Fallen Stop Blowing my Mind
UFPB

Maximus__, danilocb21 and GabrielCampelo

Thursday 19th February, 2026

Contents

1 Data Structures

1.1 BIT

```
// build - O(n)
// update - O(log(n))
// query - O(log(n))

714 struct BIT {
406     vector<ll> bit;
1a8     int n;

32c     BIT(int n) {
b75         this->n = n;
a72         bit.assign(n, 0);
3a4     }

14d     BIT(vector<ll> const &a) : BIT(a.size()) {
eb6         for (int i = 0; i < a.size(); ++i) {
a76             bit[i] += a[i];
136             int r = i | (i + 1);
1f8             if (r < n) bit[r] += bit[i];
cc4         }
c72     }

ad7     ll sum(int r) {
b73         ll ret = 0;
817         for (; r >= 0; r = (r & (r + 1)) - 1)
b6b             ret += bit[r];
edf         return ret;
ad0     }

77a     ll sum(int l, int r) {
67b         return sum(r) - sum(l - 1);
b26     }

f52     void add(int idx, ll delta) {
718         for (; idx < n; idx = idx | (idx + 1))
7ed             bit[idx] += delta;
ac0     }
53d };
```

1.2 Difference Arrays

```
// https://codeforces.com/blog/entry/78762

e8d int main() {
a85     int n = 5; // Size of array
13b     vector<int> elements{0, 1, 1, 1, 1, 1}; // 1 based indexing
        // n+2 because we need are not using the 0-th index and we
   need one more element in the array.
e56     vector<int> diff(n + 2, 0);

348     int updateValue = 10;
3c8     int l = 2, r = 5;
e8d     diff[l] += updateValue;
ae0     diff[r + 1] -= updateValue;

78a     for (int i = 1; i <= n; i++) {
fb6         diff[i] += diff[i - 1];
093         elements[i] += diff[i];
717     }
014     for (int i = 1; i <= n; i++) cout << elements[i] << " ";
a3c }
```

1.3 Disjoint Set Union

```
// Estrutura que permite combinar conjuntos e
// dizer de qual conjunto cada elemento faz parte

// Operacoes:
// make_set(v) -> cria um novo conjunto com o elemento v
// union_sets(a, b) -> combina os conjuntos do qual os elementos a e b
    fazem parte
// find_set(v) -> retorna o elemento que representa o conjunto do qual
    v faz parte

// Cada operacao e aproximadamente O(1)

d56 struct DSU {
7b5     vector<int> parent, rank;

f9a     DSU(int n) {
62b         parent.resize(n);
9f0         rank.resize(n, 0);
603         for (int i = 0; i < n; i++) {
236             parent[i] = i;
2c0         }
b4c     }

369     void make_set(int v) {
2b9         parent[v] = v;
a83         rank[v] = 0;
760     }

94f     int find_set(int v) {
1ff         if (v == parent[v])
6dc             return v;
daf         return parent[v] = find_set(parent[v]); // Path compression
3bc     }

674     void union_sets(int a, int b) {
0ae         a = find_set(a);
d6a         b = find_set(b);
1d3         if (a != b) {
e81             if (rank[a] < rank[b])
257                 swap(a, b);
263             parent[b] = a;
21f             if (rank[a] == rank[b])
bd6                 rank[a]++;
9f8         }
8e7     }
4fd };
```

1.4 Kadane

```
// Calcula o subarray com maior soma em O(n)

a5c int kadane(vector<int> &vec) {
0b3     int mx = INT_MIN;
6f5     int curr = 0;
bdb     for (int x : vec) {
ad4         curr += x;
e95         mx = max(mx, curr);
f67         curr = max(curr, 0);
9d6     }
55e     return curr;
b0d }
```

1.5 Prefix 2d

```
e8d int main(){

809     prefix[i][j] = array[i][j] + prefix[i-1][j] + prefix[i][j-1] -
    prefix[i-1][j-1];

        // Achando o valor de alguma celula:
397     valor = prefix[y2][x2] - prefix[y2][x1-1] - prefix[y1-1][x2] +
    prefix[y1-1][x1-1];
bc1 }
```

1.6 SegTree

```
// build : O(n)
// update : O(logn)
// query : O(logn)

3c9 struct node {
97f     int val;

5e1     node() {
aa1         val = 0;
            // val = elemento neutro
a06     }

a25     node(int val) : val(val) {
```

```
a6f     }

40c     node operator + (const node &rhs) const {
6d8         return node(val + rhs.val);
            // return node(val op rhs.val);
ad2     }
772 };

383 struct SegTree {
1a8     int n;
093     vector<node> st;

bd8     SegTree(){}
dd4     SegTree(int n) : n(n) {
502         st.resize(4 * n + 2);
2d1     }
d75     SegTree(vector<int> &a) {
9dc         n = a.size();
502         st.resize(4 * n + 2);
77f         build(1, 0, n - 1, a);
fa6     }

a6c     void build(int pos, int l, int r, vector<int> &a) {
893         if(l == r) {
76e             st[pos] = node(a[l]);
505             return;
b03         }
f7e         int mi = (l + r) / 2;
b01         build(2 * pos, l, mi, a);
e9b         build(2 * pos + 1, mi + 1, r, a);
4f4         st[pos] = st[2 * pos] + st[2 * pos + 1];
3eb     }

ea5     void update(int x, int y, int pos, int l, int r) { //void
    update(int x, node y, int pos, int l, int r)
893         if(l == r) {
90b             st[pos] = node(y); //st[pos] = y;
505             return;
cdd         }
f7e         int mi = (l + r) / 2;
a9a         if(x <= mi) update(x, y, 2 * pos, l, mi);
40b         else update(x, y, 2 * pos + 1, mi + 1, r);
4f4         st[pos] = st[2 * pos] + st[2 * pos + 1];
bc4     }
105     void update(int x, int y) { // void update(int x, node y)
051         update(x, y, 1, 0, n - 1);
2ec     }

052     node query(int x, int y, int pos, int l, int r) {
fe9         if(y < l || r < x) return node();
c0e         if(x <= l && r <= y) return st[pos];
f7e         int mi = (l + r) / 2;
5f9         return query(x, y, 2 * pos, l, mi) + query(x, y, 2 * pos +
    1, mi + 1, r);
6e5     }
9a5     node query(int x, int y) {
0c9         return query(x, y, 1, 0, n - 1);
cb8     }
b73 };
```

1.7 Subarrays

```
// Subarray Sums II https://cses.fi/problemset/task/1661/

// Calculando a quantidade de subarrays validos.
// Se a soma de um subarray e igual a k

e8d int main(){
0e8     int n, k; cin >> n >> k;

aa8     vector<int> arr(n);
9e5     for (auto &i : arr) cin >> i;

04b     ll ans = 0;
271     ll prefix_sum = 0;
419     map<ll, int> rastreio;

90f     rastreio[0]++;
a19     for (int x : arr){
b33         prefix_sum += x;

71e         ans += rastreio[prefix_sum - k];

2d9         rastreio[prefix_sum]++;
b47     }

f49     cout << ans << "\n";
7e7 }


// Subarray Divisibility: https://cses.fi/problemset/view/1662/
// Contando a quantidade de subarrays que a soma e divisivel por n.
```

```
e8d int main(){

9ee     int n; cin >> n;

788     vector<ll> arr(n);
5cf     arr[0]++;
fe2     ll prefix = 0;
78a     for (int i = 1; i <= n; i++){
40a         ll a; cin >> a;

29e         prefix += a;
bc3         arr[((prefix % n) + n) % n]++;
cae     }

04b     ll ans = 0;
c06     for (auto x : arr){
507         ans += ((x-1)*(x))/2;
a54     }

f49     cout << ans << "\n";
b9e }
```

2 Dynamic Programming

2.1 Edit Distance DP

```
// Encontrar o valor minimo de operacoes
// para tornar uma string igual a outra.

// Operacoes:
// Adicionar um elemento na string
// Remover um elemento da string
// Modificar um elemento da string

// O(|S| * |T|)

2b7 #include <bits/stdc++.h>
ca4 using namespace std;

ae8 constexpr int N = 2010;
991 int dp[N][N];

e8d int main() {
ac1     string s,t;
fee     cin>>s>>t;

5ce     int n = s.size(), m = t.size();

4e3     for (int i = 0; i <= n; i++)
1c2         dp[i][0] = i;
cf2     for (int i = 0; i <= m; i++)
309         dp[0][i] = i;

78a     for (int i = 1; i <= n; i++) {
cbc         for (int j = 1; j <= m; j++) {
842             if (s[i-1] == t[j-1])
aaf                 dp[i][j] = dp[i-1][j-1];
295             else
a5f                 dp[i][j] = 1 + min({dp[i-1][j-1], dp[i][j-1],
    dp[i-1][j]});
649         }
20c     }

0ed     cout << dp[n][m] << '\n';
261 }
```

2.2 Longest Common Subsequence

```
// Encontrar a maior subsequencia de duas strings

// O(|S| * |T|)

2b7 #include <bits/stdc++.h>
ca4 using namespace std;

8fe constexpr int N = 3010;
991 int dp[N][N];

e8d int main() {
ac1     string s,t;
fee     cin>>s>>t;

5ce     int n = s.size(), m = t.size();

78a     for (int i = 1; i <= n; i++) {
cbc         for (int j = 1; j <= m; j++) {
842             if (s[i-1] == t[j-1])
008                 dp[i][j] = dp[i-1][j-1] + 1;
295             else
398                 dp[i][j] = max(dp[i-1][j], dp[i][j-1]);
```

```
520         }
5b3     }

        // Tamanho da maior subsequencia
0ed     cout << dp[n][m] << '\n';

208     string lcs;
df2     int i = n, j = m;
811     while (i > 0 && j > 0) {
6d5         if (s[i-1] == t[j-1]) {
450             lcs.push_back(s[i-1]);
5f9             i--, j--;
b52         }
c64         else if (dp[i-1][j] > dp[i][j-1]) i--;
4b7         else j--;
dca     }
1be     reverse(lcs.begin(), lcs.end());

641     cout << lcs << '\n';
1c2 }
```

2.3 Longest Increasing Subsequence

```
// Maior subsequencia crescente de um vetor de numeros
// O(n log(n))

2b7 #include <bits/stdc++.h>
ca4 using namespace std;

e1f #define INF 0x3f3f3f3f

e8d int main() {
1a8     int n;
a68     cin>>n;
70a     vector<int> v(n);
830     for (int i = 0; i < n; i++)
44f         cin>>v[i];

0ff     vector<int> lis;
603     for (int i = 0; i < n; i++) {
64d         auto it = lower_bound(lis.begin(), lis.end(), v[i]);
b05         if (it == lis.end()) lis.push_back(v[i]);
4e6         else {
4c7             int k = it-lis.begin();
c04             lis[k] = v[i];
a83         }
a3a     }

        // Tamanho da maior subsequencia
3ff     cout << lis.size() << '\n';
7c1 }
```

2.4 Mochila

```
7df int solve(int n, int C, vector<pair<int, int>> &v) {
11e     int res = 0;
0f7     for(int mask = 1, l = 1 << n; mask < l; mask++) {
0bf         int W = 0, V = 0;
04b         for(int i = 0, p = 1; i < n; i++, p <<= 1) {
b81             if(mask & p) {
c90                 W += v[i].first;
1ec                 V += v[i].second;
8a9             }
c0e         }
806         if(W <= C) {
ff0             res = max(res, V);
f87         }
12c     }
b50     return res;
5e3 }

e8d int main() {
4ad     int n, C;
6b9     cin >> n >> C;
6f7     vector<pair<int, int>> v(n); // (w, v)
385     for(int i = 0; i < n; i++) cin >> v[i].first >> v[i].second;
044     cout << solve(n, C, v) << endl;
bb3     return 0;
ec5 }
```

2.5 Subset Sum

```
// Time Complexity: O(N*K)
// Space Complexity: O(K/32)

2b7 #include <bits/stdc++.h>
ca4 using namespace std;

dca int n,k;
```

```
990 vector<int> v;

e8d int main() {
0a1     cin>>n>>k;
e2b     v.resize(n);

830     for (int i = 0; i < n; i++)
44f         cin>>v[i];

004     vector<bool> dp(k+1);
832     dp[0] = true;  // Base case: we can always form sum 0 with an
    empty set

603     for (int i = 0; i < n; i++) {
987         for (int j = k; j >= v[i]; j--) {
6d6             dp[j] = dp[j] || dp[j - v[i]];
193         }
140     }

f9c     cout << dp[k] << '\n';
5f8 }
```

3 Graph Theory

3.1 Bellman-Ford

```
// Encontra o menor caminho de um ponto a outro de um grafo
// que pode conter arestas negativas.

// V - Numero de vertices
// E - Numero de arestas
// O(V*E)

e1f #define INF 0x3f3f3f3f

e9b struct Edge {
6a7     int src, dest, weight;
818 };

520 int V, E;
5d9 vector<Edge> edges(E);
e3a vector<int> dist(V, INF);

fd1 int bellmanFord(int src, int dest) {
e13     dist[src] = 0;
079     vector<int> prnt(V, -1);

        // Relaxa os vertices |V-1| vezes para garantir a menor
    distancia.
f0f     for (int i = 0; i < V - 1; i++) {
c9b         for (const auto& [u, v, wei] : edges) {
34a             if (dist[u] != INF && dist[u] + wei < dist[v]) {
842                 dist[v] = dist[u] + wei;
81b                 prnt[v] = u;
572             }
125         }
fab     }

ac9     return dist[dest];
e9c }

// Ve se existe um ciclo no grafo.
// Retorna um vetor vazio se nao houver ciclo negativo
// ou um vetor com os vertices do ciclo caso exista
2c4 vector<int> findNegativeCycle() {
079     vector<int> prnt(V, -1);  // Para rastrear o predecessor de
    cada vertice
50a     dist[0] = 0;  // Pode comecar de qualquer ponto (nesse caso 0).

f0f     for (int i = 0; i < V - 1; i++) {
c9b         for (const auto& [u, v, wei] : edges) {
34a             if (dist[u] != INF && dist[u] + wei < dist[v]) {
842                 dist[v] = dist[u] + wei;
81b                 prnt[v] = u;
572             }
125         }
fab     }

        // Depois de relaxar |V-1| vezes, tentar relaxar mais
        // uma vez para encontrar o ciclo.
2c9     int cycleVertex = -1;
c9b     for (const auto& [u, v, wei] : edges) {
34a         if (dist[u] != INF && dist[u] + wei < dist[v]) {
8dd             cycleVertex = v;
c2b             break;
a8e         }
efe     }

fd6     if (cycleVertex == -1)
5fa         return {-1};  // Nao ha ciclo negativo
```

```
        // Para garantir que chegamos em um vertice do ciclo, andamos
    V passos
c5e     for (int i = 0; i < V; i++)
a91         cycleVertex = prnt[cycleVertex];

411     vector<int> cycle;
59b     for (int u = cycleVertex;; u = prnt[u]) {
e91         cycle.push_back(u);
6ab         if (u == cycleVertex && cycle.size() > 1)
c2b             break;
f45     }
563     reverse(cycle.begin(), cycle.end());

714     return cycle;
5e6 }
```

3.2 BFS

```
// Encontra o menor caminho de um ponto a outro.
// Parecido com o Dijkstra porem mais eficiente
// ja que cada aresta so tem peso 0 ou 1.

// O(n)

e1f #define INF 0x3f3f3f3f

63c vector<vector<pair<int,int>>> adj;

48d int bfs_01(int n, int s) {
ec2     vector<int> dist(n, INF);
a93     dist[s] = 0;

871     deque<int> q;
e87     q.push_front(s);

14d     while (!q.empty()) {
e4a         int u = q.front();
ced         q.pop_front();
0c8         for (const auto& [v,w] : adj[u]) {
dde             if (dist[u] + w < dist[v]) {
491                 dist[v] = dist[u] + w;
735                 if (w == 1)
c68                     q.push_back(v);
295                 else
480                     q.push_front(v);
32e             }
68d         }
f48     }

649     return dist[n-1];
8d0 }
```

3.3 Bipartite Graph Check

```
999 bool isBipartite(int n, vector<vector<int>>& adj) {
731     vector<int> side(n, -1);
596     bool is_bipartite = true;
26a     queue<int> q;
3a1     for (int st = 0; st < n; ++st) {
ce1         if (side[st] == -1) {
ea3             q.push(st);
8d5             side[st] = 0;
14d             while (!q.empty()) {
b1e                 int v = q.front();
833                 q.pop();
f74                 for (int u : adj[v]) {
55e                     if (side[u] == -1) {
003                         side[u] = side[v] ^ 1;
f73                         q.push(u);
f99                     } else {
e50                         is_bipartite &= side[u] != side[v];
e75                     }
3e0                 }
919             }
58a         }
5f1     }

024     return is_bipartite;
d3c }
```

3.4 Dijkstra

```
// Encontra o menor caminho do vertice de index s ate os outros
    vertices
//
// O(n log(n))

431 const int INF = 0x3f3f3f3f;
```

```
63c vector<vector<pair<int, int>>> adj; // {to, weight}

7bd int dijkstra(int n, int s) {
ec2     vector<int> dist(n, INF);

        // origem
a93     dist[s] = 0;

5d9     using pi = pair<int,int>;
74c     priority_queue<pi, vector<pi>, greater<pi>> q;
115     q.emplace(0,s);
14d     while (!q.empty()) {
31e         auto [w,u] = q.top();
833         q.pop();

ec2         if (u == n-1) break;
976         if (w != dist[u]) continue;

539         for (auto [W,v] : adj[u]) {
f36             if (w+W < dist[v]) {
655                 dist[v] = w+W;
990                 q.emplace(w+W,v);
039             }
8d0         }
a4d     }

649     return dist[n-1];
6b1 }
```

3.5 Floyd-Warshall

```
// encontra o menor caminho entre todo par de vertices
// returna 1 se ha ciclo negativo
//
// dist[i][i] = 0
// para i != j
//    d[i][j] = peso , se ha aresta
//    dist[i][j] = INF, c.c.
//
// O(n^3)

77e const long long LINF = 0x3f3f3f3f3f3f3f3fll;
1a8 int n;
2a4 long long dist[n][n];

b87 bool floydWarshal() {
9ba     for (int k = 0; k < n; k++) {
603         for (int i = 0; i < n; i++) {
578             for (int j = 0; j < n; j++) {
4c4                 dist[i][j] = min(dist[i][j], dist[i][k] +
    dist[k][j]);
1c1             }
37a         }
4cd     }

320     for (int i = 0; i < n; i++) if (dist[i][i] < 0) return 1;

bb3     return 0;
093 }
```

3.6 Kruskal

```
// Gera e retorna uma AGM de um grafo G
// Para a arvore geradora maxima basta que peso = -peso
//
// V = {0, 1, 3, ..., N - 1}
// O (MlogM + N^2) : M = |E|, N = |V|

e9b struct Edge {
58e     int u, v, weight;
0a1     bool operator<(Edge const& other) {
d96         return weight < other.weight;
308     }
973 };

b24 vector<Edge> kruskal(vector<Edge> *edges, int n) {
704     int cost = 0;
0a4     vector<Edge> msp;
a5e     vector<int> tree_id(n);
9e5     for (int i = 0; i < n; i++) tree_id[i] = i;

ff1     sort(edges.begin(), edges.end());

508     for (Edge e : edges) {
616         if (tree_id[e.u] != tree_id[e.v]) {
89c             cost += e.weight;
623             msp.push_back(e);

                // unite
016             int old_id = tree_id[e.u], new_id = tree_id[e.v];
603             for (int i = 0; i < n; i++) {
7c2                 if (tree_id[i] == old_id) tree_id[i] = new_id;
```

```
a2a             }
f55         }
b42     }
e67 }
```

3.7 Quantidade de Ciclos

```
// https://atcoder.jp/contests/abc399/tasks/abc399_c

539 void dfs(int v, int parent){
847     if (!vis[v]) vis[v] = true;

fa4     for (auto w : g[v]){
e24         if (!vis[w]){
3f5             dfs(w, v);
b40         }
b75         else if (w != parent) ans++;
b97     }
1c2 }

e8d int main(){
603     for (int i = 0; i < n; i++){
420         if (!vis[i]) {
5fe             dfs(i, -1);
92f         }
f4c     }

ea0     cout << ans/2 << "\n";
69a }
```

4 Math

4.1 Algoritmo de Euclides

```
// Time Complexity: O(log min(a,b))
// Auxiliary Space: O(log min(a,b))

ba6 int gcd(int a, int b){
650     if(b == 0) return a;
6c4     else return cd(b, a % b);
1a8 }
```

```
ebb int lcm(int a, int b){
        // return a * b / __gcd(a, b) could be overflow
c27     return a / __gcd(a, b) * b;
312 }
```

4.2 Big int

```
// https://vjudge.net/problem/UVA-10106 - 10^250

e8d int main() {

35c     string x, y;
6b5     cin >> x >> y;

8fc     int n = x.size(), m = y.size();
977     vector<int> a(n), b(m), p(n + m, 0);

        // invertendo para facilitar a logica
269     for (int i = 0; i < n; i++) a[i] = x[n - 1 - i] - '0';
18d     for (int j = 0; j < m; j++) b[j] = y[m - 1 - j] - '0';

        // Multiplicacao O(n*m)
603     for (int i = 0; i < n; i++) {
36d         int carry = 0;
891         for (int j = 0; j < m; j++) {
65f             int idx = i + j;
a83             int prod = a[i] * b[j] + p[idx] + carry;
4d1             p[idx] = prod % 10;
cce             carry = prod / 10;
bc0         }
            // adiciona o carry
dfa         p[i + m] += carry;
9f6     }

        // Remove zeros a esquerda (do fim do vetor invertido)
2ec     int k = n + m - 1;
f7d     while (k > 0 && p[k] == 0) --k;

        // Imprime resultado em ordem correta
6e0     for (int i = k; i >= 0; i--) {
61f         cout << p[i];
ec7     }
199     cout << "\n";
05d }
```

4.3 Crivo de Erastotenes

```
// Time Complexity: O(nloglogn)
// Auxiliary Space: O(n)
// Find primes in range [2, n]

705 vector<int> sieve(int n){
e6e     vector<int> is_prime(n + 1, true);
19e     is_prime[0] = is_prime[1] = false;

bc4     for(int i = 2; (long long)i * i <= n; i++){
4a3         if(is_prime[i]){
985             for(int j = i * i; j <= n; j += i){
db3                 is_prime[j] = false;
f80             }
b3f         }
26e     }
054     return is_prime;
0c7 }
```

4.4 Euclides Extendido

```
// Teorema de Bezout
// Time Complexity: O(log N)
// Auxiliary Space: O(log N)

// ax + by = gcd(a, b)
// gcd(a, b) = gcd(b % a, a) = (b % a) * x1 + a * y1
// ax + by = (b - (b/a) * a) * x1 + a * y1
// ax + by = a(y1 - (b/a) * x1) + b * x1
// x = y1 - (b/a) * x1
// y = x1

e4b int gcdExtended(int a, int b, int *x, int *y) {
220     if(a == 0){
b9d         *x = 0;
288         *y = 1;
73f         return b;
420     }

608     int x1, y1; // To store results of recursive call
c2d     int gcd = gcdExtended(b%a, a, &x1, &y1);

        // Update x and y using results of
        // recursive call
98c     *x = y1 - (b/a) * x1;
9bf     *y = x1;

e06     return gcd;
059 }
```

4.5 Euler's totient function

```
// Time Complexity: O(sqrt(n))
3fc int phi(int n){
efa     int result = n;
2ed     for(int i = 2; i * i <= n; i++){
c06         if(n % i == 0){
b52             int count = 0;
4cd             while(n % i == 0){
135                 n /= i;
157             }
21c             result -= result / i;
850         }
741     }
726     if(n > 1) result -= result / n;
dc8     return result;
8e9 }

// Euler's totient function 1 to n in O(nlog(log(n)))
// use the same ideas as the Sieve of Eratosthenes
429 vector<int> phi_1_to_n(int n){
675     vector<int> vec(n + 1);
4e3     for(int i = 0; i <= n; i++)
716         vec[i] = i;
508     for(int i = 2; i <= n; i++){
10b         if(vec[i] == i){
c6c             for(int j = i; j <= n; j += i){
816                 vec[j] -= vec[j] / i;
502             }
196         }
352     }
9d8     return vec;
eea }
```

4.6 Exponenciacao Rapida

```
// result = a^b % m
// Time Complexity = O(log b)
```

```
d95 ll binpow(ll a, ll b, ll m){
df2     ll result = 1;
63a     while(b > 0){
8e2         if(b & 1) result = result * a % m;
537         a = a * a % m;
1b4         b >>= 1;
68b     }
dc8     return result;
3c6 }

// a^b^c -> Pequeno teorema de Fermat | M = 1e9 + 7
// binpow(a, binpow(b, c, m-1), m)
```

4.7 Inversa Modular

```
// The exact time complexity of the this recursion is not known.
// It's is somewhere between  O(logm / loglogm) and O(m^(1/3 -2/177 +
    e))
// demo:
// m prime and a,r < m -> exist a_inv and r_inv
// m = k*a + r
// 0   k*a + r (mod m)
// -k*a   r (mod m)
// -k   r*a_inv (mod m)
// a_inv   k*r_inv (mod m)

a18 int inv(int a, int m){
033     return a <= 1 ? a : m - (long long)(m / a) * inv(m % a, m) % m;
5fc }

// Binary Exponentation method
// O(log m)
// if a and m are relatively prime and m is prime
// power(a, m - 2)   a_inv (mod m)
951 long long binpow(long long a, long long b){
3fe     long long result = 1;
63a     while(b > 0){
427         if(b & 1) result *= a;
70c         a *= a;
1b4         b >>= 1;
aa4     }
dc8     return result;
4ad }
```

```
// precompute the inverse for every number in the range [1, m- 1] in
    O(m)
e8d int main(){
7f4     int m = 1000000007;
641     int invArray[m];
7d1     invArray[1] = 1;
92c     for(int a = 2; a < m; a++){
b75         invArray[a] = m - (long long)(m / a) * invArray[m % a] % m;
463     }
c20 }
```

4.8 Método de Horner para Avaliação Polinomial

```
// f(x) = (Cn * x^n) + (Cn-1 * x^n-1) + (Cn-2 * x^n-2) + ... + (C1 *
    x) + C0
/* Ex:
ecd  f(x) = 2x3 - 6x2 + 2x - 1
649  poly = {2, -6, 2, -1}
4f0  x = 3 -> f(3) = 5
c4c */

//Time Complexity: O(n)
//Auxiliary Space: O(1)

628 int horner(vector<int> &poly, int x){
855     int result = poly[0];
bc6     int n = poly.size();

6f5     for(int i = 1; i < n; i++){
a32         result = result * x + poly[i];
27a     }

dc8     return result;
f2a }
```

5 Miscellaneous

5.1 Distancias

```
// Localizacao de um ponto em relacao a 2 duas retas
e8d int main() {
```

```
271     ll x1, y1, x2, y2, x3, y3;
3fc     cin >> x1 >> y1 >> x2 >> y2 >> x3 >> y3;

        // Vetor p1 p2 = (x2-x1, y2-y1)
        // Vetor p1 p3 = (x3-x1, y3-y1)
        // Cross = (x2-x1)*(y3-y1) - (y2-y1)*(x3-x1)
96f     ll cross = (x2 - x1) * (y3 - y1) - (y2 - y1) * (x3 - x1);

c5d     if (cross > 0) {
dea         cout << "ESQUERDA\n";
5cd     } else if (cross < 0) {
797         cout << "DIREITA\n";
5ad     } else {
ffc         cout << "TOCANDO\n";
939     }

8d6 }

// Distancia de um ponto a uma reta
8e3 double pointLineDistance(
959     ll x1, ll y1,
0c2     ll x2, ll y2,
0e5     ll x3, ll y3
0a2 ) {
        // Numerador = area do paralelogramo entre p1 p2 e p1 p3
eb7     ll num = (x2 - x1)*(y1 - y3) - (y2 - y1)*(x1 - x3);
1c3     double area2 = abs((double)num);

        // Denominador = comprimento de p1 p2
4dc     double dx = double(x2 - x1);
e8e     double dy = double(y2 - y1);
480     double len = sqrt(dx*dx + dy*dy);

        // distancia
5bc     return area2 / len;
7c0 }

e8d int main(){

c7d     long long x1, y1, x2, y2, x3, y3;
3fc     cin >> x1 >> y1 >> x2 >> y2 >> x3 >> y3;

6f3     double d = pointLineDistance(x1,y1, x2,y2, x3,y3);
71b     cout << fixed << setprecision(6) << d << "\n";
bb3     return 0;
367 }
```

```
// Distancia de um ponto ao plano

b1e double pointPlaneDistance(
    // pontos do plano
1ea     double x1, double y1, double z1,
daa     double x2, double y2, double z2,
cb0     double x3, double y3, double z3,
    // ponto externo
62e     double x0, double y0, double z0
0a2 ) {
        // vetores u = P2-P1, v = P3-P1
c47     double ux = x2 - x1, uy = y2 - y1, uz = z2 - z1;
692     double vx = x3 - x1, vy = y3 - y1, vz = z3 - z1;
        // normal n = u x v
169     double A = uy * vz - uz * vy;
1af     double B = uz * vx - ux * vz;
8d1     double C = ux * vy - uy * vx;
        // coeficiente D
0ba     double D = - (A * x1 + B * y1 + C * z1);

        // avalia distancia
055     double num = fabs(A * x0 + B * y0 + C * z0 + D);
34c     double den = sqrt(A*A + B*B + C*C);
29a     return num / den;
080 }

e8d int main(){
740     double x1,y1,z1, x2,y2,z2, x3,y3,z3, x0,y0,z0;
        // Plano
d15     cin >> x1 >> y1 >> z1
6f2         >> x2 >> y2 >> z2
631         >> x3 >> y3 >> z3;
        // Ponto
494     cin >> x0 >> y0 >> z0;

4d0     double d = pointPlaneDistance(
cb0         x1,y1,z1, x2,y2,z2, x3,y3,z3,
ada         x0,y0,z0
616     );
71b     cout << fixed << setprecision(6) << d << "\n";
bb3     return 0;
bbc }

// Ponto esta dentro ou fora do Plano

e8d int main(){
```

```
        // Plano
136     double x1,y1,z1, x2,y2,z2, x3,y3,z3;
d15     cin >> x1 >> y1 >> z1
6f2         >> x2 >> y2 >> z2
631         >> x3 >> y3 >> z3;

        // Ponto
1e5     double x0,y0,z0;
494     cin >> x0 >> y0 >> z0;

        // Normal (A,B,C) = (P2-P1) (P3-P1)
c47     double ux = x2 - x1, uy = y2 - y1, uz = z2 - z1;
692     double vx = x3 - x1, vy = y3 - y1, vz = z3 - z1;
169     double A = uy*vz - uz*vy;
1af     double B = uz*vx - ux*vz;
8d1     double C = ux*vy - uy*vx;
        // Constante D da equacao do plano
0ba     double D = - (A*x1 + B*y1 + C*z1);

        // Avalia F(P0)
182     double F = A*x0 + B*y0 + C*z0 + D;

934     if (F > 0) {
8e4         cout << "LADO POSITIVO\n";
fe8     } else if (F < 0) {
dc4         cout << "LADO NEGATIVO\n";
d9b     } else {
1c5         cout << "SOBRE O PLANO\n";
3b0     }
f9d }

// Ponto dentro ou fora da circunferencia
e8d int main(){

f2c     double xc, yc, R;
548     cin >> xc >> yc >> R;

662     double x, y;
6b5     cin >> x >> y;

bdd     double dx = x - xc;
dc3     double dy = y - yc;
7d3     double dist2 = dx*dx + dy*dy;
b67     double R2 = R*R;

126     if (dist2 < R2) {
19a         cout << "DENTRO\n";
a1d     }
1a7     else if (dist2 == R2) {
2a3         cout << "SOBRE\n";
d14     }
4e6     else {
045         cout << "FORA\n";
3e1     }
7e5 }
```

5.2 Intersecao de Retangulos

```
2b7 #include <bits/stdc++.h>

ca4 using namespace std;

ef2 struct Rect {
619     int x1, y1, x2, y2;
0ac     int area(){
065         return (y2 - y1) * (x2 - x1);
b37     }
985 };

c93 int intersect(Rect p, Rect q){
6a6     int xOverlap = max(0, min(p.x2, q.x2) - max(p.x1, q.x1));
931     int yOverlap = max(0, min(p.y2, q.y2) - max(p.y1, q.y1));
1e5     return xOverlap * yOverlap;
e02 }
```

5.3 Invertendo Matrizes 90º

```
// 90 graus anti-horario

927 void rotateMatrix(vector<vector<int>>& mat, int N) {
aaa     for (int x = 0; x < N / 2; x++) {
3e3         for (int y = x; y < N - x - 1; y++) {
e14             int temp = mat[x][y];

                // direita   topo
730             mat[x][y] = mat[y][N - 1 - x];

                // fundo   direita
f44             mat[y][N - 1 - x] = mat[N - 1 - x][N - 1 - y];

                // esquerda   fundo
```

```
6bd             mat[N - 1 - x][N - 1 - y] = mat[N - 1 - y][x];

                // topo (temp)   esquerda
f80             mat[N - 1 - y][x] = temp;
bd8         }
276     }
674 }

// 90 graus horario (ou 3x anti-horario)

f9d void rotateMatrixCW(vector<vector<int>>& mat, int N) {
aaa     for (int x = 0; x < N / 2; x++) {
3e3         for (int y = x; y < N - x - 1; y++) {
e14             int temp = mat[x][y];

                // esquerda   topo
3f4             mat[x][y] = mat[N - 1 - y][x];

                // fundo   esquerda
1da             mat[N - 1 - y][x] = mat[N - 1 - x][N - 1 - y];

                // direita   fundo
eae             mat[N - 1 - x][N - 1 - y] = mat[y][N - 1 - x];

                // topo (temp)   direita
d0e             mat[y][N - 1 - x] = temp;
457         }
509     }
ec8 }
```

5.4 Permutacoes de String

```
// CSES - Creating Strings
// https://cses.fi/problemset/task/1622/

2b7 #include <bits/stdc++.h>

ca4 using namespace std;

d25 string str;
f6a vector<string> perms;
e41 int char_count[26];

497 void search(const string &curr = ""){
532     if(curr.size() == str.size()){
5b5         perms.push_back(curr);
505         return;
d12     }
42f     for(int i = 0; i < 26; i++){
962         if(char_count[i] > 0){
19c             char_count[i]--;
efd             search(curr + (char)('a' + i));
411             char_count[i]++;
818         }
08b     }
9ac }

e8d int main(){
912     cin >> str;
e28     for(char c : str){
f9d         char_count[c - 'a']++;
b4a     }

ff4     search();

161     cout << perms.size() << endl;
63a     for(int i = 0; i < perms.size(); i++){
efc         cout << perms[i] << endl;
657     }

bb3     return 0;
e7e }
```

5.5 Problema das 8 Damas

```
2b7 #include <bits/stdc++.h>
f3d #define _ ios_base::sync_with_stdio(0);cin.tie(0);
42a #define endl '\n'
efe #define pb push_back
ca4 using namespace std;

1a8 int n;
ac9 int cnt = 0;

890 void add(vector<bool> &cols, vector<bool> &diag1, vector<bool>
    &diag2, int row, int col) {
3ba     cols[col] = true;
fd8     diag1[row - col + n - 1] = true;
5de     diag2[row + col] = true;
294 }
```

```
b21 void rem(vector<bool> &cols, vector<bool> &diag1, vector<bool>
    &diag2, int row, int col) {
bfe     cols[col] = false;
9fb     diag1[row - col + n - 1] = false;
8b3     diag2[row + col] = false;
d15 }

4a5 void backtracking(int row, vector<bool> &cols, vector<bool>
    &diag1, vector<bool> &diag2) {
e99     if(row == n) {
b8d         cnt += 1;
505         return;
a21     }
27b     for(int col = 0; col < n; col++) {
a5b         if(!cols[col] && !diag1[row - col + n - 1] && !diag2[row +
    col]) {
b88             add(cols, diag1, diag2, row, col);
8c7             backtracking(row + 1, cols, diag1, diag2);
2c6             rem(cols, diag1, diag2, row, col);
ef2         }
4d7     }
826 }

f77 int main(){ _
a68     cin >> n; // number of rows = columns
a4f     vector<bool> cols(n, false), diag1(2 * n - 1, false), diag2(2
    * n - 1, false);
72c     backtracking(0, cols, diag1, diag2);
0a9     cout << cnt << endl;

bb3     return 0;
24f }
```

5.6 Quadrado Perfeito

```
2b7 #include <bits/stdc++.h>

// Time Complexity: O(log n)
// Auxiliary Space: O(1)
d5a bool isPerfectSquare(long double n){
884     if(n >= 0){
            // se a raiz de n e inteira nao havera arredondamento para
    o menor inteiro
569         long long sr = sqrt(n);

            // verifica se houve o arredondamento e retorna a reposta
9b6         return (sr * sr == n);
9b3     }

        //retorna falso caso x < 0
d1f     return false;
e0b }

// Time Complexity: O(log n)
// Auxiliary Space: O(1)
c38 bool binary_isPerfectSquare(long long n){
        // caso 0 e 1
8e8     if(n <= 1) return true;

        // limites da busca binaria
fb9     long long l = 1, r = n;
e47     long long square, mid;
3d5     while(l <= r){
            // calcular valor do meio
b7b         mid = (l + r) / 2;

            // calcular quadrado do termo do meio
f94         square = mid * mid;

e55         if(square == n){
8a6             return true;
cbc         }

            // buscar na direita
852         else if(square < n){
0dc             l = mid + 1;
fb0         }

            // buscar na esquerda
4e6         else {
982             r = mid - 1;
2e2         }
665     }

        // caso saia do loop sem achar um quadrado perfeito
d1f     return false;
f49 }


// Time Complexity: O(sqrt(n))
// Auxiliary Space: O(1)
d5a bool isPerfectSquare(long double n){
5c7     if(floor(sqrt(n)) == ceil(sqrt(n))) return true;
```

```
00b     else return false;
e67 }
```

5.7 Ruina do Jogador

```
2b7 #include <bits/stdc++.h>

ca4 using namespace std;

// p e a probabilidade de ganhar um turno
// q = 1 - p, ou seka, a probabilidade de perder um turno
// Ri e a quantia inicial de dinheiro
// N e quantidade de dinheiro para ser vitorioso

773 double solve(double p, double q, int Ri, int N){
425     if(Ri == 0) return 0;
b21     if(Ri == N) return 1;

        // jogo justo
3b2     if(p == q) return (double)Ri / N;

        // p != q
4c3     return (1 - (double)pow(q / p, Ri)) / (1 - (double)pow(q / p,
    N));
afc }
```

6 Strings

6.1 Manacher's Algorithm

```
// Encontra todos os sub-palindromos de uma string em tempo linear

// O(n)

457 vector<int> manacher(string& s) {
f7b     string t = "#";
b4f     for (char c : s) {
0fa         t.pb(c);
428         t += "#";
297     }

625     int n = t.size();
aa3     t = "$" + t + "^";

1a9     vector<int> p(n + 2);
187     int l = 0, r = 1;
78a     for(int i = 1; i <= n; i++) {
3f5         p[i] = min(r - i, p[l + r - i]);

c2e         while(t[i - p[i]] == t[i + p[i]])
fbd             p[i]++;

03e         if (i + p[i] > r) {
26d             l = i - p[i];
a69             r = i + p[i];
4cc         }
337     }
fa1     return vector<int>(begin(p) + 1, end(p) - 1);
7ca }
```

7 Extra

7.1 Template.cpp

```cpp
#include <bits/stdc++.h>
#define _ ios_base::sync_with_stdio(0);cin.tie(0);
#define endl '\n'
#define pb push_back
#define all(x) (x).begin(), (x).end()

using namespace std;

typedef long long ll;
typedef unsigned long long llu;

const int INF = 0x3f3f3f3f;
const ll LINF = 0x3f3f3f3f3f3f3f3fll;

int main() { _
    int tt;
    cin >> tt;
    while(tt--) {

    }

    return 0;
};
```