Team Notebook

Rathin Bhargava

December 14, 2018

Contents

1 AVLTree

```
#include<iostream>
#include<algorithm>
#include<deque>
using namespace std;

//
// For any question, if you figure out the stuff you need to
      store in the nodes, you have to change/update the
// code in the following places
// 1. struct node
// 2. generate
// 3. Insert
// 4. Delete - children 1,2, outside
// 5. zigzags and zigzigs
// 6. Make a function like height() or number() if you want
     to
// 7. In the inorderSuccesor
// 8. Make different functions for what you want

int count1 = 0;
struct node
{
  node* left;
  node* right;
  int height;
  int data;
};

node* generate(int key,int height1)
{
  node* n = new node;
  n->left = NULL;
  n->right = NULL;
  // if(p!=NULL)n->parent = p;
  // else n->parent = NULL;
  n->height = height1;
  n->data = key;
  return n;
}

int height(node* root)
{

  if(root==NULL) {return -1;}
  else {return root->height;}
}

void level(node* root)
{
  if(root != NULL)
  {
    deque< node* > q;
    q.push_back(root);
    q.push_back(NULL);
    while(!q.empty())
    {
      node* n = q.front();
      if(n!=NULL)
      {
        if(n->left)q.push_back(n->left);
        if(n->right)q.push_back(n->right);
        cout<<n->data<<" ";
        q.pop_front();
      }
      else
      {
        cout<<endl;
        q.pop_front();
        if(q.front()!=NULL) q.push_back(NULL);
      }

    }
  }
}

// void level(node* root)
// {
//   if(root != NULL)
//   {
//     deque< node* > q;
//     q.push_back(root);
//     // q.push_back(NULL);
//     while(!q.empty())
//     {
//       node* n = q.front();
//       if(n!=NULL)
//       {
//         if(n->left)q.push_back(n->left);
//         if(n->right)q.push_back(n->right);
//         cout<<n->data<<" "<<n->height<<endl;
//         q.pop_front();
//       }
//       else
//       {
//         cout<<endl;
//         q.pop_front();
//         if(q.front()!=NULL) q.push_back(NULL);
//       }
//
//     }
//   }
// }

void inorder(node* root)
{
  if(root!=NULL)
  {
    inorder(root->left);
    cout<<root->data<<" ";
    inorder(root->right);
  }
}

void preorder(node* root)
{
  if(root!=NULL)
  {
    cout<<root->data<<" ";
    preorder(root->left);
    preorder(root->right);
  }
}

int computeHeight(node* root)
{
  if(root!=NULL)
  {
    root->height = max(computeHeight(root->left),
         computeHeight(root->right))+1;
    return root->height;;
  }
  else return -1;
}

node* zigzig(node* z,node* y,node* x,node* t1,node* t2,node*
     t3,node* t4)
{
  // cout<<"data: "<<z->data<<" height "<<z->height<<" data:
       "<<y->data<<" height "<<y->height<<" data: "<<x->data
      <<" height "<<x->height<<endl;
  z->left = t1;z->right = t2;
  y->left = z; y->right = x;
  x->left = t3; x->right = t4;
  z->height-=2;
  // cout<<"data: "<<z->data<<" height "<<z->height<<" data:
       "<<y->data<<" height "<<y->height<<" data: "<<x->data
      <<" height "<<x->height<<endl;
  return y;
```

```
}

node* zigzag(node* z,node* y,node* x,node* t1,node* t2,node*
      t3,node* t4)
{
  cout<<"data: "<<z->data<<" height "<<z->height<<" data: "
       <<y->data<<" height "<<y->height<<" data: "<<x->data<<
       " height "<<x->height<<endl;
  x->left = z; x-> right = y;
  z->left = t1; z->right = t2;
  y->left = t3; y->right = t4;
  x->height+=1;
  y->height-=1;
  z->height-=2;
  cout<<"data: "<<z->data<<" height "<<z->height<<" data: "
       <<y->data<<" height "<<y->height<<" data: "<<x->data<<
       " height "<<x->height<<endl;
  return x;
}

node* zagzag(node* z,node* y,node* x,node* t1,node* t2,node*
      t3,node* t4)
{
  // cout<<"data: "<<z->data<<" height "<<z->height<<" data:
       "<<y->data<<" height "<<y->height<<" data: "<<x->data
       <<" height "<<x->height<<endl;
  x->left = t1;x->right = t2;
  y->left = x; y->right = z;
  z->left = t3; z->right = t4;
  z->height-=2;
  // cout<<"data: "<<z->data<<" height "<<z->height<<" data:
       "<<y->data<<" height "<<y->height<<" data: "<<x->data
       <<" height "<<x->height<<endl;
  return y;
}

node* zagzig(node* z,node* y,node* x,node* t1,node* t2,node*
      t3,node* t4)
{
  cout<<"data: "<<z->data<<" height "<<z->height<<" data: "
       <<y->data<<" height "<<y->height<<" data: "<<x->data<<
       " height "<<x->height<<endl;
  x->left = y; x-> right = z;
  y->left = t1; y->right = t2;
  z->left = t3; z->right = t4;
  y->height-=1;
  z->height-=2;
  x->height+=1;
  cout<<"data: "<<z->data<<" height "<<z->height<<" data: "
       <<y->data<<" height "<<y->height<<" data: "<<x->data<<
       " height "<<x->height<<endl;
  return x;
}

node* decide(node* z)
{
  count1++;
  node* y = new node;
  node* x = new node;
  // cout<<"z data left: "<<z->left->data<<" height: "<<z->
       left->height<<" z data right: "<<z->right->data<<"
       height: "<<z->right->height<<endl;
  if(height(z->right) > height(z->left))
  {
    y = z->right;
    if(height(y->right) > height(y->left)) {x = y->right; z =
         zigzig(z,y,x,z->left,y->left,x->left,x->right);} //
        The parameters are z,y,x,t1,t2,t3,t4
    else {x = y->left; z = zigzag(z,y,x,z->left,x->left,x->
        right,y->right);}
  }
  else
  {
    y = z->left;
    if(height(y->right) > height(y->left)) {x = y->right; z =
         zagzig(z,y,x,y->left,x->left,x->right,z->right);}
    else {x = y->left; z = zagzag(z,y,x,x->left,x->right,y->
        right,z->right);}
  }
  // cout<<"z data left: "<<z->left->data<<" height: "<<z->
       left->height<<" z data right: "<<z->right->data<<"
       height: "<<z->right->height<<endl;
  return z;
}

node* insert(node* root,int key)
{
  if(root!=NULL)
  {
    if(key > root->data) root->right = insert(root->right,key
        );
    else root->left = insert(root->left,key);
    root->height = max(height(root->right),height(root->left)
        )+1;
    // cout<<"data: "<<root->data<<" left: "<<height(root->
        left)<<" right: "<<height(root->right)<<endl;
    if(abs(height(root->left) - height(root->right)) > 1)
    {
      // cout<<"data: "<<root->data<<" left: "<<height(root->
          left)<<" right: "<<height(root->right)<<endl;
      root = decide(root);
    }
    return root;
  }
  else
  {
    return generate(key,0);
  }
}

node* inorderSuccesor(node* root)
{
  while(root->left!=NULL)
  {
    root->height = max(height(root->left) -1,height(root->
        right)) +1;
    root = root->left;
  }
  return root;
}

node* delete1(node* root,int key)
{
  // cout<<"data: "<<root->data<<endl;
  if(key< root->data) {root->left = delete1(root->left,key);
      if(root->left) cout<<"root left: "<<root->left->data<<
      " "<<root->data<<endl;}
  else if(key>root->data) {root->right = delete1(root->right
      ,key);if(root->right) cout<<"root right: "<<root->
      right->data<<" "<<root->data<<endl;}
  else
  {
    int children = 2;
    if(root->left == NULL) children-=1;
    if(root->right== NULL) children-=1;
    if(children == 0) {root = NULL;return NULL;}
    else if(children == 1)
    {
      node* child;
      if(root->left!=NULL) child = root->left;
      else child = root->right;
      child->height = max(height(child->left),height(child->
          right)) +1;
      return child;
    }
    else //Children ==2
    {
      node* child = inorderSuccesor(root->right);
      root->data = child->data;
      root->right = delete1(root->right,child->data);
```

```
    // cout<<"root data: "<<root->right->data<<endl;
    root->height = max(height(root->left),height(root->
        right)) +1;
    if(abs(height(root->left) - height(root->right)) > 1)
        root = decide(root);
    return root;
   }
  }
  if(abs(height(root->left) - height(root->right)) > 1)
  {
   // cout<<"data: "<<root->data<<" left: "<<height(root->
       left)<<" right: "<<height(root->right)<<endl;
   root = decide(root);
   // cout<<"data: "<<root->data<<" left: "<<height(root->
       left)<<" right: "<<height(root->right)<<endl;
  }
  root->height = max(height(root->left),height(root->right))
      +1;
  // cout<<"root height: "<<root->height<<endl;
  return root;
}

int main()
{
  //typedef struct node node;
  int n;
  cin>>n;
  int l[n];
  for(int i=0;i<n;i++) cin>>l[i];
  node* root = NULL;
  node *p = root;
  root = insert(root,l[0]);
  for(int i=1;i<n;i++) root = insert(root,l[i]);
  level(root);
  cout<<"Inorder"<<endl;
  inorder(root);
  cout<<endl;
  cout<<"Preorder"<<endl;
  preorder(root);
  cout<<endl;
  int q;
  cin>>q;
  for(int h=0;h<q;h++)
  {
    int a;
    cin>>a;
    if(a==1)//Insert value
    {
      int key;
      cin>>key;
      root = insert(root,key);
      level(root);
      cout<<"Inorder"<<endl;
      inorder(root);
      cout<<endl;
      cout<<"Preorder"<<endl;
      preorder(root);
      cout<<endl;
    }
    else if(a==2) //Delete value
    {
      int key;
      cin>>key;
      root = delete1(root,key);
      cout<<"Level order"<<endl;
      level(root);
      cout<<"Inorder"<<endl;
      inorder(root);
      cout<<endl;
      cout<<"Preorder"<<endl;
      preorder(root);
      cout<<endl;
      cout<<"Count: "<<count1<<endl;
    }
  }
}
/*
8
4 7 10 11 6 2 3 1
8
1 0 1 -1 1 -2 2 2
*/
/*
8
10 7 12 3 9 11 17 8
2 2 10 2 11

*/
```

2 BIT

```
#include<bits/stdc++.h>
#define mt make_tuple
#define mp make_pair
#define pu push_back
#define INF 1000000001
#define MOD 1000000007
#define ll long long int
#define ld long double
#define vi vector<int>
#define vll vector<long long int>
#define sc(n) scanf("%d",&n);
#define scll(n) scanf("%lld",&n);
#define scld(n) scanf("%Lf",&n);
#define scr(s) {char temp[1000000];scanf("%s",temp);s = temp
    ;}

using namespace std;

typedef struct BIT //int
{
  vector<int> bit;
  int n;

  void init(int n)
  {
    this->n = n;
    bit.assign(n,0);
  }

  int sum(int i)
  {
    int res = 0;
    for(;i>=0;i = (i&(i+1)) -1)
    {
      res+=bit[i];
    }
    return res;
  }

  void inc(int i, int delta)
  {
    for(;i<n;i = i|(i+1))
    {
      bit[i]+=delta;
    }
  }

  int getsum(int l,int r)
  {
    // If l==0, sum(-1) automatically returns the default
        calue of res, 0
    return sum(r)-sum(l-1);
  }

  void init(vector<int> v)
  {
    init(v.size());
```

```
    for(int i=0;i<v.size();i++) inc(i,v[i]);
  }
} BIT;

typedef struct BIT //ll
{
  vector<ll> bit;
  int n;

  void init(int n)
  {
    this->n = n;
    bit.assign(n,0);
  }

  ll sum(int i)
  {
    ll res = 0;
    for(;i>=0;i = (i&(i+1)) -1)
    {
      res+=bit[i];
    }
    return res;
  }

  void inc(int i, ll delta)
  {
    for(;i<n;i = i|(i+1))
    {
      bit[i]+=delta;
    }
  }

  ll getsum(int l,int r)
  {
    // If l==0, sum(-1) automatically returns the default
         calue of res, 0
    return sum(r)-sum(l-1);
  }

  void init(vector<ll> v)
  {
    init(v.size());
    for(int i=0;i<v.size();i++) inc(i,v[i]);
  }
} BIT;

int main()
{
  typedef struct BIT BIT;
  int n;
  sc(n);
  vector<int> v;
  for(int i=0;i<n;i++) {int a;sc(a);v.pu(a);}
  BIT bit;
  bit.init(v);
  int q;
  sc(q);
  for(int h=0;h<q;h++)
  {
    cout<<bit.getsum(0,4)<<endl;
    bit.inc(0,5);
    cout<<bit.getsum(0,4)<<endl;
  }
  return 0;
}
```

3 BST

```
#include<iostream>
#include<deque>
using namespace std;

struct node
{
  node* left;
  node* right;
  node* parent;
  int data;
};

node* generate(int key,node*p)
{
  node* n = new node;
  n->left = NULL;
  n->right = NULL;
  if(p!=NULL)n->parent = p;
  else n->parent = NULL;
  n->data = key;
  return n;
}

void level(node* root)
{
  if(root != NULL){
  deque< node* > q;
  q.push_back(root);
  while(!q.empty())
  {
    node* n = q.front();
    if(n->left)q.push_back(n->left);
    if(n->right)q.push_back(n->right);
    if(n->parent!=NULL)cout<<n->data<<" "<<n->parent->data<<
         endl;
    else cout<<n->data<<" NULL"<<endl;
    q.pop_front();
  }
  cout<<endl;}
}

node* insert(node* root,node* parent,int n)
{
  if(root != NULL)
  {
    if(n > root->data)
    {
      root->right = insert(root->right,root,n);
    }
    else
    {
      root->left = insert(root->left,root,n);
    }
    return root;
  }
  else
  {
    return generate(n,parent);
  }
}

node* search(node*root,int key)
{
  if(root!=NULL)
  {
    if(root->data == key) return root;
    node *a = new node;
    node*b = new node;
    a = search(root->left,key);
    b = search(root->right,key);
    if(a!=NULL)return a;
    else return b;
  }
  else return NULL;
}

int getIdentity(node*root)
{
  if(root->parent!=NULL)
```

```
  {
    if(root->parent->left==root) return 0;
    else return 1;
  }
  return -1;
}

node* delete1(node* root,node* ultraroot)
{
  int boo = 0;
  if(root==ultraroot) boo = 1;
  int children = 2;
  if(root->left == NULL) children-=1;
  if(root->right== NULL) children-=1;
  int identity = getIdentity(root);
  if(children==0)
  {
    if(identity==-1) {root = NULL;ultraroot = NULL;}
    else if(identity==0) {root->parent->left = NULL; root =
        NULL;}
    else if(identity==1) {root->parent->right = NULL; root =
        NULL;}
    if(boo)return NULL;
    else return ultraroot;
  }
  else if(children==1)
  {
    node* child;
    if(root->left!=NULL) child = root->left;
    else child = root->right;
    if(identity==-1) {child->parent = NULL; root = NULL;}
    else if(identity == 0) {root->parent->left = child; child
        ->parent = root->parent; root = NULL;}
    else {root->parent->right = child; child->parent = root->
        parent; root = NULL;}
    if(boo) return child;
    else return ultraroot;
  }
  else
  {
    node* child = root->right;
    while(child->left!=NULL) child = child->left;
    delete1(child,ultraroot);
    //Taking care of children
    root->left->parent = child;
    if(root->right!=NULL)root->right->parent = child;
    //Taking care of self and child(node replacing self)
    child->left = root->left;
    child->right = root->right;
    child->parent = root->parent;
    if(identity==-1) {root = NULL;}
    else if(identity==0){root->parent->left = child;root =
        NULL;}
    else if(identity==1){root->parent->right = child;root =
        NULL;}
    if(boo) return child;
    else return ultraroot;
  }
}
int main()
{
  typedef struct node node;
  int n;
  cin>>n;
  int l[n];
  for(int i=0;i<n;i++) cin>>l[i];
  node* root = NULL;
  node *p = root;
  root = insert(root,NULL,l[0]);
  // cout<<root->data<<endl;
  for(int i=1;i<n;i++)insert(root,NULL,l[i]);
  // level(root);
  int q;
  cin>>q;
  for(int h=0;h<q;h++)
  {
    int a;
    cin>>a;
    if(a==1)//Insert value
    {
      int b;
      cin>>b;
      insert(root,NULL,b);
      // level(root);
    }
    else if(a==2)
    {
      int key;
      cin>>key;
      node* p = search(root,key);
      node*k = delete1(p,root);
      if(k!=NULL)root = k;
      level(root);
    }
  //   else if(a==3)
  //   {
  //     int b;
  //     cin>b;
  //     search(b);
  //   }
  }
}
```

4 BinaryHeap

```
#include<iostream>
#include<vector>
#include<map>
using namespace std;

map <long long int,long long int> m;
map <long long int,long long int> ::iterator it;
int count1 = 0;

void bottomupheapify(vector <long long int> &l,long long int
     n,long long int index)
{
  m[l[index]] = index;
  if(index==0)count1 ++;
  while(index>0 && l[(index-1)/2]>l[index])
  {
    m[l[index]] = (index-1)/2;
    m[l[(index-1)/2]] = index;
    long long int k = l[index];
    l[index] = l[(index-1)/2];
    l[(index-1)/2] = k;
    index = (index-1)/2;
  }
}

void topdownheapify(vector <long long int> &l,long long int
    n,long long int index)
{
  while(2*index+1<n)
  {
    long long int left = 2*index+1,right = 2*index+2,c;
    if(right<n)
    {
      cout<<c<<endl;
      if(l[left]<l[index] && l[left]<l[right]) c = left;
      else if(l[right]<l[index]) c = right;
      else break;
      cout<<c<<endl;
    }
    else if(right>=n)
    {
      if(l[left]<l[index]) c = left;
      else break;
```

```
    }
    else break;
    long long int k = l[index];
    l[index] = l[c];
    l[c] = k;
    m[l[index]] = index;
    index = c;
    m[l[index]] = index;
  }
}

void insert(vector <long long int> &l,long long int n,long
     long int key)
{
  l.push_back(key);
  bottomupheapify(l,n,n-1);
}

void deletemin(vector <long long int> &l, long long int n)
{
  long long int k = l[0];
  l[0] = l[n-1];
  l[n-1] = k;
  m[l[0]] = 0;
  m[l[n-1]] = n-1;
  l.pop_back();
  m.erase(l[n-1]);
  topdownheapify(l,n-1,0);
}

void delete1(vector<long long int> &l,long long int n,long
     long int key)
{
  long long int index = m[key];
  l[index] = l[0]-1;
  bottomupheapify(l,n,index);
  deletemin(l,n);
}

void decreasekey(vector <long long int> &l,long long int n,
     long long int key,long long int value) // Decreases key
      TO value
{
  int index = d[key];
  l[index] = value;
  bottomupheapify(l,n,index);
}

void increasekey(vector <long long int> &l,long long int n,
     long long int key,long long int value)
{
  int index = d[key];
  l[index] = value;
  topdownheapify(l,n,index);
}


int main()
{
  vector<long long int> v;
  long long int q;
  cin>>q;
  long long int n=0;
  for(long long int h=0;h<q;h++)
  {
    long long int a;
    cin>>a;
    if(a==1)
    {
      long long int b;
      cin>>b;
      n++;
      insert(v,n,b);
    }
    else if(a==2)
    {
      long long int b;
      cin>>b;
      delete1(v,n,b);
      n--;
    }
    else if(a==3)
    {
      cout<<v[0]<<endl;
    }
  }
  return 0;
}

/*
18
1 1
1 10 1 2
1 11 1 12 1 3 1 4
1 13 1 14 1 15 1 16 1 5 1 6 1 7 1 8

*/
```

5 DSU

```
const int N = 100100;
int p[N], sz[N];

void create(int x){
 p[x] = x;
 sz[x] = 1;
 return;
}

int find(int x){
  if(x == p[x]) return x;
  return p[x] = find(p[x]);
}

void merge(int x, int y){
 int x = find(x), y = find(y);
 if(x == y) return;
  if(sz[x] < sz[y]) swap(x, y);
  p[y] = x;
  sz[x] += sz[y];
  return;
}

int main(){
 return 0;
}
```

6 DSUTree

```
#include<bits/stdc++.h>
#define mt make_tuple
#define mp make_pair
#define pu push_back
#define INF 1000000001
#define MOD 1000000007
#define ll long long int
#define ld long double
#define vi vector<int>
#define vll vector<long long int>
#define sc(n) scanf("%d",&n);
#define scll(n) scanf("%lld",&n);
#define scld(n) scanf("%Lf",&n);
#define scr(s) {char temp[1000000];scanf("%s",temp);s = temp
    ;}
```

```
using namespace std;

int count[100001] = {0};
int colour[100001] = {0};
bool big[100001] = {false};
vector<int> adj[100001];
int sz[100001] = {0};

void add(int u,int p, int k)
{
  count[colour[u]] +=k;
  for(int i=0;i<adj[u].size();i++)
  {
    if(!big[adj[u][i]] && adj[u][i]!=p) add(adj[u][i],p,k);
  }
}

int dfs(int u,int p)
{
  if(adj[u].size()==1 && adj[u][0]==p) {sz[u] = 1;return 1;
       }
  int count = 0;
  for(int i=0;i<adj[u].size();i++)
  {
    if(adj[u][i]!=p) count+=dfs(adj[u][i],u);
  }
  sz[u] = count;
  return count;
}

void dfs(int u,int p, bool boo)
{
  int max1 = -1, big1 = -1;
  for(int i=0;i<adj[u][i].size();i++)
  {
    if(adj[u][i]!=p)
    {
      if(sz[adj[u][i]]>max1)
      {
        max1 = sz[adj[u][i]];
        big1 =adj[u][i];
        big[adj[u][i]] = true;
      }
    }
  }

  // Do the dfs for small
  for(int i=0;i<adj[u][i].size();i++)
  {
    if(adj[u][i]!=p && ! big[adj[u][i]]) dfs(adj[u][i],u,0);
  }

  // Do the dfs for big after small
  if(big1!=-1) dfs(big1,u,1);

  add(u,p,1);

  // Ans queries here

  // Clear the bigchild if any
  if(big1!=-1) big[big1] = false;

  // Clear if small
  if(!boo) add(u,p,-1);
}

int main()
{

  return 0;
}
```

7 Dijkstra

```
#include<bits/stdc++.h>
#define mt make_tuple
#define mp make_pair
#define pu push_back
#define INF 1000000001
#define MOD 1000000007
#define ll long long int
#define ld long double
#define vi vector<int>
#define vll vector<long long int>
#define sc(n) scanf("%d",&n);
#define scll(n) scanf("%lld",&n);
#define scld(n) scanf("%Lf",&n);
#define scr(s) {char temp[1000000];scanf("%s",temp);s = temp
    ;}

using namespace std;

map <long long int,long long int> d;
map <long long int,long long int> ::iterator it;
long long int* parent;
vector< pair<long long int,long long int> >prt;
vector < pair <long long int,long long int> > *adj;

void addEdge(ll a,ll b,ll c)
{
  adj[a].pu(mp(b,c));
  adj[b].pu(mp(a,c));
}

void bottomupheapify(vector< pair<long long int,long long
    int> > &l,long long int n,long long int index)
{
  d[l[index].first] = index;
  while(index>0 && l[(index-1)/2].second>l[index].second)
  {
    //Swapping for the map
    d[l[index].first] = (index-1)/2;
    d[l[(index-1)/2].first] = index;
    //Swapping the elements
    long long int k = l[index].second;
    long long int k1 = l[index].first;
    l[index].second = l[(index-1)/2].second;
    l[index].first = l[(index-1)/2].first;
    l[(index-1)/2].second = k;
    l[(index-1)/2].first = k1;
    index = (index-1)/2;
  }
}

void topdownheapify(vector< pair<long long int,long long int
    > >&l,long long int n,long long int index)
{
  while(2*index+1<n)
  {
    long long int left = 2*index+1,right = 2*index+2,c;
    if(right<n)
    {
      if(l[left].second<l[index].second && l[left].second<l[
          right].second) c = left;
      else if(l[right].second<l[index].second) c = right;
      else break;
    }
    else if(right>=n)
    {
      if(l[left].second<l[index].second) c = left;
      else break;
    }
    else break;
    long long int k = l[index].second;
    long long int k1 = l[index].first;
    l[index].second = l[c].second;
    l[index].first = l[c].first;
    l[c].second = k;
```

```
    l[c].first = k1;
    d[l[index].first] = index;
    index = c;
    d[l[index].first] = index;
  }
}

void insert(vector< pair<long long int,long long int> >&l,
     long long int n,pair <long long int,long long int> p)
{
  // cout<<"Entered: "<<p.first<<" n: "<<n<<endl;
  l.push_back(p);
  // print(l);
  bottomupheapify(l,n,n-1);
  // print(l);
  // cout<<"Exit: "<<p.first<<" "<<d[p.first]<<endl;
}

void deletemin(vector< pair<long long int,long long int> >&l
     , long long int n)
{
  long long int k = l[0].second;
  long long int k1 = l[0].first;
  l[0].second = l[n-1].second;
  l[0].first = l[n-1].first;
  l[n-1].second = k;
  l[n-1].first = k1;
  d[l[0].first] = 0;
  d[l[n-1].first] = n-1;
  d.erase(l[n-1].first);
  l.pop_back();
  topdownheapify(l,n-1,0);
}

void delete1(vector< pair<long long int,long long int> >&l,
     long long int n,long long int key)
{
  long long int index = d[key];
  l[index].second = l[0].second-1;
  bottomupheapify(l,n,index);
  deletemin(l,n);
}

void dijkstra(long long int s,long long int n)
{
  // cout<<"jere"<<endl;
  vector< pair <long long int,long long int> >heap; //
       Initialising the heap
  // Priority
  long long int* colour = new long long int[n];
  // cout<<"here2"<<endl;
  for(long long int i=0;i<n;i++) {colour[i] = 0;parent[i] =
       i;}
  for(long long int i=0;i<n;i++) prt.pu(mp(i,INF));
  prt[s].second = 0;
  parent[s] = s;
  // Dijkstra initialisation
  heap.pu(prt[s]);
  long long int counter = 1;
  colour[s] = 1;
  // cout<<"here"<<endl;
  while(!heap.empty())
  {
    pair <long long int,long long int> p = heap[0];
    long long int u = p.first;
    long long int pr = p.second;
    // cout<<"Min: "<<u<<" "<<pr<<endl;
    deletemin(heap,counter);
    // print(heap);
    counter--;
    // printm();
    colour[u] = 2;
    // cout<<"u: "<<u<<" pr: "<<pr<<endl;
    for(long long int i=0;i<adj[u].size();i++)
    {
      // printprt();
      // printcolour(colour,n);
      // for(int i=0;i<n;i++)
      // {
      //   cout<<"i: "<<i<<" parent: "<<parent[i]<<endl;
      // }
      long long int v = adj[u][i].first;
      long long int w = adj[u][i].second;
      pair <long long int,long long int> p1;
      ll c = w+pr;

      // if(k<=c%(2*k) && adj[u][i].first!=n-1) c+=(2*k - c
           %(2*k));
      if(colour[v]==0)
      {
        counter++;
        colour[v] = 1;
        p1 = mp(v,c);
        prt[v] = p1;
        // cout<<"Inserting "<<p1.first<<endl;
        insert(heap,counter,p1);
        // cout<<"Succesful! "<<p1.first<<endl;
        // printm();
        parent[v] = u;
      }
      else if(colour[v]==1)
      {
        p1 = mp(v,min(c,prt[v].second));
        if(c<prt[v].second)parent[v] = u;
        prt[v] = p1;
        long long int index = d[v];
        heap[index].second = p1.second;
        // cout<<"Heapify "<<p1.first<<endl;
        topdownheapify(heap,counter,index);
        // cout<<"Succesful Heapify! "<<p1.first<<endl;
        // printm();
      }
    }
    // cout<<endl;
    // print(heap);
  }
}


int main()
{
  int t;
  cin>>t;
  while(t--)
  {
    ll n,q;
    cin>>n>>q;
    // tuple<int,int,int> t;
    parent = new ll[n];
    adj = new vector<pair<ll,ll> >[n];
    set<ll>*visited = new set<ll>[n];
    map<pair<ll,ll>, ll> e;
    map<pair<ll,ll>, ll> :: iterator it1;
    // ll*dp = new int[n];
    // for(int i=0;i<n;i++) prt[i] = INF;
    for(int h=0;h<q;h++)
    {
      ll a,b,c;
      cin>>a>>b>>c;
      a--;b--;
      // cout<<a<<" "<<b<<endl;
      if(a!=b)
      {
        ll a1 = min((int)a,(int)b);
        ll b1 = max((int)a,(int)b);
        a = a1;
        b = b1;
        if(e.find(mp(a,b))!=e.end()) {e[mp(a,b)] = min((int)e
             [mp(a,b)],(int)c);}
        else e[mp(a,b)] = c;
```

```
      // else addEdge(a,b,c);
    }
  }
  for(it1=e.begin();it1!=e.end();it1++)
  {
    pair<ll,ll> p = it1->first;
    ll key= it1->second;
    addEdge(p.first,p.second,key);
  }
  // dp[0] = 0;
  // dfs(adj,visited,0,k,dp,n);
  int source;
  cin>>source;
  dijkstra(source-1,n);
  for(int i=0;i<n;i++)
  {
    if(i!=source-1)
    {
      if(prt[i].second!=INF)cout<<prt[i].second<<" ";
      else cout<<-1<<" ";
    }
  }
  cout<<endl;
  d.clear();
  prt.clear();
  for(int i=0;i<n;i++) adj[i].clear();
  for(int i=0;i<n;i++) cout<<parent[i]+1<<" parent of "<<i
       +1<<" distance is "<<prt[i].second<<endl;
  // cout<<"here"<<endl;
  // printf("%lld\n",prt[n-1].second);
 }
 return 0;
}


/*
5 1 7
0 3 3
2 3 4
1 3 3
1 2 5
4 3 5
1 4 6
2 4 7

8 1 11
1 2 50
2 3 35
3 4 30
3 5 25
2 4 25
0 2 60
6 2 40
4 6 45
0 6 20
6 7 20
0 7 10

0 : 1 7
1 : 0 7 2
2 : 1 5 3 8
3 : 2 5 4
4 : 3 5
5 : 6 2 3 4
6 : 7 5 8
7 : 0 1 8 6
8 : 7 6 2

*/
```

8 GCD

```
int gcd(int a, int b)
{
  if (a == 0)
    return b;
  return gcd(b % a, a);
}
```

9 Graph

```
#include<iostream>
#include<vector>
#include<utility>
#include<queue>
#define pu push_back
#define m make_pair
using namespace std;

// class Graph
// {
//   int n;
//   vector< pair <int,int> > adj;
//   Graph(int vertices)
//   {
//     n = vertices;
//     adj = new vector< pair <int,int> >[n];
//   }
//   addEdge(int u,int v,int key = 1)
//   {
//     adj[u].pu(m(v,key));
//   }
// }

void print(vector<int> *adj,int n)
{
  for(int i=0;i<n;i++)
  {
    cout<<i<<": ";
    for(int j=0;j<adj[i].size();j++)cout<<adj[i][j]<<" ";cout
        <<endl;
  }
}
void addEdge(int u,int v, int key,vector< pair <int,int> >
     adj[])
{
  adj[u].pu(m(v,key));
  adj[v].pu(m(u,key));
  // cout<<adj[u].back().first<<" "<<adj[u].back().second<<
       endl;
}

void bfs(vector < pair <int,int> > adj[],int n)
{
  queue<int> q;
  q.push(1);
  // int n = adj.size();
  int* visited = (int*)calloc(n,sizeof(int));
  while(!q.empty())
  {
    int u = q.front();
    cout<<u<<" ";
    visited[u] = 1;
    q.pop();
    for(int i=0;i<adj[u].size();i++)
    {
      if(visited[adj[u][i].first]==0){q.push(adj[u][i].first)
           ;visited[adj[u][i].first] = 1;}
    }
  }
}

void dfs(vector < pair <int,int> > *adj,int u,int *visited)
{
  if(visited[u]!=0) return;
```

```
  cout<<u<<" ";
  visited[u] = 1;
  for(int i=0;i<adj[u].size();i++)
  {
    dfs(adj,adj[u][i].first,visited);
  }
}
int main()
{
  int n;
  cin>>n;
  vector <pair <int,int> > adj[n];
  int q;
  cin>>q;
  for(int h=0;h<q;h++)
  {
    int a,b,key;
    cin>>a>>b>>key;
    addEdge(a,b,key,adj);
  }

  cout<<"BFS: ";
  bfs(adj,n);
  cout<<endl;
  int* visited1 = (int*)calloc(n,sizeof(int));
  cout<<"DFS: ";
  dfs(adj,1,visited1);
  cout<<endl;
}
```

10 Kruskal

```
// #include<iostream>
// #include<tuple>
// #include<list>
// #include<algorithm>
// #include<iterator>
// #include<vector>
#include<bits/stdc++.h>
#define pu push_back
#define m make_pair
using namespace std;

// Functor to compare by the Mth element
template<int M, template<typename> class F = std::less>
struct TupleCompare
{
    template<typename T>
    bool operator()(T const &t1, T const &t2)
    {
        return F<typename tuple_element<M, T>::type>()(std::
            get<M>(t1), std::get<M>(t2));
    }
};

void addEdge(vector < pair <int,int> > adj[],int u,int v,int
     key)
{
  adj[u].pu(m(v,key));
  adj[v].pu(m(u,key));
}

int main()
{
  int n;
  int q;
  cin>>n>>q;
  vector < pair <int,int> > adj[n];
  vector < tuple <int,int,int> > v;
  for(int h=0;h<q;h++)
  {
    int a,b,key;
    cin>>a>>b>>key;
    a-=1;b-=1;
    // addEdge(a,b,key,adj);
    v.pu(make_tuple(a,b,key));
  }
  sort(begin(v),end(v), TupleCompare<2>());
  int colour[n];
  int number[n];
  list <int> refer[n];
  list <int> :: iterator it;
  int countcolour = n;
  for(int i=0;i<n;i++)
  {
    colour[i] = i;
    number[i] = 1;
    refer[i].pu(i);
  }
  int count = 0;
  for(int i=0;i<v.size()&& countcolour;i++)
  {
    int a = get<0>(v[i]), b = get<1>(v[i]), c = get<2>(v[i]);
    int d = colour[a],e = colour[b];
    if(d==e) continue;
    else if(number[d]>number[e])
    {
      // for(int j=0;j<number[e];j++) colour[refer[e][j]] = d
          ;
      for(it = refer[e].begin();it!=refer[e].end();it++)
          colour[*it] = d;
      number[d]+=number[e];
      number[e] = 0;
      count+=c;
      addEdge(adj,a,b,c);
      countcolour--;
      // cout<<c<<endl;
      refer[d].splice(refer[d].end(),refer[e]);
    }
    else
    {
      // for(int j=0;j<number[d];j++) colour[refer[d][j]] = e
          ;
      for(it = refer[d].begin();it!=refer[d].end();it++)
          colour[*it] = e;
      number[e]+=number[d];
      number[d] = 0;
      count+=c;
      addEdge(adj,a,b,c);
      countcolour--;
      // cout<<c<<endl;
      refer[e].splice(refer[e].end(), refer[d]);
    }
  }
  for(int i=0;i<n;i++)
  {
    cout<<i<<": ";
    for(int j=0;j<adj[i].size();j++) cout<<adj[i][j].first<<"
        ";
    cout<<endl;
  }
  cout<<"Min Weight: "<<count<<endl;
  return 0;
}
/*
int main()
{
    vector<tuple<int, string>> v;
    v.push_back(make_tuple(1, "Hello"));
    v.push_back(make_tuple(2, "Aha"));
    a.splice(a.end(), b); // extends the list a by moving the
        elements from b to the end of a
    std::sort(begin(v), end(v), TupleCompare<0>());
    return 0;
    8 11
1 2 50
2 3 35
```

```
3 4 30
3 5 25
2 4 95
0 2 60
2 6 40
0 6 20
0 7 10
6 7 20
4 6 45

Kruskal.cpp87:37
LF
I
UTF-8C++
master104 files

}
*/
```

11 LCARMQ

```
// I don't think it's completed
#include<iostream>
#include<cmath>
#include<algorithm>
#define INF 1000000001
//#define NUL -1000000001
// #define NUL 0
using namespace std;

// Define the structure you want here
typedef struct
{
  int data;
}node;

void print(int* l,int n)
{
  int prod = 0;
  for(int i=0;i<n;i++)
  {
    if(i+1==pow(2,prod))
    {
      cout<<endl;
      prod++;
    }
    cout<<l[i]<<" ";
  }
  cout<<endl;
}

node lca(node*tree, node a, node b)
{
  node n = find(a.data,b.data)
}

node build(node* tree,int*l,int pos,int beg,int end,int k,
    int n)
{
  // cout<<"pos: "<<pos<<" beg: "<<beg<<" end: "<<end<<" pos
      -k/2: "<<pos-k/2<<endl;
  //Empty node
  node NUL;
  NUL.data = INF;
  //Define The null pointer here
  // Eg. NUL.maxsum = 0;NUL.maxnumber = 0;
  if(beg<=end)
  {
    if (pos-k/2>=n)return NUL;
    //Add the leaf node here
    // Eg.{tree[pos].maxsum = l[pos-k/2];tree[pos].maxnumber
        = l[pos-k/2];return tree[pos];} //Leaf node
    if(beg==end){tree[pos].data = l[pos-k/2];return tree[pos
        ];}
    int mid = (beg+end)/2;
    node n1,n2;
    n1 = build(tree,l,2*pos+1,beg,mid,k,n);
    n2 = build(tree,l,2*pos+2,mid+1,end,k,n);
    tree[pos].data = min(n1.data,n2.data);
    //This is where the recursive formulae come in
    // tree[pos].maxsum = max(max(n1.maxsum,n2.maxsum),n1.
        maxnumber+n2.maxnumber); //The recursive formulae
    // tree[pos].maxnumber = max(n1.maxnumber,n2.maxnumber);
    return tree[pos];
  }
  else //NULL version of the structure
  {
    return NUL;
  }
}

node update(node*tree,int index,int key,int pos,int beg,int
    end)
{
  int mid = (beg+end)/2;
  // cout<<"pos: "<<pos<<" beg: "<<beg<<" end: "<<end<<" mid
      : "<<mid<<endl;
  // Define The NUL pointer here(for the blank nodes)
  // Eg. NUL.maxsum = 0;NUL.maxnumber = 0;
  node NUL;
  NUL.data = INF;
  if(beg<=end)
  {
    if(beg==end){tree[pos].data = key;return tree[pos];} //
        Leaf nodes
    int mid = (beg+end)/2;
    node n1,n2;
    if(index<=mid)
    {
      n1 = update(tree,index,key,2*pos+1,beg,mid);
      n2 = tree[2*pos+2];
      tree[pos].data = min(n1.data,n2.data);
      // tree[pos].maxsum = max(max(n1.maxsum,n2.maxsum),n1.
          maxnumber+n2.maxnumber); //The recursive formulae
      // tree[pos].maxnumber = max(n1.maxnumber,n2.maxnumber)
          ;
    }
    else
    {
      n1 = tree[pos*2+1];
      n2 = update(tree,index,key,2*pos+2,mid+1,end);
      tree[pos].data = min(n1.data,n2.data);
      // tree[pos].maxsum = max(max(n1.maxsum,n2.maxsum),n1.
          maxnumber+n2.maxnumber); //The recursive formulae
      // tree[pos].maxnumber = max(n1.maxnumber,n2.maxnumber)
          ;
    }
    return tree[pos];
  }
  else
  {
    return NUL;
  }
}

node find(node* tree,int l,int r,int beg,int end,int pos,int
    k,int n)
{
  //3 cases Go left, go right, split up
  int mid = (beg+end)/2;
  // cout<<"beg: "<<beg<<" end: "<<end;
  // cout<<" mid: "<<mid<<" pos: "<<pos<<endl;
  if(l==beg&&r==end) return tree[pos];
  if(r<=mid) return find(tree,l,r,beg,mid,2*pos+1,k,n);
  else if(l>mid) return find(tree,l,r,mid+1,end,2*pos+2,k,n)
      ;
  else
  {
```

```
    node n1,n2,n3;
    n1 = find(tree,l,mid,beg,mid,2*pos+1,k,n);
    n2 = find(tree,mid+1,r,mid+1,end,2*pos+2,k,n);
    n3.data = min(n1.data,n2.data);
    // n3.maxsum = max(max(n1.maxsum,n2.maxsum),n1.maxnumber+
        n2.maxnumber); //The recursive formulae
    // n3.maxnumber = max(n1.maxnumber,n2.maxnumber);
    return n3;
  }
}

int main()
{
  // cout<<"There are 2 queries to find the max, 1 <index> <
      insert value>, 2 <Left> <Right> "<<endl;
  int n;
  cin>>n;
  int l[n];
  for(int i=0;i<n;i++)cin>>l[i];
  int k = 2*pow(2,ceil(log(n)/log(2.0)))-1;
  node* tree = (node*)calloc(k,sizeof(node));
  node NUL; //Define the blank nodes
  NUL.data = INF;
  for(int i=0;i<k;i++)tree[i] = NUL;
  build(tree,l,0,0,(k+1)/2 - 1,k,n);
  // for(int i=0;i<k;i++)cout<<"i: "<<i<<" maxsum: "<<tree[i
      ].maxsum<<" maxnumber: "<<tree[i].maxnumber<<endl;
  int q;
  cin>>q;
  for(int i=0;i<q;i++)
  {
    char q1;
    cin>>q1;
    if(q1=='U')
    {
      int index,value;
      cin>>index>>value;
      update(tree,index-1,value,0,0,(k+1)/2 - 1);
      // for(int i=0;i<k;i++)cout<<"i: "<<i<<" maxsum: "<<
          tree[i].maxsum<<" maxnumber: "<<tree[i].maxnumber
          <<endl;;
    }
    else if(q1=='Q')
    {
      int beg,end;
      cin>>beg>>end;
      beg-=1;end-=1;
      cout<<find(tree,beg,end,0,(k+1)/2 - 1,0,k,(k+1)/2).data
          <<endl;
    }
  }
  return 0;
}
```

12 Longest Palindrome

```
#include<iostream>
#include<string>
#include<algorithm>
using namespace std;

int rollinghash(int,string,int);
int bsearch(int,string);
int main()
{
  string s;
  cin>>s;
  int n = s.length();
  int left=0,right=n;
  int a = bsearch(1,s);
  int b = bsearch(0,s);
  cout<<a<<" "<<b<<endl;
  cout<<max(a,b)<<endl;
  return 0;
}

int rollinghash(int size,string s,int max) //This commputes
    if a palindrome of given size exists or not.
{
  //int l[4084061];
  cout<<"size: "<<size<<endl;
  int MOD = 4084061;
  //fill(array, array+4084061, -1);
  int n = s.length();
  long long int hash=0,hashr = 0;
  long long int k = 27,prod=1;
  reverse(s.begin(),s.end());
  string t = s;
  reverse(s.begin(),s.end());
  int boo=-1;
  for(int i=0;i<size;i++)
  {
    hash*=k;
    hash+=(s[i]);hash%=MOD;
    hashr*=k;
    hashr+=(t[i]);hashr%=MOD;
    if(i<size-1)prod*=k;prod%=MOD;
  }
  cout<<prod<<endl;
  int* l = (int*)calloc(n-size+1,sizeof(int));
  for(int i=0;i<n-size+1;i++)
  {
    l[i] = hashr;
    hashr-=t[i]*prod;
    hashr = ((hashr%MOD)+MOD)%MOD;
    hashr*=k;
    hashr%=MOD;
    if(i<n-size)hashr+=t[i+size];
    hashr%=MOD;
  }
  for(int i=0;i<n-size+1;i++)cout<<l[i]<<" ";
  cout<<endl<<hash<<endl;
  for(int i=0;i<n-size+1;i++)
  {
    cout<<hash<<" "<<l[n-i-size]<<endl;
    if(hash==l[n-i-size]) {boo = i;cout<<"boo";break;}
    hash-=s[i]*prod;
    hash = ((hash%MOD)+MOD)%MOD;hash*=k;
    if(i<n-size)hash+=s[i+size];
    hash%=MOD;
  }
  cout<<hash<<" "<<l[boo]<<" ash "<<boo<<endl;
  if(boo!=-1) {max = size;cout<<hash<<" "<<hashr<<" ash "<<
      boo<<endl;return max;}
  return 0;
}

int bsearch(int c,string s)
{
  int max = 0;
  int left = 0,right = s.length();
  int mid = (left+right)/2;
  while(left<=right&&mid<right)
  {
    cout<<mid<<" "<<c<<" "<<left<<" "<<right<<endl;
    if(mid%2==c)mid+=1;
    cout<<mid<<" "<<c<<" "<<left<<" "<<right<<endl;
    int k = rollinghash(mid,s,0);
    cout<<k<<endl;
    if(k!=0) {if(k>max)max = k;left = mid+1;}
    else right = mid-1;
    mid = (left+right)/2;
  }
  return max;
}
```

13 MillerRabin

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

int power(int x, int y, int p)
{
    int res = 1;     // Initialize result
    x = x % p; // Update x if it is more than or
    while (y > 0)
    {
        if (y & 1) res = (res*x) % p;
        y = y>>1; // y = y/2
        x = (x*x) % p;
    }
    return res;
}

int MillerRabin(int n)
{
  int p = n;
  p--;
  int prod = 1;
  int s = 0;
  while(p%2==0)
  {
    p/=2;
    prod*=2;
    s++;
  }
  cout<<"n: "<<n<<" p: "<<p<<" s: "<<s<<endl;
  int l[3] = {2,7,61};
  int boo = 0;
  for(int i=0;i<3;i++)
  {
    for(int j=0;j<s;j++)
    {
      if(l[i]>=n) continue;
      cout<<"l[i]: "<<l[i]<<" p: "<<p<<" n: "<<n<<" power(l[i
          ],p,n): "<<power(l[i],p,n)<<endl;
      cout<<"j: "<<j<<" power(2,j,n): "<<power(2,j,n)<<"
          power(l[i],pow*p,n): "<<power(l[i],power(2,j,n)*p,
          n) <<endl;
      if(power(l[i],p,n)!=1 && power(l[i],power(2,j,n)*p,n)
          != n-1) continue;
      else return 1;
    }
  }
  return 0;
}

int main()
{
  // for(int i=3;i<100;i++) if(MillerRabin(i)) cout<<i<<" ";
  // cout<<endl;
  int p;
  cin>>p;
  cout<<MillerRabin(p)<<endl;
  return 0;
}
```

14 Mo Algorithm

```
#include<bits/stdc++.h>
#define mt make_tuple
#define mp make_pair
#define pu push_back
#define INF 1000000001
#define MOD 1000000007
#define ll long long int
#define ld long double
#define vi vector<int>
#define vll vector<long long int>
#define fi first
#define se second
#define sc(n) scanf("%d",&n);
#define scll(n) scanf("%lld",&n);
#define scld(n) scanf("%Lf",&n);
#define scr(s) {char temp[1000000];scanf("%s",temp);s = temp
    ;}

using namespace std;

int block;

typedef struct node
{
  int first,second,i;
}node;

bool sorter(const node &a, const node &b)
{
  if(a.fi/block < b.fi/block) return true;
  else if(a.fi/block> b.fi/block) return false;
  else
  {
    if(a.se<b.se) return true;
    return false;
  }
}

// int bring(int dest,int src, int*counter,int*l,int boo)
// {
//   // cout<<"dest: "<<dest<<" src: "<<src<<" boo: "<<boo<<
    endl;
//   int count = 0;
//   while(dest!=src)
//   {
//     if((dest>src))
//     {
//       if(!boo)
//       {
//         if(src==-1) {src++;continue;}
//         if(counter[l[src]]==1) count--;
//         counter[l[src]]--;
//         src++;
//       }
//       else
//       {
//         src++;
//         counter[l[src]]++;
//         if(counter[l[src]]==1) count++;
//       }
//     }
//     else
//     {
//       if(!boo)
//       {
//         src--;
//         if(src<=-1) break;
//         counter[l[src]]++;
//         if(counter[l[src]]==1) count++;
//       }
//       else
//       {
//         if(counter[l[src]]==1) count--;
//         counter[l[src]]--;
//         src--;
//       }
//     }
//   }
//   return count;
// }

int add(int*counter,int*l,int pos)
{
  int count = 0;
```

```
  counter[l[pos]]++;
  if(counter[l[pos]]==1) count++;
  return count;
}

int remove(int*counter,int*l,int pos)
{
  int count =0 ;
  counter[l[pos]]--;
  if(counter[l[pos]]==0) count--;
  return count;
}

int main()
{
  // ios_base::sync_with_stdio(0);cin.tie(0);
  int n;
  sc(n);
  int*l = new int[n];
  int*counter = (int*)calloc(1000001,sizeof(int));
  for(int i=0;i<n;i++) sc(l[i]);
  int q;
  sc(q);
  int ans[q];
  node *arr = new node[q];
  for(int i=0;i<q;i++)
  {
    sc(arr[i].fi);
    sc(arr[i].se);
    arr[i].fi--;arr[i].se--;
    arr[i].i = i;
  }

  block = int(sqrt(n));
  sort(arr,arr+q,sorter);

  int left = -1,right = -1,count = 0;
  for(int i=0;i<q;i++)
  {
    // node p = arr[i];
    // int a = p.fi,b = p.se;
    // count+=bring(a,left,counter,l,0);
    // left = a;
    // count+=bring(b,right,counter,l,1);
    // right = b;
    // ans[p.i] = count;
    node p = arr[i];
    int a = p.fi,b = p.se;
    while(left>a)
    {
      left--;
      count+=add(counter,l,left);
    }
    while(left<a)
    {
      if(left==-1) {left++;continue;}
      count+=remove(counter,l,left);
      left++;
    }
    while(right<b)
    {
      right++;
      count+=add(counter,l,right);
    }
    while(right>b)
    {
      count+=remove(counter,l,right);
      right--;
    }
    ans[p.i] = count;
  }

  for(int i=0;i<q;i++)
  {
    printf("%d\n",ans[i]);
  }
  return 0;
}
```

15 ModularExponentiation

```
#include<bits/stdc++.h>
#define mt make_tuple
#define mp make_pair
#define pu push_back
#define INF 1000000001
#define ll long long int
#define vi vector<int>

using namespace std;

ll power(ll x, ll y, ll p)
{
    ll res = 1;     // Initialize result
    x = x % p; // Update x if it is more than or
    while (y > 0)
    {
        if (y & 1)
            res = (res*x) % p;
        y = y>>1; // y = y/2
        x = (x*x) % p;
    }
    return res;
}

int main()
{
  cout<<power(3,10000,100)<<endl;
  return 0;
}
```

16 Nextsmallest

```
#include<bits/stdc++.h>
#define MAX 1000000001
using namespace std;

void nextsmallest(int l[],int m[],int n)
{
  map<int,int> d;
  for(int i=0;i<n;i++) d[l[i]] = i;
  stack<int> s;
  for(int i=0;i<n;i++)
  {
    if(!s.empty() && l[i]<s.top())
    {
      while(!s.empty() && l[i]<s.top())
      {
        m[d[s.top()]] = i;
        s.pop();
      }
      s.push(l[i]);
    }
    else s.push(l[i]);
  }
}

int main()
{
  int n;
  cin>>n;
  int l[n];
  for(int i=0;i<n;i++) cin>>l[i];
  int m[n];
  for(int i=0;i<n;i++) m[i] = MAX;
  nextsmallest(l,m,n);
```

```
  for(int i=0;i<n;i++) cout<<m[i]<<" ";
  cout<<endl;
  return 0;
}
```

17 PaliwalSegementTree

```
#include <cstdio>
#include <iostream>
#include <cmath>
#include <algorithm>
#include <cstring>
#include <map>
#include <set>
#include <vector>
#include <utility>
#include <queue>
#include <stack>

#define sd(x) scanf("%d",&x)
#define sd2(x,y) scanf("%d%d",&x,&y)
#define sd3(x,y,z) scanf("%d%d%d",&x,&y,&z)

#define fi first
#define se second
#define pb(x) push_back(x)
#define mp(x,y) make_pair(x,y)
#define LET(x, a) __typeof(a) x(a)
#define foreach(it, v) for(LET(it, v.begin()); it != v.end()
    ; it++)

#define _ ios_base::sync_with_stdio(false);cin.tie(NULL);
    cout.tie(NULL);
#define __ freopen("input.txt","r",stdin);freopen("output.
    txt","w",stdout);

#define tr(x) cout<<x<<endl;
#define tr2(x,y) cout<<x<<" "<<y<<endl;
#define tr3(x,y,z) cout<<x<<" "<<y<<" "<<z<<endl;
#define tr4(w,x,y,z) cout<<w<<" "<<x<<" "<<y<<" "<<z<<endl;
#define tr5(v,w,x,y,z) cout<<v<<" "<<w<<" "<<x<<" "<<y<<" "
    <<z<<endl;
#define tr6(u,v,w,x,y,z) cout<<u<<" "<<v<<" "<<w<<" "<<x<<"
    "<<y<<" "<<z<<endl;

using namespace std;

const int N = 1 << 17;
```

```
struct node{
 int cnt;
 void assign(int value){
  cnt = value;
 }
 void update(int value){
  cnt += value;
 }
 void combine(node &left, node &right){
  cnt = left.cnt + right.cnt;
 }
};

int n, a[N], lazy[N];
node tree[2*N];

// [l, r)
void build(int id = 1, int l = 0, int r = n)
{
 if(l+1 == r){
  tree[id].assign(a[l]);
  return;
 }
 int left = id<<1, right = left+1, mid = (l+r)>>1;

 build(left, l, mid); build(right, mid, r);

 tree[id].combine(tree[left], tree[right]);
 return;
}

// point update -> update(index, value);
void update(int index, int val, int id = 1, int l = 0, int r
     = n)
{
 if(l+1 == r){
  tree[id].assign(val);
  return;
 }
 int left = id<<1, right = left+1, mid = (l+r)>>1;

 if(index < mid) update(index, val, left, l, mid);
 else update(index, val, right, mid, r);

 tree[id].combine(tree[left], tree[right]);
}


// range update and utility functions
```

```
void upd(int id,int l,int r,int x)
{ // update the current node and its index in the lazy array
 lazy[id] += x;
 tree[id].update((r - l) * x);
}

void shift(int id,int l,int r)
{ //propogate update information to the children
 if(lazy[id] and l+1 < r){
  int mid = (l+r)/2;
  upd(id * 2, l, mid, lazy[id]);
  upd(id * 2 + 1, mid, r, lazy[id]);
  lazy[id] = 0; // passing is done, reset the index in the
       lazy array
 }
}

// range update -> update(x, y, val);
void update(int x, int y, int val, int id = 1, int l = 0,
     int r = n)
{
 if(x >= r or l >= y) return;
 if(x <= l && r <= y){
  upd(id, l, r, val);
  return;
 }

 shift(id, l, r); // pass the updates to the children

 int left = id<<1, right = left+1, mid = (l+r)>>1;

 update(x, y, val, left, l, mid);
 update(x, y, val, right, mid, r);

 tree[id].combine(tree[left], tree[right]);
 return;
}

// range query -> query(x, y);
// for point query, traverse like in point update
int query(int x, int y, int id = 1, int l = 0, int r = n)
{
 if(x >= r or l >= y) return 0;
 if(x <= l && r <= y) return tree[id].cnt;

 shift(id, l, r); //use this with lazy propogation

 int left = id<<1, right = left+1, mid = (l+r)>>1;
```

```
 return query(x, y, left, l, mid) + query(x, y, right, mid,
     r);
}

int main()
{
 return 0;
}
```

18 Permutations

```
#include<bits/stdc++.h>
#define mt make_tuple
#define mp make_pair
#define pu push_back
#define INF 1000000001
#define MOD 1000000007
#define ll long long int
#define ld long double
#define vi vector<int>
#define vll vector<long long int>
#define sc(n) scanf("%d",&n);
#define scll(n) scanf("%lld",&n);
#define scld(n) scanf("%Lf",&n);
#define scr(s) {char temp[1000000];scanf("%s",temp);s = temp
    ;}

using namespace std;

void recur(set<int> s, vector<int> v,int n)
{
  if(s.empty())
  {
    for(int i=0;i<v.size();i++) cout<<v[i]<<" ";
    cout<<endl;
    return;
  }

  for(int i=1;i<=n;i++)
  {
    if(s.find(i)!=s.end())
    {
      v.pu(i);
      s.erase(i);
      recur(s,v,n);
      v.pop_back();
      s.insert(i);
    }
  }
}

int main()
{
  int n;
  string s;
  while(getline(cin,s))
  {
    n = stoi(s);
    set<int> s1;
    vector<int> v;
    for(int i=1;i<=n;i++) s1.insert(i);
    recur(s1,v,n);
  }
  return 0;
}
```

19 SegmentTree

```
#include<iostream>
#include<cmath>
#include<algorithm>
//#define NUL -1000000001
#define NUL 0
using namespace std;

void print(int* l,int n)
{
  int prod = 0;
  for(int i=0;i<n;i++)
  {
    if(i+1==pow(2,prod))
    {
      cout<<endl;
      prod++;
    }
    cout<<l[i]<<" ";
  }
  cout<<endl;
}

int build(int* tree,int*l,int pos,int beg,int end,int k,int
    n)
{
  cout<<"pos: "<<pos<<" beg: "<<beg<<" end: "<<end<<" pos-k
      /2: "<<pos-k/2<<" k: "<<k<<endl;
  if(beg<=end)
  {if (pos-k/2>=n)return NUL;
  if(beg==end ){tree[pos] = l[pos-k/2];return l[pos-k/2];}
  // else if(beg==end) return NUL;
  int mid = (beg+end)/2;
  tree[pos] = max(build(tree,l,2*pos+1,beg,mid,k,n),build(
      tree,l,2*pos+2,mid+1,end,k,n));
  return tree[pos];}
  else return NUL;
}

int update(int*tree,int index,int key,int pos,int beg,int
    end)
{
  // cout<<"pos: "<<pos<<" beg: "<<beg<<" end: "<<end<<endl;
  if(beg==end){tree[pos] = key;return tree[pos];}
  int mid = (beg+end)/2;
  if(index<=mid) tree[pos] = max(update(tree,index,key,2*pos
      +1,beg,mid),tree[2*pos+2]);
  else tree[pos] = max(tree[pos*2+1],update(tree,index,key
      ,2*pos+2,mid+1,end));
  return tree[pos];
}

int find(int* tree,int l,int r,int beg,int end,int pos,int k
    ,int n)
{
  //3 cases Go left, go right, split up
  int mid = (beg+end)/2;
  cout<<"beg: "<<beg<<" end: "<<end;
  cout<<" mid: "<<mid<<endl;
  if(l==beg&&r==end) return tree[pos];
  if(r<=mid) return find(tree,l,r,beg,mid,2*pos+1,k,n);
  else if(l>mid) return find(tree,l,r,mid+1,end,2*pos+2,k,n)
      ;
  else
  {
    return max(find(tree,l,mid,beg,mid,2*pos+1,k,n),find(tree
        ,mid+1,r,mid+1,end,2*pos+2,k,n));
  }
}

int main()
{
  // cout<<"There are 2 queries to find the max, 1 <index> <
      insert value>, 2 <Left> <Right> "<<endl;
  int n;
  cin>>n;
  int l[n];
  for(int i=0;i<n;i++)cin>>l[i];
  int k = 2*pow(2,ceil(log(n)/log(2.0)))-1;
```

```
  int* tree = (int*)calloc(k,sizeof(int));
  for(int i=0;i<k;i++)tree[i] = NUL;
  build(tree,l,0,0,k/2-1,k,n);
  for(int i=0;i<k;i++)cout<<tree[i]<<" ";
  cout<<endl;
  int q;
  cin>>q;
  for(int i=0;i<q;i++)
  {
    int q1;
    cin>>q1;
    if(q1==1)
    {
      int index,value;
      cin>>index>>value;
      update(tree,index-1,value,0,0,k/2-1);
      // for(int j=0;j<k;j++)cout<<tree[j]<<" ";
      // cout<<endl;
    }
    else if(q1==2)
    {
      int beg,end;
      cin>>beg>>end;
      beg-=1;end-=1;
      cout<<find(tree,beg,end,0,k/2-1,0,k,n)<<endl;
    }
  }
  return 0;
}
```

20 SegmentTreeStructureTemplate

```
#include<bits/stdc++.h>
#define mt make_tuple
#define mp make_pair
#define pu push_back
#define INF 1000000001
#define MOD 1000000007
#define ll long long int
#define ld long double
#define vi vector<int>
#define vll vector<long long int>
#define sc(n) scanf("%d",&n);
#define scll(n) scanf("%lld",&n);
#define scld(n) scanf("%Lf",&n);
#define scr(s) {char temp[1000000];scanf("%s",temp);s = temp
    ;}
using namespace std;

// Define the structure you want here
struct node
{
  int a;
  void combine(node &n1,node &n2)
  {
    // The recursive formulae here
    //maxsum = max(max(n1.maxsum,n2.maxsum),n1.maxnumber+n2.
        maxnumber); //The recursive formulae
  }
  void assign(int value)
  {
    // Assigning values to the leaf nodes
  }
};

// Define the NUL node structure here
node generate()
{
  node NUL;
  // Eg NUL.maxsum = 0
  return NUL;
}

node build(node* tree,int*l,int pos,int beg,int end,int k,
    int n)
{
  // cout<<"pos: "<<pos<<" beg: "<<beg<<" end: "<<end<<" pos
      -k/2: "<<pos-k/2<<endl;
  node NUL = generate();
  if(beg<=end)
  {
    if (pos-k/2>=n)return NUL;
    // Here take care of the leaf nodes condition. Also,
        remember to return tree[pos].
    int value = l[pos-k/2];
    if(beg==end){tree[pos].assign(value);return tree[pos];}
        //Leaf node
    int mid = (beg+end)/2;
    node n1,n2;
    n1 = build(tree,l,2*pos+1,beg,mid,k,n);
    n2 = build(tree,l,2*pos+2,mid+1,end,k,n);
    tree[pos].combine(n1,n2);
    return tree[pos];
  }
  else //NULL version of the structure
  {
    return NUL;
  }
}

node update(node*tree,int index,int key,int pos,int beg,int
    end)
{
  int mid = (beg+end)/2;
  // cout<<"pos: "<<pos<<" beg: "<<beg<<" end: "<<end<<" mid
      : "<<mid<<endl;
  node NUL = generate();
  if(beg<=end)
  {
    // Here take care of the leaf nodes condition. Also,
        remember to return tree[pos]
    int value = l[pos-k/2];
    if(beg==end){tree[pos].update(value);return tree[pos];}
        //Leaf node
    int mid = (beg+end)/2;
    node n1,n2;
    if(index<=mid)
    {
      n1 = update(tree,index,key,2*pos+1,beg,mid);
      n2 = tree[2*pos+2];
      tree[pos].combine(n1,n2);
    }
    else
    {
      n1 = tree[pos*2+1];
      n2 = update(tree,index,key,2*pos+2,mid+1,end);
      tree[pos].combine(n1,n2);
    }
    return tree[pos];
  }
  else
  {
    return NUL;
  }
}

node find(node* tree,int l,int r,int beg,int end,int pos,int
    k,int n)
{
  //3 cases Go left, go right, split up
  int mid = (beg+end)/2;
  if(l==beg&&r==end) return tree[pos];
  if(r<=mid) return find(tree,l,r,beg,mid,2*pos+1,k,n);
  else if(l>mid) return find(tree,l,r,mid+1,end,2*pos+2,k,n)
      ;
  else
  {
```

```
    node n1,n2,n3;
    n1 = find(tree,l,mid,beg,mid,2*pos+1,k,n);
    n2 = find(tree,mid+1,r,mid+1,end,2*pos+2,k,n);
    n3.combine(n1,n2);
    return n3;
  }
}

int main()
{
  typedef struct node node;
  int n;
  sc(n);
  int l[n];
  for(int i=0;i<n;i++) sc(l[i]);
  int k = 2*pow(2,ceil(log(n)/log(2.0)))-1;
  node* tree = (node*)calloc(k,sizeof(node));
  node NUL = generate();
  for(int i=0;i<k;i++)tree[i] = NUL;
  build(tree,l,0,0,(k+1)/2 - 1,k,n);
  // for(int i=0;i<k;i++)cout<<"i: "<<i<<" maxsum: "<<tree[i
       ].maxsum<<" maxnumber: "<<tree[i].maxnumber<<endl;
  int q;
  cin>>q;
  for(int i=0;i<q;i++)
  {
    int q1;
    sc(q1);
    if(q1==0)
    {
      ll index,value;
      cin>>index>>value;
      update(tree,index-1,value,0,0,(k+1)/2 - 1);
      // for(int i=0;i<k;i++)cout<<"i: "<<i<<" maxsum: "<<
           tree[i].maxsum<<" maxnumber: "<<tree[i].maxnumber
           <<endl;;
    }
    else if(q1==1)
    {
      int beg,end;
      cin>>beg>>end;
      beg-=1;end-=1;
      cout<<find(tree,beg,end,0,(k+1)/2 - 1,0,k,(k+1)/2).
           maxsum<<endl;
    }
  }
  return 0;
}
```

21 Sieve

```
#include<bitset>
#include<iostream>
#include<vector>
#define mt make_tuple
#define mp make_pair
#define pu push_back
#define INF 1000000001
#define ll long long int
#define vi vector<int>

using namespace std;

long long int size;
bitset<10000010> bs;
vi prime;

void sieve(ll upperbound)
{
  size = upperbound + 1;
  bs.set();
  bs[0] = bs[1] = false;
  for(ll i = 2;i< size;i++)
  {
    if(bs[i])
    {
      for(ll j = i*i;j<size;j+=i) bs[j] = false;
    }
    prime.pu((int)i);
  }
}

bool isPrime(ll N)
{
  if(N<size) return bs.test(N);
  for(int i=0;i<primes.size();i++) if(N%primes[i]==0) return
       false;
  return true;
}

int main()
{
  sieve(10000000);
  ll n;
  cin>>n;
  cout<<isPrime(n)<<endl;
}
```

22 SuffixArray

```
#include<bits/stdc++.h>
#define mt make_tuple
#define mp make_pair
#define pu push_back
#define INF 1000000001
#define MOD 1000000007
#define ll long long int
#define ld long double
#define vi vector<int>
#define vll vector<long long int>
#define sc(n) scanf("%d",&n);
#define scll(n) scanf("%lld",&n);
#define scld(n) scanf("%Lf",&n);
#define scr(s) {char temp[1000000];scanf("%s",temp);s = temp
     ;}

using namespace std;

map<pair<int,int>,int > d;

void countSort(vector<pair<int,int> > &v)
{
  vector<pair<int,int> > *adj1 = new vector<pair<int,int> >[
       n];
  vector<pair<int,int> > *adj2 = new vector<pair<int,int> >[
       n];
  for(int i=0;i<v.size();i++)
  {
    adj1[v[i].second].pu(v[i]);
  }
  v.clear();
  for(int i=0;i<=n;i++)
  {
    for(int j=0;j<adj1[i].size();j++)
    {
      v.pu(adj1[i][j]);
    }
  }
  for(int i=0;i<v.size();i++)
  {
    adj2[v[i].second].pu(v[i]);
  }
  v.clear();
  int count = 0;
  for(int i=0;i<=n;i++)
  {
    for(int j=0;j<adj2[i].size();j++)
    {
```

```
      if(d.find(adj2[i][j])!=d.end())
      {
        count++;
        d[adj2[i][j]] = count;
      }
      v.pu(adj2[i][j]);
    }
  }
}

void findRank(int*l, string s)
{
  int* rank = (int*)calloc(130,sizeof(int));
  for(int i=0;i<s.size();i++)
  {
    rank[s[i]] = 1;
  }
  int count = 0;
  for(int i=0;i<130;i++) {count+=s[i];s[i] = count;}
  for(int i=0;i<s.size();i++)
  {
    l[i] = rank[s[i]];
  }
}

int main()
{
  string s;
  scr(s);
  s+='$';
  int*rank = (int*)calloc(n,sizeof(int));

  return 0;
}
```

23 check

```
file1 = "Stress.out"
file2 = "Answer.out"

s = open(file1, 'r').readlines()
t = open(file2, 'r').readlines()

for i,j in zip(s,t):
    if(i[:len(i)-2]+i[len(i)-1]==j):
        continue
    else:
        print len(i),len(j)
        print i,j
    print "Here"
```

24 inversion

```
#include<stdio.h>
#include<stdlib.h>

void sort(int* ,int,int);
void merge(int* ,int,int,int);
long long int count = 0;
int main()
{
  int t;
  scanf("%d",&t);
  char s[5];
  for(int _=0;_<t;_++)
  {
    int n;
    count = 0;
    // scanf("%s",s);
    scanf("%d",&n);
    int l[n];
    for(int i=0;i<n;i++)
    {
      scanf("%d",&l[i]);
    }
    //int **p;
    //*p = &(*l);
    //printf("%d",l[0]);
    sort(l,0,n-1);
    // for(int i=0;i<n;i++)
    // {
    //   printf("%d ",l[i]);
    // }
    // printf("\n");
    printf("%lld\n",count);
  }
  return 0;
}

void sort(int *l,int i,int j)
{
  //int *l = *p;
  if(i<j)
  {
    int c = (i+j)/2;
    sort(l,i,c);
    sort(l,c+1,j);
    merge(l,i,c,j);
  }
    // printf("%d %d Indices \t",i,j);
    // for(int k=i;k<j+1;k++)
    // {
    //   printf("%d ",l[k]);
    // }
    // printf("\n");
    // printf("%lld\n",count);
}

void merge(int*l,int i,int c,int j)
{
  //int *l = *p;
  int*m = (int*)calloc((j-i+1),sizeof(int));
  int x=i,y=c+1;
  int k=0;
  while(k<j-i+1)
  {
    if(y==j+1)
    {
      while(x<=c)
      {
        m[k] = l[x];
        count+=(y-c-1);
        x++;
        k++;
      }
      continue;
    }
    else if(x==c+1)
    {
      while(y<=j)
      {
        m[k] = l[y];
        y++;
        k++;
      }
      continue;
    }
    else if(l[x]<l[y])
    {
      m[k] = l[x];
      count+=(y-c-1);
      x++;
    }
    else if(l[y]<=l[x])
    {
      m[k] = l[y];
```

```
      y++;
    }
    k++;
  }
  //Copying
  for(int k=0;k<j-i+1;k++)
  {
    l[k+i] = m[k];
  }
}
```

25 nCr

```
long long int nCr(int n,int k) //Not my code
{
    long long int ans=1;
    k=k>n-k?n-k:k;
    int j=1;
    for(;j<=k;j++,n--)
    {
        if(n%j==0)
        {
            ans*=n/j;
        }else
        if(ans%j==0)
        {
            ans=ans/j*n;
        }else
        {
            ans=(ans*n)/j;
        }
    }
    return ans;
}

// Less constraints, but higher usage of repeated result=\
];

int dp[max][max];
//Initialise array elements with zero
int nCr(int n, int r)
{
       if(n==r) return dp[n][r] = 1; //Base Case
       if(r==0) return dp[n][r] = 1; //Base Case
       if(r==1) return dp[n][r] = n;
       if(dp[n][r]) return dp[n][r]; // Using Subproblem
            Result
       return dp[n][r] = nCr(n-1,r) + nCr(n-1,r-1);
}
```

26 primeFactors

```
void primeFactors(int n,vector<int> &v)
{
  while (n%2 == 0)
  {
    v.pu(2);
    n = n/2;
  }

  for (int i = 3; i <= sqrt(n); i = i+2)
  {
    while (n%i == 0)
    {
      v.pu(i);
      n = n/i;
    }
  }

  if (n > 2) v.pu(n);
}
```

27 rabinkarp

```
#include<stdio.h>
#include<string.h>
int bruteforce(char*,char*,int,int);
void rabinkarp(char*,char*,int,int);

int main()
{
  int n,m;
  scanf("%d",&n);
  char s[n];
  scanf("%s",s);
  scanf("%d",&m);
  char t[m];
  scanf("%s",t);
  printf("%d\n",bruteforce(s,t,n,m));
  //rabinkarp(s,t,n,m);
  return 0;
}
int bruteforce(char* s,char* t,int n,int m)
{
  int count = 0,j=0;
  for(j=0;j<n-m+1;j++)
  {
    count = 0;
    for(int i=0;i<m;i++)
    {
      if (s[i+j]==t[i]) count++;
      else break;
    }
    if(count==m)return 1;
  }
  return 0;
}

void rabinkarp(char* s,char*t,int n,int m)
{
  int* hash[1000000];
  int d = 26;
  int hasht=0,hashs=0;
  int p=1001;
  int prod=1;
  for(int i=0;i<m;i++)
  {
    hashs+=(s[i]-'0')*prod;
    hasht+=(t[i]-'0')*prod;
    prod*=d;hashs%=p;hasht%=p;prod%=p;
  }
}

typedef struct
{
  char* data;
  s1 *next;
} s1;
```

28 stack

```
l = map(int,raw_input().split())
max1 = 0
m = []
for i in range(len(l)):
    #print m
    if l[i]>max1:
        m.extend(range(max1+1,l[i]+1))
        max1 = max(l[:i+1])
        m.pop()
    elif i==0:
```

```
        m.extend(range(1,l[i]+1))
        m.pop()
    else:
        if l[i]==m[len(m)-1]:m.pop()
        else:print "NO";exit()
print "YES"
```

29 template

```
#include<bits/stdc++.h>
#define mt make_tuple
#define mp make_pair
#define pu push_back
#define INF 1000000001
#define MOD 1000000007
#define ll long long int
#define ld long double
#define vi vector<int>
#define vll vector<long long int>
#define fi first
#define se second
#define sc(n) scanf("%d",&n);
#define scll(n) scanf("%lld",&n);
#define scld(n) scanf("%Lf",&n);
#define scr(s) {char temp[1000000];scanf("%s",temp);s = temp
    ;}
#define t1(x)             cerr<<#x<<" : "<<x<<endl
#define t2(x, y)          cerr<<#x<<" : "<<x<<" "<<#y<<" : "
    <<y<<endl
#define t3(x, y, z)       cerr<<#x<<" :" <<x<<" "<<#y<<" : "
    <<y<<" "<<#z<<" : "<<z<<endl
#define t4(a,b,c,d)       cerr<<#a<<" : "<<a<<" "<<#b<<" : "
    <<b<<" "<<#c<<" : "<<c<<" "<<#d<<" : "<<d<<endl
#define t5(a,b,c,d,e)       cerr<<#a<<" : "<<a<<" "<<#b<<" :
     "<<b<<" "<<#c<<" : "<<c<<" "<<#d<<" : "<<d<<" "<<#e<<"
     : "<<e<<endl
#define GET_MACRO(_1,_2,_3,_4,_5,NAME,...) NAME
#define t(...) GET_MACRO(__VA_ARGS__,t5, t4, t3, t2, t1)(
    __VA_ARGS__)
#define _ cout<<"here"<<endl;


using namespace std;

int main()
{

  return 0;
}
```