Sliding Window

Count maximum length substring contains no more than k’ 2

#include<bits/stdc++.h>

using namespace std;

int main() {

  int n, k;

  cin >> n >> k;

  string s;

  cin >> s;

  int max\_len = 0, count\_of\_2 = 0, left = 0, right = 0;

  while(right < n) {

    if(s[right] == '2') count\_of\_2++;

    while(count\_of\_2 > k) {

      if(s[left] == '2') count\_of\_2--;

      left++;

    }

    max\_len = max(max\_len, right - left + 1);

    right++;

  }

  cout << max\_len << endl;

  return 0;

}

Binary Seacrh

Find smallest integer >= x

#include<bits/stdc++.h>

using namespace std;

int main() {

    int n, m;

    cin >> n >> m;

    vector<long long> a(n), q(m), prefsum(n);

    for(int i = 0; i < n; i++) {

        cin >> a[i];

        if(i == 0) prefsum[0] = a[0];

        else prefsum[i] = prefsum[i-1] + a[i];

    }

    for(int i = 0; i < m; i++) {

        cin >> q[i];

        if(q[i] > prefsum[n-1]) {

          cout << -1 << endl;

          continue;

    }

    if(q[i] == 0) {

      cout << 0 << endl;

      continue;

    }

        int ans = -1;

        int left = 0, right = n - 1;

        while(left <= right) {

            int mid = left + (right - left) / 2;

            if(prefsum[mid] >= q[i]) {

                ans = mid;

                right = mid-1;

            } else {

                left = mid+1;

            }

        }

        if(ans == -1) cout << -1 << endl;

        else cout << ans + 1 << endl;

    }

    return 0;

}

Multi-sources BFS

#include<bits/stdc++.h>

using namespace std;

const long long INF = 1e18;

void multi\_bfs(vector<int> &sources, vector<vector<int>> &v, vector<long long> &dist) {

  queue<int> q;

  for(int source : sources) {

    dist[source] = 0;

    q.push(source);

  }

  while(!q.empty()) {

    int node = q.front();

    q.pop();

    for(int neighbor : v[node]) {

      if(dist[neighbor] == INF) {

        dist[neighbor] = dist[node] + 1;

        q.push(neighbor);

      }

    }

  }

}

int main() {

  int n, m, s, k, g;

  cin >> n >> m >> s >> k >> g;

  int x, y;

  vector<vector<int>> v(n+1);

  for(int i = 0; i < m; i++) {

    cin >> x >> y;

    v[x].push\_back(y);

    v[y].push\_back(x);

  }

  vector<int> e(k), r(g);

  for(int i = 0; i < k; i++) cin >> e[i];

  for(int i = 0; i < g; i++) cin >> r[i];

  vector<long long> dist\_h(n+1, INF), dist\_g(n+1, INF);

  vector<int> h\_source = {s};

  multi\_bfs(h\_source, v, dist\_h);

  multi\_bfs(r, v, dist\_g);

  int ans = 0;

  for(int i = 0; i < k; i ++) {

    if(dist\_h[e[i]] < dist\_g[e[i]]) ans++;

  }

  cout << ans << endl;

  return 0;

}

Sieve of Eratosthenes

Count how many integers <= n are almost prime (have exactly 2 primes factor)

#include<bits/stdc++.h>

using namespace std;

int main() {

int n;

cin >> n;

vector<int> prime;

bool isPrime[n+1];

memset(isPrime, true, sizeof(isPrime));

isPrime[0] = false;

isPrime[1] = false;

for(int p = 2; p <= n; p++) {

if(isPrime[p]) {

prime.push\_back(p);

for(int m = 2; m\*p <= n; m++) {

isPrime[m\*p] = false;

}

}

}

int ans = 0;

for(int k = 6; k <= n; k++) {

int div = 0;

for(int i = 0; i < prime.size(); i++) {

if(k % prime[i] == 0) div++;

if(div > 2) break;

}

if(div == 2) ans++;

}

cout << ans << endl;

return 0;

}

DP on DFS (tree)

#include<bits/stdc++.h>

using namespace std;

struct child {

int left;

int right;

};

const int INF = 1e9;

int dfs(int node, int dp[], vector<child>& v, const string& s) {

if(dp[node] != INF) return dp[node];

if(v[node].left != 0) {

if(s[node-1] == 'L') dp[node] = dfs(v[node].left, dp, v, s);

else dp[node] = dfs(v[node].left, dp, v, s) + 1;

}

if(v[node].right != 0) {

if(s[node-1] == 'R') dp[node] = min(dp[node], dfs(v[node].right, dp, v, s));

else dp[node] = min(dp[node], dfs(v[node].right, dp, v, s) + 1);

}

return dp[node];

}

int main() {

int t;

cin >> t;

while(t--) {

int n;

cin >> n;

string s;

cin >> s;

vector<child> v(n+1);

int dp[n+1];

fill(dp, dp + n+1, INF);

int x, y;

for(int i = 1; i <= n; i++) {

cin >> x >> y;

v[i] = {x, y};

if(x == 0 && y == 0) dp[i] = 0;

}

cout << dfs(1, dp, v, s) << endl;

}

return 0;

}