

Рјешења задатака  
Републичко Такмичење из Информатике  
25.04.2026

## 1 Шест Седам

```
#include <iostream>

using namespace std;

int main()
{
    int t;
    cin >> t;

    while (t--)
    {

        string s1;
        cin >> s1;

        int n = s1.size(), brojacSedmica = 0;

        if (s1[0] != '6')
        {
            cout << "NE" << endl;
            continue;
        }
        int i = 0;
        for (i = 1; i < n; i++)
        {
            if (s1[i] == '6')
                brojacSedmica = 0;
            else if (s1[i] == '7')
                brojacSedmica++;
            if ((s1[i] != '6' && s1[i] != '7') || brojacSedmica > 2)
            {
                cout << "NE" << endl;
                break;
            }
        }
        if (i >= n)
        {
            cout << "DA" << endl;
        }
    }
    return 0;
}
```

## 2 Партиционисање Диска

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

int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);

    int m, n;
    cin >> m >> n;
    vector<long long> arr(m);
    for (int i = 0; i < m; i++) cin >> arr[i];

    vector<long long> pc(m - 1);
    for (int i = 0; i < m - 1; i++)
        pc[i] = arr[i] + arr[i + 1];

    sort(pc.begin(), pc.end());

    long long base = arr[0] + arr[m - 1];
    long long mn = base, mx = base;
    for (int i = 0; i < n - 1; i++) {
        mn += pc[i];
        mx += pc[m - 2 - i];
    }

    cout << mn << " " << mx << endl;
}
```

### 3 УПИТИ

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

int main()
{
    string s1, s2;
    cin >> s1 >> s2;

    int m = s1.size(), n = s2.size();

    vector<vector<int>> dp(m + 1, vector<int>(n + 1, 0));

    for (int i = 0; i <= m; i++)
        dp[i][0] = i;
    for (int j = 0; j <= n; j++)
        dp[0][j] = j;

    for (int i = 1; i <= m; i++)
    {
        for (int j = 1; j <= n; j++)
        {
            int cost = (s1[i - 1] == s2[j - 1]) ? 0 : 1;

            dp[i][j] = min({
                dp[i - 1][j] + 1,          // delete
                dp[i][j - 1] + 1,          // insert
                dp[i - 1][j - 1] + cost    // replace
            });

            // swap
            if (i > 1 && j > 1 && s1[i - 1] == s2[j - 2] && s1[i - 2] == s2[j - 1])
            {
                dp[i][j] = min(dp[i][j], dp[i - 2][j - 2] + cost);
            }
        }
    }

    cout << dp[m][n];

    return 0;
}
```

## 4 Планете

```
#include <iostream>
#include <vector>
#include <queue>
#include <limits>
using namespace std;

typedef vector<int> vi;
typedef vector<vector<int>> vvi;

const int INF = numeric_limits<int>::max();

bool can(int st, int n, const vvi& adj, const vi& t) {
    queue<int> q;
    vi dist(n, -1);

    q.push(0);
    dist[0] = 0;

    while(!q.empty()) {
        int u = q.front();
        q.pop();

        if(u == n-1) {
            return true;
        }

        for(int v : adj[u]) {
            if(dist[v] == -1 && dist[u]+1 < t[v] - st) {
                dist[v] = dist[u] + 1;
                q.push(v);
            }
        }
    }

    return false;
}

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

    vi t(n);
    for(int i=0; i<n-1; i++) {
        cin >> t[i];
    }
    t[n-1] = INF;

    vvi adj(n);
    for(int i=0; i<m; i++) {
        int u, v;
        cin >> u >> v;
        u--, v--;
        adj[u].push_back(v);
        adj[v].push_back(u);
    }
}
```

```

int lo = 0, hi = 1;

while(can(hi, n, adj, t)){
    hi <= 1;
}

int sol = -1;
while(lo <= hi) {
    int mid = lo + (hi - lo) / 2;
    if(can(mid, n, adj, t)) {
        sol = mid;
        lo = mid + 1;
    } else {
        hi = mid - 1;
    }
}

cout << sol << '\n';

return 0;
}

```

## 5 Анселот

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

const int MAX_N = 200005;
const long long INF = 2e18;

int N, Q;
vector<int> e(MAX_N);
vector<long long> segtreeLeft(4 * MAX_N);
vector<long long> segtreeRight(4 * MAX_N);

void update(
    vector<long long>& segtree,
    int treeCurr,
    int treeLeft, int treeRight,
    int pos, long long newValue
) {
    if (treeLeft == treeRight) {
        segtree[treeCurr] = newValue;
    }
    else {
        int treeMid = (treeLeft + treeRight) / 2;

        if (pos <= treeMid)
            update(
                segtree,
                2 * treeCurr,
                treeLeft, treeMid,
                pos, newValue
            );
        else
            update(
                segtree,
                2 * treeCurr + 1,
                treeMid + 1, treeRight,
                pos, newValue
            );

        segtree[treeCurr] = min(
            segtree[2 * treeCurr],
            segtree[2 * treeCurr + 1]
        );
    }
}

long long query(
    vector<long long>& segtree,
    int treeCurr,
    int treeLeft, int treeRight,
    int queryLeft, int queryRight
) {
    if (treeRight < queryLeft || queryRight < treeLeft) {
        return INF;
    }
    if (queryLeft <= treeLeft && treeRight <= queryRight) {
        return segtree[treeCurr];
    }
}
```

```

    int treeMid = (treeLeft + treeRight) / 2;

    long long leftAns = query(
        segtree,
        2 * treeCurr,
        treeLeft, treeMid,
        queryLeft, queryRight
    );
    long long rightAns = query(
        segtree,
        2 * treeCurr + 1,
        treeMid + 1, treeRight,
        queryLeft, queryRight
    );

    return min(leftAns, rightAns);
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(NULL);

    cin >> N >> Q;

    for (int i = 1; i <= N; i++) {
        cin >> e[i];

        update(segtreeLeft, 1, 1, N, i, (long long)e[i] - i);
        update(segtreeRight, 1, 1, N, i, (long long)e[i] + i);
    }

    while (Q--) {
        int t;
        cin >> t;

        if (t == 1) {
            int k, x;
            cin >> k >> x;

            e[k] = x;

            update(segtreeLeft, 1, 1, N, k, (long long)x - k);
            update(segtreeRight, 1, 1, N, k, (long long)x + k);
        }
        else {
            int k;
            cin >> k;

            long long leftAns = query(segtreeLeft, 1, 1, N, 1, k) + k;
            long long rightAns = query(segtreeRight, 1, 1, N, k, N) - k;

            cout << min(leftAns, rightAns) << "\n";
        }
    }

    return 0;
}

```