. Next[i]!=NuLL)
74
75
                                        T[\,now\,] . fail=T[prefix]. Next[i];
76
77
                                        break;
78
                                   prefix=T[prefix].fail;
79
80
81
                              if ( prefix=NuLL)T[now ] . fail=root;
```

```
Q. push (now);
 82
 83
                     }
84
               }
 85
 86
87 };
    void AhoCorasick(const Trie &A, const char *str)
 88
89
90
          int p=root;
          int cnt1=0, cnt2=0;
91
          for (int i=0; str[i]; i++)
92
 93
               int id=getID(str[i]);
94
               while (A.T[p].Next[id]==NuLL\&p!=root)p=A.T[p].fail;
 95
               \begin{array}{l} \textbf{if} \; (\, p! = \text{NuLL\&\&A} \, . \, T \, [\, p \, ] \, . \; \text{Next} \, [\, \text{id} \, ] \, ! = \text{NuLL}) \, p = \!\! A \, . \, T \, [\, p \, ] \, . \; \text{Next} \, [\, \text{id} \, ] \, ; \end{array}
 96
               int tp=p;
97
               while (tp!=root)
98
99
100
                     if (A.T[tp].stringMap>0)ans[A.T[tp].stringMap]++;
                     tp=A.T[tp].fail;
102
104
    Trie A;
105
    int main()
106
107
         #ifdef _ANICK_
108
               freopen ("input.txt", "r", stdin);
109
         #endif // _ANICK_
          int test;
112
          scanf("%d",&test);
          for (int t=1;t<=test;t++)
113
114
115
               Map.clear();
116
               v. clear ();
117
               memset(ans, 0, sizeof(ans));
               A. clear();
118
119
               int N;
               \operatorname{scanf}("%d",&N);
120
121
               \operatorname{scanf}("\%s", \operatorname{text});
               int cnt=1;
122
               for(int i=0; i<\!\!N; i++)
123
124
                     scanf("%s", inp);
                     if(Map. find(inp)=Map. end())Map[inp]=cnt++;
126
                     A. Insert (inp, Map[inp]);
127
                     v.push_back(Map[inp]);
128
129
               A. ComputeFailure();
130
131
               AhoCorasick (A, text);
               printf("Case %d:\n",t);
               for (int i=0; i< N; i++)
134
                     printf("\%d \ n", ans[v[i]]);
136
137
         return 0;
138
```

139 }

### 5.3 Manacher's Algorithm

```
1 #include < bits / stdc++.h>
using namespace std;
3 string s, t;
4 char str[1000005];
5 void prepare_string()
       int i;
t = "^#";
7
8
       for(i = 0; i < s.size(); i++)
9
          t += s[i], t += "#";
10
       t += "$";
11
12 }
13
14 int manacher()
15 {
       prepare_string();
16
17
       int P[t.size()], c = 0, r = 0, i, i-mirror, n = t.size() - 1;
18
19
20
       for (i = 1; i < n; i++)
21
           i_mirror = (2 * c) - i;
22
23
           P[i] = r > i? min(r - i, P[i_mirror]) : 0;
24
            while (t[i + 1 + P[i]] = t[i - 1 - P[i]])
26
                P[i]++;
27
28
            if(i + P[i] > r)
29
30
           {
                c = i;
31
                r = i + P[i];
32
33
       }
34
       return *max_element(P + 1, P + n);
35
36 }
37
38 int main()
39 {
       int kase = 1;
while(scanf(" %s", str) && str[0] != 'E')
40
41
42
43
            printf("Case %d: %d\n", kase++, manacher());
44
45
       return 0;
46
47 }
```

Computational geometry

## Math

### 7.1 Reduce Ratio

```
\left(\frac{A}{B}\right) ratio reduce to \left(\frac{x}{y}\right)
```

```
int main()
{
    int A,B,x,y;
    cin>>A>>B>>x>>y;
    int g=_-gcd(x,y);
    x/=g,y/=g;
    int t=min(A/x,B/y);
    cout<<x*t<<" "<<y*t<<endl;
    return 0;
}</pre>
```

# Number Theory

### 8.1 NCR

#### 8.1.1 Lucas Theorem

```
Fine NCR % M when N C M are large number.
2
       using Lucas theorem.
4 **/
5 #include < bits / stdc++.h>
6 using namespace std;
7 typedef long long LLD;
8 LLD mod=1000003;
9 LLD big_mod(LLD n, LLD p, LLD m)
10 {
       if (p==0)return (LLD) 1;
11
       LLD x=big_mod(n,p/2,m);
12
       x=(x*x)%m;
13
       if(p&1)x=(x*n)\%m;
14
       return x;
16 }
17 LLD inverse_modulo(LLD t,LLD m)
18 {
       return big_mod(t,m-2,m);
19
21 LLD combi(LLD n, LLD k, LLD m)
22 {
       i f (n<k)
23
           return 0;
24
       if (n-k<k)
25
           return combi(n,n-k,m);
26
       LLD i, p=1, t=1;
       for (i=n-k+1; i \le n; i++)
28
          p=(p*i)\%m;
       for (i=1; i \le k; i++)
30
           t = (t * i) \% m;
31
       return (p*inverse_modulo(t,m))%m;
32
33 }
```

```
^{34} LLD lucas(LLD n, LLD k, LLD m) ^{35} {
        if(n < k)
36
37
            return 0;
        if (k==0 || n==k)
return 1;
38
39
        return (lucas (n/m, k/m,m)*combi(n/m, k/m,m))/m;
40
41 }
42 int main()
43 {
        return 0;
44
44
45 }
```