Team Notebook

Metropolitan University

February 17, 2024

Contents

1 Boilerplate

```
#include<bits/stdc++.h>
using namespace std;
#define pi acos(-1)
#define MOD 1000000007
#define inf 1000000010
#define endl "\n"
#define ull unsigned long long
#define con (f?"YES":"NO")
#define yes cout<<"YES"<<endl
#define no cout<<"NO"<<endl

#define dpos(n) fixed << setprecision(n)

#define clear1(a) memset(a, -1, sizeof(a))
#define clear0(a) memset(a, 0, sizeof(a))

#define sortn(a,x,n) sort(a+x, a+x+n)
#define sortv(s) sort(s.begin(), s.end())
#define reversev(s) reverse(s.begin(), s.end())
#define rsortv(s) sort(s.rbegin(),s.rend())
#define unik(a) unique(a.begin(), a.end()) - a.begin()
#define iotav(s, x) iota(s.begin(), s.end(), x)

#define lowerbound(v,x) lower_bound(v.begin(), v.end(), x)-v
     .begin()
#define upperbound(v,x) upper_bound(v.begin(), v.end(), x)-v
     .begin()

#define pb push_back
#define loj(i,j) "Case "<<i<<": "<<j
#define gap " "

#define auto(x,a) for (auto& x : a)
#define print(x) cout << #x << " = " << x << endl

long long dx[] = {1, -1, 0, 0, 1, 1, -1, -1};
long long dy[] = {0, 0, 1, -1, 1, -1, 1, -1};

int main() {
   ios_base::sync_with_stdio(false); cin.tie(0),cout.tie(0);

   #ifndef ONLINE_JUDGE
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);
   #endif

    long long t;
    cin >> t;
    while (t--) {
        long long n;
        cin>>n;

        long long a[n+2];
        for(long long i=1;i<=n;i++) cin>>a[i];

        for(long long i=1;i<=n;i++) cout<<a[i]<<" ";
    }
}
```

2 DP

2.1 Coin Changing

```
long long minNumberOfCoin(vector<long long>&v, long long val
     , long long n, vector<long long>&dp){
    if(val==0) return 0;
    if(dp[val]!=-1) return dp[val];

    long long ans = LLONG_MAX;

    for(long long i=0;i<n;i++){
        if(v[i]<=val){
            long long subAns = coinChange(v, val-v[i], n, dp)
                ;
            if(subAns!=long long_MAX && subAns+1<ans) ans =
                 subAns+1;
        }
    }

    dp[val] = ans;
    return ans;
}

int numberOfWays(int coins[], int n, int sum)
{

 int dp[sum + 1];
 memset(dp, 0, sizeof(dp));
 dp[0] = 1;
 for (int i = 0; i < n; i++)
  for (int j = coins[i]; j <= sum; j++)
   dp[j] += dp[j - coins[i]];
 return dp[sum];
}
```

2.2 Knapsack

```
long long knapSack(long long w, long long i, long long *
     marks, long long *cap, vector<vector<long long>> &dp)
{
    if (i < 0)
        return 0;
    if (dp[i][w] != -1)
        return dp[i][w];

    if (cap[i] > w) dp[i][w] = knapSack(w, i - 1, marks, cap,
         dp);
    else dp[i][w] = max(marks[i] + knapSack(w - cap[i], i -
         1, marks, cap, dp), knapSack(w, i - 1, marks, cap,
         dp));

    return dp[i][w];
}
```

3 DSU

```
struct DSU
{
 int connected;
 vector<int> par, sz;

 void init(int n)
 {
  par = sz = vector<int> (n + 1, 0);
  for(int i = 1; i <= n; i++)
   par[i] = i, sz[i] = 1;
  connected = n;
 }

 int getPar(int u)
 {
  while(u != par[u])
  {
   par[u] = par[par[u]];
   u = par[u];
  }
  return u;
 }

 int getSize(int u)
 {
  return sz[getPar(u)];
 }
 void unite(int u, int v)
 {
  int par1 = getPar(u), par2 = getPar(v);

  if(par1 == par2)
   return;

  connected--;

  if(sz[par1] > sz[par2])
   swap(par1, par2);

  sz[par2] += sz[par1];
  sz[par1] = 0;
  par[par1] = par[par2];
 }
};
```

4 Data Structure

4.1 BIT with Lazy

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

template <class T>
struct Fenwick { // 1-indexed
    int n;
    vector<T> t;

    Fenwick() {}

    Fenwick(int _n) {
        n = _n;
        t.assign(n + 1, 0);
    }

    T query(int i) {
        T ans = 0;
        for (; i >= 1; i -= (i & -i))
            ans += t[i];
        return ans;
    }

    void upd(int i, T val) {
        if (i <= 0)
            return;
        for (; i <= n; i += (i & -i))
            t[i] += val;
    }

    void upd(int l, int r, T val) {
        upd(l, val);
        upd(r + 1, -val);
    }

    T query(int l, int r) {
        return query(r) - query(l - 1);
    }
};

int main() {
    long long t;
    // cin >> t;
    t = 1;
    while (t--) {
        long long n, q;
        cin >> n >> q;
        long long a[n + 2];
        Fenwick<long long> tree(n);

        for (long long i = 1; i <= n; i++)
            cin >> a[i];

        for (long long i = 1; i <= n; i++)
            tree.upd(i, i, a[i]);

        while (q--) {
            int type;
            cin >> type;
            if (type == 1) {
                long long x, y, val;
                cin >> x >> y >> val;
                tree.upd(x, y, val);
            } else {
                long long x, y;
                cin >> x;
                cout << tree.query(x) << endl;
            }
        }
    }
    return 0;
}
```

4.2 BIT

```
void update(int i, int val, int n, int *tree)
{
    while (i <= n)
    {
        tree[i] += val;
        i += (i & -i);
    }
}
//sum from 1 to i
int getSum(int i, int *tree)
{
    int sum = 0;
    while (i > 0)
    {
        sum += tree[i];
        i ^= (i & -i);
    }
    return sum;
}
```

4.3 Segement Tree

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

void build(long long *tree, long long *a, long long node,
     long long l, long long r)
{
    if (l == r)
    {
        tree[node] = a[l];
        return;
    }

    long long left = 2 * node, right = left + 1;
    long long mid = l + (r-l)/2;

    build(tree, a, left, l, mid);
    build(tree, a, right, mid + 1, r);

    tree[node] = tree[left] + tree[right];
}

long long query(long long *tree, long long *a, long long
     node, long long l, long long r, long long begin, long
     long end)
{
    if (r < begin || end < l)
```

```
        return 0;

    if (begin <= l && r <= end)
    {
        return tree[node];
    }

    long long left = 2 * node, right = left + 1;
    long long mid = l + (r-l)/2;

    long long left_value = query(tree, a, left, l, mid, begin
        , end);
    long long right_value = query(tree, a, right, mid + 1, r,
         begin, end);

    return left_value + right_value;
}

void update(long long *tree, long long *a, long long node,
     long long l, long long r, long long index, long long
     value)
{
    if (l == r)
    {
        a[l] = value;
        tree[node] = a[l];
        return;
    }

    long long left = 2 * node, right = left + 1;
    long long mid = l + (r-l)/2;

    if (index <= mid)
        update(tree, a, left, l, mid, index, value);
    else
        update(tree, a, right, mid + 1, r, index, value);

    tree[node] = tree[left] + tree[right];
}

int main()
{
    long long n;
    cin >> n;
    long long q;
    cin >> q;

    long long a[n + 2], tree[4 * n];
    for (long long i = 1; i <= n; i++)
        cin >> a[i];

    build(tree, a, 1, 1, n);

    while (q--)
    {
        long long tt, x, y;
        cin>>tt>>x>>y;
        if(tt==1){
            update(tree, a, 1, 1, n, x, y);
            continue;
        }
        long long desire_value = query(tree, a, 1, 1, n, x, y
            );
        cout << desire_value << endl;
    }
}
```

4.4 Segment Tree with Lazy

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

void build(long long *tree, long long *lazy, long long *a,
     long long node, long long l, long long r)
{
    lazy[node] = 0;
    if (l == r)
    {
        tree[node] = a[l];
        return;
    }

    long long left = 2 * node, right = left + 1, mid = l + (r
        - l) / 2;

    build(tree, lazy, a, left, l, mid);
    build(tree, lazy, a, right, mid + 1, r);
    tree[node] = tree[left] + tree[right];
}

void propagate(long long *tree, long long *lazy, long long
     node, long long l, long long r)
{
    if (lazy[node])
    {
        tree[node] += (r - l + 1) * lazy[node];
        if (l != r)
        {
            lazy[2 * node] += lazy[node];
            lazy[2 * node + 1] += lazy[node];
        }
        lazy[node] = 0;
    }
}

long long query(long long *tree, long long *lazy, long long
     *a, long long node, long long l, long long r, long long
      b, long long e)
{
    propagate(tree, lazy, node, l, r);
    if (r < b || e < l)
        return 0;

    if (b <= l && r <= e)
        return tree[node];

    long long mid = l + (r - l) / 2;

    return query(tree, lazy, a, 2 * node, l, mid, b, e) +
         query(tree, lazy, a, 2 * node + 1, mid + 1, r, b, e)
         ;
}

void update_range(long long *tree, long long *lazy, long
     long *a, long long node, long long l, long long r, long
      long b, long long e, long long val)
{
    propagate(tree, lazy, node, l, r);
    if (r < b || e < l)
        return;
    if (b <= l && r <= e)
    {
        tree[node] += (r - l + 1) * val;

        if (l != r)
        {
            lazy[2 * node] += val;
            lazy[2 * node + 1] += val;
        }
        return;
    }

    long long mid = l + (r - l) / 2;

    update_range(tree, lazy, a, 2 * node, l, mid, b, e, val);
    update_range(tree, lazy, a, 2 * node + 1, mid + 1, r, b,
         e, val);

    tree[node] = tree[2 * node] + tree[2 * node + 1];
}

int main()
{
    long long t;
    // cin >> t;
    t=1;
    while (t--)
    {
        long long n, q;
        cin >> n >> q;
        long long a[n + 2], tree[4 * n], lazy[4 * n];

        for (long long i = 1; i <= n; i++)
            cin >> a[i];
        build(tree, lazy, a, 1, 1, n);

        while (q--)
        {
            int type;
            cin >> type;
            if (type == 1)
            {
                long long x, y, val;
                cin >> x >> y >> val;
                update_range(tree, lazy, a, 1, 1, n, x, y, val
                    );
                continue;
            }
            long long x;
            cin >> x;
            cout << query(tree, lazy, a, 1, 1, n, x, x) <<
                endl;
        }
    }
}
```

4.5 Sparse Table

```
const int N = 2e5 + 5, K = 20; // K = log2(N)
int a[N], t[N][K];
void build(int n)
{
    for (int i = 1; i <= n; i++) t[i][0] = a[i];
    for (int j = 1; j < K; j++)
    {
        for (int i = 1; i + (1 << j) - 1 <= n; i++)
        {
            t[i][j] = min(t[i][j - 1], t[i + (1 << (j - 1))][
                j - 1]);
        }
    }
}
int query(int l, int r)
{
    int k = __lg(r - l + 1);
    return min(t[l][k], t[r - (1 << k) + 1][k]);
}
```

5 Graph

5.1 DFS on Tree

```
vector<vector<int>> p;
vector<long long> depth;
vector<long long> reverse_depth;
void dfs(int u, int par)
{
    if (p[u].size() == 1 && p[u][0] == par){
        depth[u] = depth[par]+1;
        reverse_depth[u] = 1;
    }
    else
    {
        for (auto v : p[u])
        {
            if (v != par)
            {
                depth[v] = 1 + depth[u];
                dfs(v, u);
                reverse_depth[u] = 1 + reverse_depth[v];
            }
        }
    }
}
void solve(int n)
{
    p.assign(n + 2, vector<int>());
    depth.assign(n + 2, 0);
    reverse_depth.assign(n + 2, 0);
    for (int i = 1; i < n; i++)
    {
        int x, y;
        cin >> x >> y;
        p[x].push_back(y);
        p[y].push_back(x);
    }
    depth[1]=1;
    dfs(1, -1);
    int x = 2;
    cout<<depth[x]<<" "<<reverse_depth[x]<<endl;
}
```

5.2 Detect Cycle in Tree

```
void dfs(int u, vector<bool> &vis, vector<vector<int>>&p,
     int prev)
{
    if(vis[u]){
        cycle = true;
        return;
    }

    vis[u] = true;

    for (int i = 0; i < p[u].size(); i++)
    {
        int v = p[u][i];
        if(prev!=v){
            dfs(v, vis, p, u);
        }
    }
}
```

5.3 Dijkstra

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

const long long N = 2e5 + 3;
const long long inf = 1e18;

vector<pair<long long, long long>> edges[N];
vector<long long> dist(N, inf);

int main()
{
    long long n, m;
    cin >> n >> m;

    while (m--)
    {
        long long x, y, w;
        cin >> x >> y >> w;
        edges[x].push_back({y, w});
    }

    dist[1] = 0;

    priority_queue<pair<long long, long long>, vector<pair<
        long long, long long>>, greater<pair<long long, long
         long>>> pq;

    pq.push({0, 1});

    while (!pq.empty())
    {
        long long u = pq.top().second, d = pq.top().first;
        pq.pop();
        if (dist[u] < d)
            continue;
        for (auto e : edges[u])
        {
            long long w = e.second, v = e.first;
            if (dist[v] > dist[u] + w)
            {
                dist[v] = dist[u] + w;
                pq.push({dist[v], v});
            }
        }
    }

    for (long long i = 1; i <= n; i++)
        cout << dist[i] << " ";
}
```

5.4 Distance of Leaf from root

```
#include<bits/stdc++.h>
using namespace std;
//K= root, n=node
void find(vector<long long>a[], long long n, long long k)
{
  queue<long long>q;
  long long vis[n+2]={};
  long long dis[n+2]={},maxx=0ll;
  vis[k]=0;
  dis[k]=1;
  q.push(k);
  while(!q.empty())
  {
    long long x=q.front();
    q.pop();
    long long l=a[x].size();
    for(long long i=0;i<l;i++)
    {
      long long y=a[x][i];
      if(!vis[y])
      {
        q.push(y);
        vis[y]=1;
        dis[y]=dis[x]+1;
        maxx=max(maxx,dis[y]);
      }
    }
  }
  cout<<maxx<<endl;
}
```

6 Math

6.1 String and int multiply

```
string multyply(string a, int b)
{
    int carry = 0, l = a.size();

    string ans = "";

    for (int i = l - 1; i >= 0; i--)
    {
        carry = ((a[i] - '0') * b + carry);
        ans += carry % 10 + '0';
        carry /= 10;
    }
    while (carry != 0)
    {
        ans += carry % 10 + '0';
        carry /= 10;
    }
    reverse(ans.begin(), ans.end());
    return ans;
}
```

6.2 Sum of Absolute Diff of All Pairs

```
int
    sum_of_absolute_differences_of_all_pairs_in_a_given_array
    (int a[], int n)
{
    int ans = 0;
    sort(a, a + n);
    for (long long i = 0; i < n; i++)
        ans += a[i] * (2 * i - n + 1);
    return ans;
}
```

7 Number Theory

7.1 Big Mul

```
long long bigMul(long long n, long long m, long long p)
{
  if(m<=0) return 0;
    long long res = bigMul(n, m/2, p);
    long long ans = (2*res)%p;
    if(m%2) ans = (ans+n)%p;
    return ans;
}
```

7.2 Bigmod , Inverse MOD, ncr

```
#define MOD 1000000007
long long bigMod(long long a, long long b)
{
  a %= MOD;
  if (!b)
    return 1;
  long long res = bigMod(a, b / 2);
  long long ans = (res * res) % MOD;
  if (b % 2)
    ans = (ans * a) % MOD;
  return ans;
}

long long inverseMod(long long a)
{
  return bigMod(a, MOD - 2);
}


long long fact[MOD];
void factorial()
{
  fact[0] = 1;
  for (long long i = 1; i < MOD; i++)
    fact[i] = (((i % MOD) * (fact[i - 1] % MOD)) % MOD);
}

long long nCr(long long n, long long r)
{
  return ((fact[n] % MOD) * (inverseMod((fact[r] * fact[n -
      r]) % MOD) % MOD)) % MOD;
}
```

7.3 Bigmod with Loop

```
#define MOD 1000000007
long long Big(long long x, long long n)
{
  long long ans=1;
  while(n>0){
    x%=MOD;
    if(n&1) ans*=x;
    ans%=MOD;
    x*=x;
    n>>=1;
  }
  return ans;
}
```

7.4 Generate Number of Divisor 1 to N

```
vector<int>generateNumberOfDivisor(int n= 1e6){
    vector<int>divisor(n+1, 1);
    for(int i=2;i<=n;i++){
        if(divisor[i]==1){
            for(int j=i;j<=n;j+=i){
                int num = j, primeFactor = 0;
                while(num%i==0){
                    num/=i;
                    primeFactor++;
                }
                divisor[j] *= (primeFactor+1);
            }
        }
    }
    return divisor;
}
```

7.5 Get Prime

```
#define INF 1000005
int prime[INF];
bool vis[INF];

void getPrime()
{
    int k = 1;
    prime[k++] = 2;
    for (long long i = 3; i <INF; i += 2)
    {
        if (!vis[i] && i % 2)
            prime[k++] = i;
        for (long long j = i * i; j < INF; j += i)
        {
            vis[j] = true;
        }
    }
}
```

7.6 Is Prime

```
vector<bool> isPrime(long long n = 1e6)
{
    vector<bool> vis(n + 5);
    vis[1] = true;
    for (long long i = 3; i <= n; i+=2)
    {
        if (!vis[i])
            for (long long j = i * i; j <= n; j += i)
                vis[j] = true;
    }
    return vis;
}
```

7.7 MOD Jog Gun

```
#define MOD 1000000007

long long modGunKoro(long long a, long long b){
    return ((a%MOD)*(b%MOD))%MOD;
}

long long modJogKoro(long long a, long long b){
    return ((a%MOD)+(b%MOD))%MOD;
}
```

7.8 Number of Divisor

```
#define nn 1000010115.
long long int notprime[nn] = {}, prime[nn];
long long numberofDivisor(long long n)
{
    long long int c = 1, i, j, ans = 1;
    for (i = 3; i * i <= nn; i += 2)
    {

        if (!notprime[i])
        {
            for (j = i * i; j <= nn; j += i)
                notprime[j] = 1;
        }
    }
    prime[c++] = 2;
    for (i = 3; i <= nn; i += 2)
    {
        if (!notprime[i])
        {

            prime[c++] = i;
        }
    }

    for (i = 1; i <= nn && prime[i] * prime[i] <= n; i++)
    {
        if (n % prime[i] == 0)
        {
            int cnt = 1;
            while (n > 1 && n % prime[i] == 0)
            {
                n /= prime[i];
                cnt++;
            }
            ans *= cnt;
        }
    }
    if (n != 1)
        ans *= 2;
}
```

7.9 Number of Prime Divisor

```
vector<int>generateNumberOfPrimeDivisor(int n = 1e6){
    vector<int>primeDivisor(n+1, 0);
    for(int i=2;i<=n;i++){
        if(primeDivisor[i]==0){
            for(int j=i;j<=n;j+=i){
                primeDivisor[j] ++;
            }
        }
    }
    return primeDivisor;
}
```

7.10 Sum of Divisor

```
vector<int> generateSumOfDivisor(int n = 2e6)
{
    vector<int> divSum(n + 5, 1);
    divSum[1]=0;
    for (int i = 2; i <= n; i++)
    {
        for(int j=i+i;j<=n;j+=i){
            divSum[j]+=i;
        }
    }
    return divSum;
}
```

8 String

8.1 KMP

```
vector<int> lps(string pattern)
{
    int n = pattern.size();
    vector<int> v(n);
    int index = 0;

    for (int i = 1; i < n;)
    {
        if (pattern[i] == pattern[index])
        {
            v[i] = index + 1;
            i++;
            index++;
        }
        else
        {
            if (index)
                index = v[index - 1];
            else
            {
                v[i] = 0;
                i++;
            }
        }
    }
    return v;
}

int kmp(string s, string pattern)
{
    int n = s.size(), m = pattern.size();
    int i = 0, j = 0;
    int ans = 0;

    vector<int> v = lps(pattern);

    while (i < n)
    {
        if (s[i] == pattern[j])
        {
            i++;
            j++;
        }
        else
        {
            if (j)
                j = v[j - 1];
            else
                i++;
        }

        if (j == m) //to count how many pattern match
        {
            ans++;
            j = v[j - 1];
        }
    }

    return ans;
}
```

8.2 lcs

```
int lcs(string X, string Y, int m, int n, vector<vector<int
    >> &dp)
{
    if (m == 0 || n == 0)
        return 0;
    if (X[m - 1] == Y[n - 1])
        return dp[m][n] = 1 + lcs(X, Y, m - 1, n - 1, dp);

    if (dp[m][n] != -1)
    {
        return dp[m][n];
    }
    return dp[m][n] = max(lcs(X, Y, m, n - 1, dp), lcs(X, Y,
        m - 1, n, dp));
}
```

9 Trie

```
#include <bits/stdc++.h>
using namespace std;
#define endl "\n"
#define con (f ? "YES" : "NO")
#define loj(i, j) "Case " << i << ": " << j

struct Trie
{
    bool lastLetter;
    Trie *children[10];

    Trie()
    {
        for (int i = 0; i < 10; i++)
        {
            lastLetter = false;
            children[i] = nullptr;
        }
    }
};

void insert(string &s, Trie *root)
{
    int n = s.size();

    for (char c : s)
    {
        int index = c - '0';
        if (root->children[index] == nullptr)
            root->children[index] = new Trie();
        root = root->children[index];
    }
    root->lastLetter = true;
}
```

```
bool isPrefix(Trie *node)
{
    for (int i = 0; i < 10; i++)
    {
        if (node->children[i] != nullptr)
        {
            if (node->lastLetter)
                return true;
            if (isPrefix(node->children[i]))
                return true;
        }
    }
    return false;
}

void clear(Trie *node)
{
    for (int i = 0; i < 10; i++)
    {
        if (node->children[i] != nullptr)
        {
            clear(node->children[i]);
            node->children[i] = nullptr;
        }
    }
    delete (node);
}

int main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(0), cout.tie(0);

    long long t, k = 0;
    cin >> t;
    while (t--)
    {
        long long n;
        cin >> n;

        Trie *root = new Trie();

        while (n--)
        {
            string s;
            cin >> s;
            insert(s, root);
        }

        bool f = !isPrefix(root);
        cout << loj(++k, con) << endl;
        clear(root);
    }
}
```

10 Z Function

```
#include <bits/stdc++.h>
using namespace std;
#define endl  \ n
vector<int> z_function(string s)
{
    int n = (int)s.size();
    vector<int> z(n);
    z[0] = n;
    for (int i = 1, l = 0, r = 0; i < n; ++i)
    {
        if (i <= r)
            z[i] = min(r - i + 1, z[i - l]);
        while (i + z[i] < n && s[z[i]] == s[i + z[i]])
            ++z[i];
        if (i + z[i] - 1 > r)
            l = i, r = i + z[i] - 1;
    }
    return z;
}
int main()
{
    // Simple is BEAST
    int n;
    cin >> n;
    cout << n * n << endl;
    return 0;
}
```