ALGORITHM II
LAB REPORT

Title: Algorithm II Lab
Course code: CSE-258
2nd Year 2nd Semester
Submitted to

Mohammad Ashraful Islam **Assistant Professor** **Department of Computer Science and Engineering**	**Bulbul Ahammad** **Assistant Professor** **Department of Computer Science and Engineering**

NAME	EXAM ROLL	CLASS ROLL	REGISTRATION NUMBER
Md Tanvir Hossain Saon	**202200**	**388**	**20200650758**

Sparse Table

Source Code :

```
#include<bits/stdc++.h>
using namespace std;
#define N 1000
int arr[]={0,2,6,1,4,9,4,6,1,7,3};
int Table[20][N];
void build(int n)
{
    for(int i=1;i<=n;i++)Table[0][i]=arr[i];
    for(int i=1;i<=3;i++)
    {
        for(int j=1;j<=n;j++)
        {
            Table[i][j]=Table[i-1][j];
            if(j+(1<<(i-1)) <=n)Table[i][j] = min(Table[i][j], Table[i-1][j+ (1<<(i-1))]);
        }
    }
}
int query(int lft,int rgt)
{
    int range = rgt - lft + 1;
    int dg=31-__builtin_clz(range);
    return min (Table[dg][lft] , Table[dg][rgt - ( 1<<dg )+1  ]);
}
int main()
{
    build(10);
    cout<<"  ";
    for(int j=1;j<=10;j++)
        {
            cout<<setw(3)<<fixed<<j<<" ";
        }
        cout<<"\n=====================================\n";
    for(int i=0;i<=3;i++)
    {
        cout<<i<<": ";
        for(int j=1;j<=10;j++)
        {
            cout<<setw(3)<<fixed<<Table[i][j]<<" ";
        }
        cout<<endl;
    }
    cout<<"Min of 2 to 6 range is "<<query(2,6)<<endl;
}
```

Output:

```
"D:\202200_388_Algorithm II\SparseTable.exe"
    1   2   3   4   5   6   7   8   9  10
=====================================
0:   2   6   1   4   9   4   6   1   7   3
1:   2   1   1   4   4   4   1   1   3   3
2:   1   1   1   4   1   1   1   1   3   3
3:   1   1   1   1   1   1   1   1   3   3
Min of 2 to 6 range is 1

Process returned 0 (0x0)   execution time : 0.078 s
Press any key to continue.
```

Robin Karp

Source Code :

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

vector<int>RabinKarp(string const& s, string const& t) {
    const int p = 37;
    const int m = 1e9 + 7;
    int ls = (int)s.size(), lt = (int)t.size();
    int T =lt;
    vector<long long> p_pow(max(ls, lt));
    p_pow[0] = 1;
    for (int i = 1; i < (int)p_pow.size(); i++)
        p_pow[i] = (p_pow[i-1] * p) % m;

    vector<long long>h(lt + 1, 0);
    for (int i = 0; i < lt; i++)
        h[i+1] = (h[i] + (t[i] - 'a' + 1) * p_pow[i]) % m;
    long long h_s = 0;
    for (int i = 0; i < ls; i++)
        h_s = (h_s + (s[i] - 'a' + 1) * p_pow[i]) % m;

    vector<int>occurences;
    for (int i = 0; i + ls - 1 < T; i++) {
        long long cur_h = (h[i+ls] + m - h[i]) % m;
        if (cur_h == h_s * p_pow[i] % m)
            occurences.emplace_back(i);
    }
    return occurences;
}
int main()
{
    string Text="abaabaaabaaab";
    string Pattern ="aba";
    vector<int>found = RabinKarp(Pattern,Text);
    cout<<found.size()<<" Place found\n";
    cout<<"Index : ";
    for(int i=0;i<found.size();i++)cout<<found[i]<<" ";
    cout<<endl;
}
```

Output:

```
"D:\202200_388_Algorithm II\Hashing.exe"
3 Place found
Index are: 0 3 7

Process returned 0 (0x0)   execution time : 0.047 s
Press any key to continue.
```

MATRIX EXPONENTIATION

Source Code :

```
#include <bits/stdc++.h>
using namespace std;
void multiply(int F[2][2], int M[2][2])
{
   int a= F[0][0]%10001 * M[0][0]%10001 + F[0][1]%10001 * M[1][0]%10001;
   int b= F[0][0]%10001 * M[0][1]%10001 + F[0][1]%10001 * M[1][1]%10001;
   int c= F[1][0]%10001 * M[0][0]%10001 + F[1][1]%10001 * M[1][0]%10001;
   int d= F[1][0]%10001 * M[0][1]%10001 + F[1][1]%10001 * M[1][1]%10001;
   F[0][0] = a;
   F[0][1] = b;
   F[1][0] = c;
   F[1][1] = d;
}
void power(int F[2][2], int n)
{
   if (n == 1 || n == 2)
      return ;
   int M[2][2] = {{1,1},{1,0}};
   for(int i=2;i<n;i++)
   {
       multiply(F, M);
   }
}
int fib_mat(int n)
   {
   int F[2][2] = {{1,1},{1,0}};
   if (n == 1)
      return 0;
   power(F, n - 1);
   return F[0][0];
   }
int main()
{
   int n;
   while (1)
        {
          cout<<"Enter the integer n to find nth fibonacci(Enter 0 to exit):";
          cin>>n;
          if (n == 0)
          break;
          cout<<fib_mat(n)<<endl;
       }
   return 0;
}
```

Output:

```
"D:\202200_388_Algorithm II\matrixexpo.exe"
Enter the integer n to find nth fibonacci(Enter 0 to exit):12
89
Enter the integer n to find nth fibonacci(Enter 0 to exit):345
13937
Enter the integer n to find nth fibonacci(Enter 0 to exit):0

Process returned 0 (0x0)   execution time : 13.903 s
Press any key to continue.
```

Segment Tree

Source Code :

```
#include<bits/stdc++.h>
using namespace std;
int arr[]={4, 9, 2, 4, 2, 9, 3, 7, 8, 0, 5, 3, 9};

struct node
{
    int st,ed;
    int mn;
    node *l,*r;
    node(){}
    node(int _x, int _y){st=_x,ed=_y;l=r=NULL; mn = 10000000; }
};
void build(node *ob )
{

    if(ob->st==ob->ed)
    {
        int ind = ob->st;
        ob->mn = arr[ind];
        return ;
    }
    int mid = (ob->st + ob->ed)/2;
    if(ob->l==NULL)ob->l = new node(ob->st, mid);
    if(ob->r==NULL)ob->r = new node (mid+1, ob->ed);
        build(ob->l);
        build(ob->r);
    ob->mn = min( ob->l->mn, ob->r->mn);
}

void insert(node *ob ,  int ind , long long  val)
{

    if(ob->st==ob->ed)
    {
        ob->mn = val;
        return ;
    }
    int mid = (ob->st + ob->ed)/2;
    if(ob->l==NULL)ob->l = new node(ob->st, mid);
    if(ob->r==NULL)ob->r = new node (mid+1, ob->ed);

    if(ind<=mid)
    {
        insert(ob->l, ind, val);
    }
    else
    {
```

```
        insert(ob->r,ind, val);
    }

    ob->mn = min( ob->l->mn, ob->r->mn);
}
long long query(node *ob , int x , int y)
{
    if(ob->st==x && ob->ed==y)
    {
        return ob->mn;
    }
    int mid = (ob->st + ob->ed)/2;
    if(y<=mid)
    {
        return query(ob->l,x,y);
    }
    else if(x>mid)
    {
        return query(ob->r,x,y);
    }
    {
        int a = query(ob->l, x, mid);
        int b = query(ob->r, mid+1, y);
        return min(a,b);
    }
}
int main()
{
    int n = 13;
    node *root =new node(0,n-1);
    cout << "Starting Node: ";
    cout<<root->st<<endl;
    cout << "Ending Node: ";
    cout<<root->ed<<endl;

    build(root);
    cout<<"query(1,1) provides "<<query(root, 1,1)<<endl;
    cout<<"query(1,10) provides "<<query(root, 1,10)<<endl;
    cout<<"query(2,8) provides "<<query(root,2,8)<<endl;
    cout<<"query(10,12) provides "<<query(root,10,12)<<endl;
    return 0;
}
```

Output:

```
"D:\202200__388__Algorithm II\Segment_Tree.exe"
Starting Node: 0
Ending Node: 12
query(1,1) provides 9
query(1,10) provides 0
query(2,8) provides 2
query(10,12) provides 3

Process returned 0 (0x0)   execution time : 0.328 s
Press any key to continue.
```

Square Root Decomposition

Source Code :

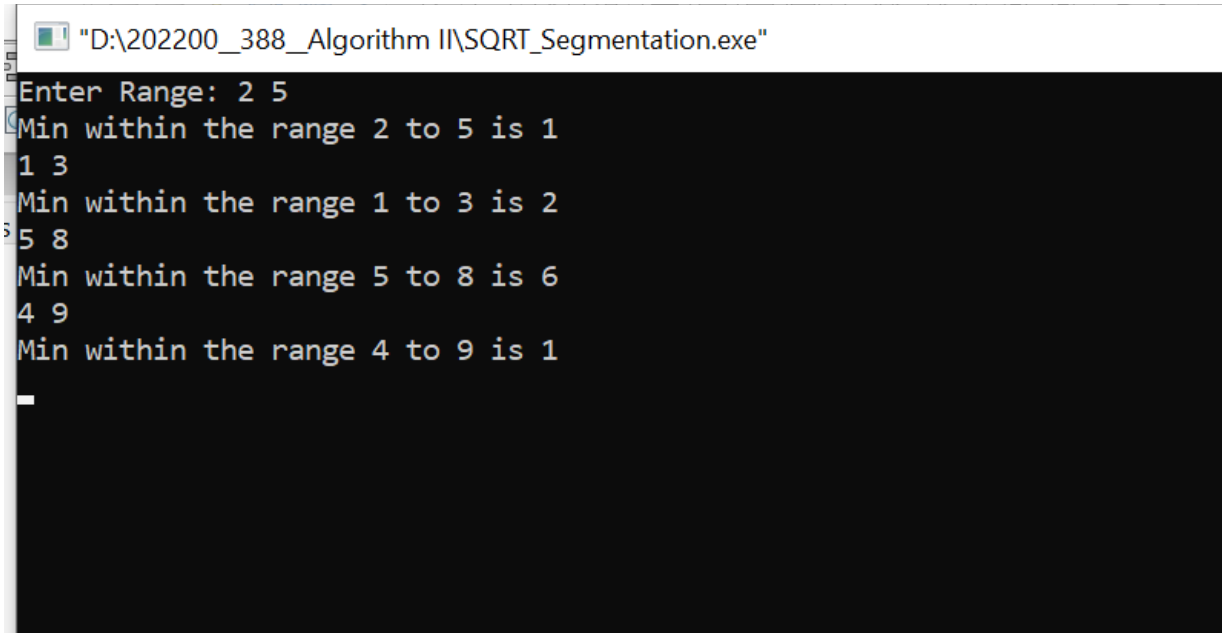
```
#include<bits/stdc++.h>
using namespace std;
int A[]={0,2,4,3,1,6,7,8,9,10,7};
int Rn,N=10;
int Block[10];
int Query(int L, int R)
{
    int Lb = (L-1)/Rn + 1;
    int Rb = (R-1)/Rn + 1;
    int res = 1000000;
    for(int i=Lb+1;i<=Rb-1;i++)res = min(res, Block[i]);
    for(int i=L;i<=min(R,Lb*Rn );i++)res = min(res,A[i]);
    for(int i= max( (Rb-1)*Rn +1, L ); i<=R;i++ ) res = min(res,A[i]);
    return res;
}
void preProcessing()
{
    int numberofBlock = 10/Rn;
    for(int i=1;i<=numberofBlock;i++)
    {
        Block[i]=1000000;
        for(int j= (i-1)*Rn + 1; j<=i*Rn;j++)
        {
            Block[i] = min( Block[i],A[j]);
        }
    }
}

int bruteForceSolution(int L, int R)
{
    int res = 100000;
    for(int i=L;i<=R;i++)res = min(res,A[i]);
    return res;
}

int main()
{
    Rn = sqrt(N);
    int numberofBlock = 10/Rn;
    int L,R;
    if(N%Rn)numberofBlock++;
    preProcessing();
    cout << "Enter Range: ";
    while(cin>>L>>R)
    {
    cout<<"Min within the range "<<L<<" to "<< R<<" is "<<Query(L,R)<<endl;
    assert(Query(L,R)==bruteForceSolution(L,R));
    }
```

```
    return 0;
}
```

Output:



KMP

Source Code :

```
#include<iostream>
#include<vector>
#include<map>
#include<algorithm>
#include<cstdio>
#include<cmath>
#include<cstdlib>
#include<cstring>
#include<queue>
#include<fstream>
#include<sstream>
#include<stack>
#include<list>
#include<deque>
#include<bitset>
#include<utility>
#include<climits>
#include<iomanip>
#include<ctime>
#include<complex>
using namespace std;


#define FOR(i,a,b) for (int i=(a);i<(b);i++)
#define RFOR(i,a,b) for (int i=(b)-1;i>=(a);i--)
#define REP(i,n) for (int i=0;i<(n);i++)
#define RREP(i,n) for (int i=(n)-1;i>=0;i--)

#define inf INT_MAX/3
#define pb push_back
#define MP make_pair
#define all(a) (a).begin(),(a).end()
#define SET(a,c) memset(a,c,sizeof a)
#define CLR(a) memset(a,0,sizeof a)
#define pii pair<int,int>
#define pcc pair<char,char>
#define pic pair<int,char>
#define pci pair<char,int>
#define VS vector<string>
#define VI vector<int>
#define debug(x) cout<<#x<<": "<<x<<endl
#define MIN(a,b) (a>b?b:a)
#define MAX(a,b) (a>b?a:b)
#define pi 2*acos(0.0)
#define INFILE() freopen("in0.txt","r",stdin)
#define OUTFILE()freopen("out0.txt","w",stdout)
#define in scanf
#define out printf
#define ll long long
```

```
#define ull unsigned long long
#define eps 1e-9
#define mod 10007

template<typename T>inline T  S(T a){return a*a;}
template<typename T>inline string tostring(T a){ostringstream os("");os << a;return os.str();}
template<typename T>inline ll tolong(T a){ll res;istringstream os(a);os>>res;return res;}
template<typename T>inline T gcd(T a, T b){if (b == 0)return a;else return gcd(b, a % b);}
template<typename T>inline ull bigmod(T a, T b, T m){if (b == 0)return 1;else if (b % 2 ==
0)return S(bigmod(a, b / 2, m)) % m;else return (a % m*bigmod(a, b - 1, m)) % m;}
template<typename T>inline VS parse(T str){VS res;string s;istringstream
os(str);while(os>>s)res.pb(s);return res;}
template<typename T>inline ull  dist(T A,T B){ull
res=(A.x-B.x)*(A.x-B.x)+(A.y-B.y)*(A.y-B.y);return res;}

char T[1000009];
char P[1000009];
int tlen,plen;
int match[1000009];
void build()
{
    match[0]=-1;
    match[1]=0;
    int i,q;
      q=0;
   for(i=2;i<=plen;)
   {

        if(P[i-1]==P[q])
        {
            q++;
            match[i]=q;
            i++;
        }
        else if(q>0)q=match[q];
        else
        {
            match[i]=0;
            i++;
        }
   }
}

void kmp()
{
    int cnt=0;
    for(int i=0,m=0; i+m<tlen;)
    {
        if(P[i]==T[i+m])
        {
            i++;
            if(i%plen==0)cnt++;
        }
```

```
        else
        {
            m=m+i-match[i];
            if(m<0)m=0;

            if(match[i]>0)i=match[i];
            else i=0;
        }
        //cout<<i<<" "<<plen<<endl;
    }
    cout<<cnt<<endl;
}


int main()
{
    int ks,cas;
    cout<<"How Many Test Case: ";
    cin>>ks;
    for(cas=1;cas<=ks;cas++)
    {
        cout << "\nEnter the Text: ";
        scanf("%s",T);
        cout << "\nEnter the Pattern: ";
        scanf("%s",P);
        tlen=strlen(T);
        plen=strlen(P);
        cout<<"Case "<<cas<<": ";
        build();
        kmp();
    }
   return 0;
}
```

Output:

```
"D:\202200_388_Algorithm II\KMP.exe"
How Many Test Case: 2

Enter the Text: abbaabaabcc

Enter the Pattern: aba
Case 1: 1

Enter the Text: dfsdfsccfbfs

Enter the Pattern: scc
Case 2: 1

Process returned 0 (0x0)   execution time : 35.935 s
Press any key to continue.
```

Binary Index Tree

Source Code :

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

vector<int> bit(1000);

void update(int i, int val) {
    while (i < bit.size()) {
        bit[i] += val;
        i += i & -i;
    }
}

int query(int i) {
    int res = 0;
    while (i > 0) {
        res += bit[i];
        i -= i & -i;
    }
    return res;
}

int main() {
    int n,q;
    cout<<"Enter number of elements & query : "<<endl;
    cin>>n>>q;
    for(int i=0;i<n;i++){
        int x;
        cin>>x;
        update(i+1,x);
    }
   while(q--){
        int p,x,y;
        cout<<endl;
        cout<<endl;

        cout<<"Press 1 For update\nPress 2 for Query"<<endl;
        cin>>p>>x>>y;
        if(p==1){
            update(x,y);
        }
        else{
            int ans = query(y)-query(x-1);
            cout<<ans<<endl;
        }
   }
    return 0;
}
```

Output:

```
"D:\202200_388_Algorithm II\BIT.exe"
Enter number of elements & query :
5 5 2 5 3 4 7


Press 1 For update
Press 2 for Query
2 1 5
21


Press 1 For update
Press 2 for Query
1
4 5


Press 1 For update
Press 2 for Query
2
1 5
26


Press 1 For update
Press 2 for Query
```