

Competitive programming Notebook

Meia noite eu te conto

Contents

1	General	2
1.1	Random	2
1.2	Split	2
1.3	Base Converter	2
1.4	Template	2
2	DS	2
2.1	Dsu	2
2.2	Ordered Set	3
3	DP	3
3.1	Knapsack	3
3.2	Edit Distance	3
4	Graph	3
4.1	Dfs	3



1 General

1.1 Random

```

1 random_device dev;
2 mt19937 rng(dev());
3
4 uniform_int_distribution<mt19937::result_type> dist
  (1, 6); // distribution in range [1, 6]
5
6 int val = dist(rng);

```

1.2 Split

```

1 vector<string> split(string s, char key=' ') {
2     vector<string> ans;
3     string aux = "";
4
5     for (int i = 0; i < (int)s.size(); i++) {
6         if (s[i] == key) {
7             if (aux.size() > 0) {
8                 ans.push_back(aux);
9                 aux = "";
10            }
11        } else {
12            aux += s[i];
13        }
14    }
15
16    if ((int)aux.size() > 0) {
17        ans.push_back(aux);
18    }
19
20    return ans;
21 }

```

1.3 Base Converter

```

1 const string digits = "0123456789
  ABCDEFGHIJKLMNOPQRSTUVWXYZ";
2
3 ll tobase10(string number, int base) {
4     map<char, int> val;
5     for (int i = 0; i < digits.size(); i++) {
6         val[digits[i]] = i;
7     }
8
9     ll ans = 0, pot = 1;
10
11    for (int i = number.size() - 1; i >= 0; i--) {
12        ans += val[number[i]] * pot;
13        pot *= base;
14    }
15
16    return ans;
17 }
18
19 string frombase10(ll number, int base) {
20     if (number == 0) return "0";
21
22     string ans = "";
23
24     while (number > 0) {
25         ans += digits[number % base];
26         number /= base;
27     }
28
29     reverse(ans.begin(), ans.end());
30
31     return ans;
32 }

```

```

33
34 // verifica se um número está na base especificada
35 bool verify_base(string num, int base) {
36     map<char, int> val;
37     for (int i = 0; i < digits.size(); i++) {
38         val[digits[i]] = i;
39     }
40
41     for (auto digit : num) {
42         if (val[digit] >= base) {
43             return false;
44         }
45     }
46
47     return true;
48 }

```

1.4 Template

```

1 // MEIA NOITE EU TE CONTO
2 #include <bits/stdc++.h>
3
4 using namespace std;
5
6 #define _ ios_base::sync_with_stdio(0);cin.tie(0);
7
8 typedef long long ll;
9
10 const int INF = 0x3f3f3f3f;
11 const ll LINF = 0x3f3f3f3f3f3f3f3f;
12
13 int main() { _
14     return 0;
15 }

```

2 DS

2.1 Dsu

```

1 /*
2 DSU - Disjoint Set Union (or Union Find)
3
4 find(x) -> find component that x is on
5 join(a, b) -> union of a set containing 'a' and set
  containing b
6
7 find / join with path compression -> O(inv_Ackermann
  (n)) [O(1)]
8 find / join without path compression -> O(logN)
9
10 https://judge.yosupo.jp/submission/126864
11 */
12
13 struct DSU {
14
15     int n = 0, components = 0;
16     vector<int> parent;
17     vector<int> size;
18
19     DSU(int nn){
20         n = nn;
21         components = n;
22         size.assign(n + 5, 1);
23         parent.assign(n + 5, 0);
24         iota(parent.begin(), parent.end(), 0);
25     }
26
27     int find(int x){
28         if(x == parent[x]) {
29             return x;
30         }

```

```

31 //path compression
32 return parent[x] = find(parent[x]);
33 }
34
35 void join(int a, int b){
36     a = find(a);
37     b = find(b);
38     if(a == b) {
39         return;
40     }
41     if(size[a] < size[b]) {
42         swap(a, b);
43     }
44     parent[b] = a;
45     size[a] += size[b];
46     components -= 1;
47 }
48
49 int sameSet(int a, int b) {
50     a = find(a);
51     b = find(b);
52     return a == b;
53 }
54
55 };

```

2.2 Ordered Set

```

1 // Ordered Set
2 //
3 // set roubado com mais operacoes
4 //
5 // para alterar para multiset
6 // trocar less para less_equal
7 //
8 // ordered_set<int> s
9 //
10 // order_of_key(k) // number of items strictly
    // smaller than k -> int
11 // find_by_order(k) // k-th element in a set (
    // counting from zero) -> iterator
12 //
13 // https://cses.fi/problemset/task/2169
14 //
15 // O(log N) para insert, erase (com iterator),
    // order_of_key, find_by_order
16
17 using namespace __gnu_pbds;
18 template <typename T>
19 using ordered_set = tree<T,null_type,less<T>,
    rb_tree_tag,tree_order_statistics_node_update>;

```

3 DP

3.1 Knapsack

3.2 Edit Distance

```

1 // Edit Distance / Levenshtein Distance
2 //
3 // numero minimo de operacoes
4 // para transformar
5 // uma string em outra
6 //
7 // tamanho da matriz da dp eh |a| x |b|
8 // edit_distance(a.size(), b.size(), a, b)
9 //
10 // https://cses.fi/problemset/task/1639
11 //
12 // O(n^2)
13
14 int tb[MAX][MAX];
15
16 int edit_distance(int i, int j, string &a, string &b)
17 {
18     if (i == 0) return j;
19     if (j == 0) return i;
20
21     int &ans = tb[i][j];
22
23     if (ans != -1) return ans;
24
25     ans = min({
26         edit_distance(i-1, j, a, b) + 1,
27         edit_distance(i, j-1, a, b) + 1,
28         edit_distance(i-1, j-1, a, b) + (a[i-1] != b[
29             j-1])
30     });
31
32     return ans;
33 }

```

4 Graph

4.1 Dfs

```

1 // DFS
2 //
3 // Percorre todos os vertices
4 // priorizando profundidade
5 //
6 // O(n+m)
7
8 vector<vector<int>> g;
9 vector<bool> vis;
10
11 void dfs(int s){
12     if(vis[s]) return;
13     vis[s] = true;
14     for(auto v : g[s]){
15         dfs(v);
16     }
17 }

```