Competitive Programming Notebook

Raul Almeida[1]

[1]Universidade Federal do Paraná

October 14, 2023

Contents

n	not-TLE algorithm	Example
$\leq [10..11]$	$\mathcal{O}(n!)$, $\mathcal{O}(n^6)$	Enumerate permutations
$\leq [15..18]$	$\mathcal{O}(2^n n^2)$	TSP with DP
$\leq [18..22]$	$\mathcal{O}(2^n n)$	Bitmask DP
≤ 100	$\mathcal{O}(n^4)$	3D DP with $\mathcal{O}(n)$ loop
≤ 400	$\mathcal{O}(n^3)$	Floyd-Warshall
$\leq 2 \cdot 10^3$	$\mathcal{O}(n^2 \lg n)$	2 nested loops + tree query
$\leq 5 \cdot 10^4$	$\mathcal{O}(n^2)$	Bubble/Selection/Insertion Sort
$\leq 10^5$	$\mathcal{O}(n \lg^2 n) = \mathcal{O}((\lg n)(\lg n))$	Build suffix array
$\leq 10^6$	$\mathcal{O}(n \lg n)$	MergeSort, build SegTree
$\leq 10^7$	$\mathcal{O}(n \lg \lg n)$	Sieve, totient function
$\leq 10^8$	$\mathcal{O}(n)$, $\mathcal{O}(\lg n)$, $\mathcal{O}(1)$	Mathy solution often with IO bottleneck ($n \leq 10^9$)

10^8 ops/second

1 Theory

1.1 Relevant comparisons

$\lg 10$ (`1E1`)	2.3
$\lg 100$ (`1E1`)	4.6
$\lg 1000$ (`1E2`)	6.9
$\lg 10000$ (`1E3`)	9.2
$\lg 100000$ (`1E4`)	11.5
$\lg 1000000$ (`1E5`)	13.8
$\lg 10000000$ (`1E6`)	16.1
$\lg 100000000$ (`1E7`)	18.4
$\lg 1000000000$ (`1E8`)	20.7
$\lg 10000000000$ (`1E9`)	23.0
$\lg 100000000000$ (`1E10`)	25.3
$\lg 1000000000000$ (`1E11`)	27.6
$\lg 10000000000000$ (`1E12`)	29.9
2^{10}	$\approx 10^3$
2^{20}	$\approx 10^6$

Sign	Type	Bits	Max	Digits
$\pm$	`char`	8	127	2
$+$	`unsigned char`	8	255	2
$\pm$	`short`	16	32 767	4
$+$	`unsigned short`	16	65 535	4
$\pm$	`int/long`	32	$\approx 2 \cdot 10^9$	9
$+$	`unsigned int/long`	32	$\approx 4 \cdot 10^9$	9
$\pm$	`long long`	64	$\approx 9 \cdot 10^{18}$	18
$+$	`unsigned long long`	64	$\approx 18 \cdot 10^{18}$	19
$\pm$	`__int128`	128	$\approx 17 \cdot 10^{37}$	38
$+$	`unsigned __int128`	128	$\approx 3 \cdot 10^{38}$	38

1.2 Prime counting function - pi(x)

Asymptotic to $\frac{x}{\log x}$ by the prime number theorem.

1.3 Progressions

$$a_n = a_k + r(n-k)$$
$$a_n = a_k q^{(n-k)}$$

- r, q: Ratio
- k: Known term

Algorithm	Time	Space
ArticBridges	$\mathcal{O}(V+E)$	$\mathcal{O}(V+E)$
Bellman-Ford	$\mathcal{O}(VE)$	$\mathcal{O}(V+E)$
Dijksta	$\mathcal{O}((V+E)\log V)$	$\mathcal{O}(V^2)$
Edmond Karp	$\mathcal{O}(VE^2)$	$\mathcal{O}(V+E)$
Euler Tour	$\mathcal{O}(E^2)$	
Floyd Warshall	$\mathcal{O}(V^3+E)$	$\mathcal{O}(V^2+E)$
Graph Check	$\mathcal{O}(V+E)$	$\mathcal{O}(V+E)$
Kahn	$\mathcal{O}(VE)$	$\mathcal{O}(V+E)$
Kruskal	$\mathcal{O}(E\log V)$	$\mathcal{O}(V+E)$
LCA	$\mathcal{O}(N\log N)$	$\mathcal{O}(N\log N)$
MCBM	$\mathcal{O}(VE)$	
Prim	$\mathcal{O}(E\log V)$	$\mathcal{O}(V+E)$
Tarjan	$\mathcal{O}(V+E)$	$\mathcal{O}(V+E)$
Extended Euclid	$\mathcal{O}(\log\min(a,b))$	$\mathcal{O}(1)$
Floyd (cycle)	$\mathcal{O}(V)$	$\mathcal{O}(1)$
PrimeFac + OptTrialDiv	$\mathcal{O}(\pi(\sqrt{n}))$	$\mathcal{O}(n)$
Sieve of Eratosthenes	$\mathcal{O}(n\log\log n)$	$\mathcal{O}(n)$
Binary Search	$\mathcal{O}(\log N)$	
Coordinate Compression	$\mathcal{O}(N\log N)$	
KMP	$\mathcal{O}(N)$	
MUF	$\mathcal{O}(AM)$	$\mathcal{O}(N)$
Bottom-Up SegTree	$\mathcal{O}(\log N)$	$\mathcal{O}(N)$

x	10	10^2	10^3	10^4
$\pi(x)$	4	25	168	1 229
x	10^5	10^6	10^7	10^8
$\pi(x)$	9 592	78 498	664 579	5 761 455

- n: Term you want

$$S_n = \frac{n(a_1 + a_n)}{2}$$
$$S_n = \frac{a_1(q^n - 1)}{q - 1}$$

1.4 Series Identities

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} \qquad \sum_{i=1}^{n} i^3 = \frac{n^2(n+1)^2}{4} = \left(\sum_{i=1}^{n} i\right)^2$$

$$g_k(n) = \sum_{i=1}^{n} i^k = \frac{1}{k+1}\left(n^{k+1} + \sum_{j=1}^{k} \binom{k+1}{j+1}(-1)^{j+1} g_{k-j}(n)\right)$$

$$\sum_{i=0}^{n} ic^i = \frac{nc^{n+2} - (n+1)c^{n+1} + c}{(c-1)^2}, \quad c \neq 1$$

$$\sum_{i=0}^{\infty} ic^i = \frac{c}{(1-c)^2}, \quad |c| < 1$$

$$l + (l+1) + \cdots + r = \frac{(l+r)\cdot(r-l+1)}{2}$$

1.5 Binomial Identities

$$\binom{n}{k} = \frac{n}{k}\binom{n-1}{k-1}$$

$$\binom{n-1}{k} - \binom{n-1}{k-1} = \frac{n-2k}{k}\binom{n}{k}$$

$$\binom{n}{h}\binom{n-h}{k} = \binom{n}{k}\binom{n-k}{h}$$

$$\binom{n}{k} = \frac{n+1-k}{k}\binom{n}{k-1}$$

$$\sum_{k=0}^{n} k\binom{n}{k} = n2^{n-1}$$

$$\sum_{k=0}^{n} k^2\binom{n}{k} = (n+n^2)2^{n-2}$$

$$\sum_{j=0}^{k} \binom{m}{j}\binom{n-m}{k-j} = \binom{n}{k}$$

$$\sum_{j=0}^{m} \binom{m}{j}^2 = \binom{2m}{m}$$

$$\sum_{m=0}^{n} \binom{m}{j}\binom{n-m}{k-j} = \binom{n+1}{k+1}$$

$$\sum_{m=k}^{n} \binom{m}{k} = \binom{n+1}{k+1}$$

$$\sum_{r=0}^{m} \binom{n+r}{r} = \binom{n+m+1}{m}$$

$$\sum_{k=0}^{\lfloor n/2 \rfloor} \binom{n-k}{k} = \mathrm{Fib}(n+1)$$

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^{n-k} y^k$$

$$(1+x)^n = \sum_{k=0}^{n} \binom{n}{k} x^k$$

$$2\sum_{i=L}^{R} \binom{n}{i} - \binom{n}{L} - \binom{n}{R} = \sum_{i=L+1}^{R} \binom{n+1}{i}$$

1.6 Lucas' Theorem

$$\binom{n}{m} = \Pi_{i=0}^{k} \binom{n_i}{m_i} \pmod{p}$$

For prime p, n_i and m_i are coefficients of the representations of n and m in base p.

1.7 Fermat Theorems

p is prime

$$\begin{aligned} a^p &= a \pmod{p} \\ a^{p-1} &= 1 \pmod{p} \\ (a+b)^p &= a^p + b^p \pmod{p} \\ a^{-1} &= a^{p-2} \pmod{p} \end{aligned}$$

1.8 Modulo @ exponent

For coprime a, m:

$$a^n \equiv a^{n \bmod \varphi(m)} \pmod{m}$$

Generally, if $n \geq \log_2 m$, then

$$a^n \equiv a^{\varphi(m)+[n \bmod \varphi(m)]} \pmod{m}$$

1.9 Heron's Formula

Area of a triangle ($s = \frac{a+b+c}{2}$)

$$A = \sqrt{s(s-a)(s-b)(s-c)}$$

1.10 Some Primes

- $10^6 + 69$
- $10^9 + 7$
- $10^9 + 9$
- $10^{18} - 11$
- $10^{18} + 3$
- $2^{61} - 1$
- 1000696969
- 998244353
- 999999937
- 1000000007
- 1000000009
- 1000000021
- 1000000033
- $10^{18} - 11$
- $10^{18} + 3$
- $2305843009213693951 = 2^{61} - 1$

1.11 Catalan Numbers

1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, 2674440, 9694845, 35357670, 129644790, 477638700, 1767263190, 6564120420, 24466267020, 91482563640, 343059613650, 1289904147324, 4861946401452, 18367353072152, 69533550916004, 263747951750360, 1002242216651368.

$$C_n = \frac{1}{n+1}\binom{2n}{n} = \frac{(2n)!}{(n+1)!n!} = \Pi_{k=2}^{n} \frac{n+k}{k}, n \geq 0.$$

C_n is:

- The number of valid parenthesis strings with n parentheses
- The number of complete binary trees with $n+1$ leaves
- How many times a $n+2$-sided convex polygon can be cut in triangles conecting its vertices with straight lines

1.12 Binomial

X is the number of successes in a sequence of n independent experiments. $P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$, and $E[X] = np$ and $Var(X) = np(1-p)$.

1.13 Trigonometry

$\sin^2\theta + \cos^2\theta = 1$, $\sin = \frac{opo}{hip}$, $\cos = \frac{adj}{hip}$, $\tan = \frac{opo}{adj}$. $\sin\theta = x \to \arcsin x = \theta$.

α degrees to x rd: $\alpha = \frac{180x}{\pi}$

1.14 Multiples of gcd

Multiples of $\gcd(A, B)$ that are $\in [0, A)$

Let $A, B > 0$, $g = \text{GCD}(A, B)$, $A = ag$ and $B = bg$.

a integers $(0 \times B)\%A$, $(1 \times B)\%A$, $(2 \times B)\%A \ldots ((a-1) \times B)\%A$ correspond to each multiple of g between 0 and $A-1$ (inclusive); note that they are all unique.

1.15 Mod value range

$A < B \implies A\%B = A$, $A \geq B \implies A\%B \leq A/2$

1.16 Expected Value

Avg value of event. For each event, add to the sum the probability of an event times the value of X in that event $\mathbb{E}(X) = \sum_{\omega \in \Omega}(P(\omega) \times X(\omega))$

Another way of looking at it:

$\mathbb{E}(X) = \sum_{i=1}^{M}(i \times P(X = i))$

Since in the expanded version of this sum $P(X = i)$ will appear i times, you're also calculating for each i the probability that $X \geq i$ ($P(x = M)$ will appear M times, once for each i; $P(x = 1)$ will appear exactly once, for $i = 1$; and so on). So

$\mathbb{E}(X) = \sum_{i=1}^{M}(i \times P(X = i)) = \sum_{i=1}^{M} P(X \geq i)$

1.17 Combination

A combination ${}_nC_k = \binom{n}{k}$ (*n chooses k*) refers to selecting k objects from a collection of n where the order of choice doesn't matter.

Without repetition: can't choose an element twice. $\binom{n}{k} = \frac{n!}{r!(n-k)!}$

With repetition: elements may be chosen more than once. $\binom{n}{k} = \frac{(k+n-1)!}{k!(n-1)!}$

1.18 Permutation

A permutation ${}_nP_k$ refers to selecting k objects from a collection of n where the order of choice matters.

With repetition: elements may be chosen more than once. ${}_nP_k = n^k$

Without repetition: can't choose an element twice. ${}_nP_k = \frac{n!}{(n-k)!}$

1.19 90deg rot matrix

```
m[i][j] = m[N-j-1][i]
```

1.20 Max LCP in list of strings

Pra pegar o maior LCP numa lista de strings, basta ordenar ela e comparar cada string i com a $i-1$ e a $i+1$

1.21 Together Square

Let $f(N)$ be the largest square divisor of an integer N, and $P(x)$ whether x is a square number.

For integers i, j, $P(i \times j) \iff \frac{i \times j}{f(i) \times f(j)}$.

Since $\frac{i}{f(i)}$ is indivisible by a prime p twice or more,

$P(\frac{i \times j}{f(i) \times f(j)}) \iff \frac{i}{f(i)} = \frac{j}{f(j)}$

2 Emergency

Pre-submit
Write a few simple test cases if sample is not enough.
Are time limits close? If so, generate max cases.
Is the memory usage fine?
Could anything overflow?
Make sure to submit the right file (check the filename you're editing).
Wrong answer
Print your solution and debug output!
Are you clearing all data structures between test cases?
Can your algorithm handle the whole range of input?
Read the full problem statement again.
Do you handle all corner cases correctly?
Have you understood the problem correctly?
Any uninitialized variables?
Any overflows?
Confusing `N` and `M`, `i` and `j`, etc.?
Are you sure your algorithm works?
What special cases have you not thought of?
Are you sure the STL functions you use work as you think?
Add some assertions, maybe resubmit.
Create some testcases to run your algorithm on.
Go through the algorithm for a simple case.
Go through this list again.
Explain your algorithm to a teammate.
Ask the teammate to look at your code.
Go for a small walk, e.g. to the toilet.
Is your output format correct? (including whitespace)
Rewrite your solution from the start or let a teammate do it.
Runtime error
Have you tested all corner cases locally?
Any uninitialized variables?
Are you reading or writing outside the range of any vector?
Any assertions that might fail?
Any possible division by 0? (`mod 0` for example)
Any possible infinite recursion?
Invalidated pointers or iterators?
Are you using too much memory?
Debug with resubmits (e.g. remapped signals, see Various).
Time limit exceeded
Do you have any possible infinite loops?
What is the complexity of your algorithm?
Are you copying a lot of unnecessary data? (use references)
How big is the input and output? (consider `scanf` and `printf`)
Avoid `vector`, `map`. (use `array`/`unordered_map`)
What do your teammates think about your algorithm?
Memory limit exceeded
What is the max amount of memory your algorithm should need?
Are you clearing all data structures between test cases?

3 Geometry

3.1 Points

```
using pt = complex<double>;
#define px real()
#define py imag()

double dot(pt a, pt b) { return (conj(a)*b).px; }
double cross(pt a, pt b) { return (conj(a)*b).py; }
pt vec(pt a, pt b) { return b-a; }
int sgn(double v) { return (v > -EPS) - (v < EPS); }
int seg_ornt(pt a, pt b, pt c) {
  return sgn(cross(vec(a, b), vec(a, c)));
}
int ccw(pt a, pt b, pt c, bool col) {
  int o = seg_ornt(a, b, c);
  return (o == 1) || (o == 0 && col);
}
const double PI = acos(-1);
double angle(pt a, pt b, pt c) {
  return abs(remainder(arg(a-b) - arg(c-b), 2.0*PI));
}
```

3.2 Convex Hull (Monotone)

$\mathcal{O}(n \log n)$, ans is vector of point indexes

```
using pti = pair<pt, int>;
#define fi first
#define se second
vi convex_hull(vector<pti>& ps, bool col = false) {
  int k = 0, n = ps.size(); vi ans (2*n);
  sort(all(ps), [](pti a, pti b) {
    return make_pair(a.fi.px, a.fi.py) < make_pair(b.fi.px,
        b.fi.py);
  });
  for (int i = 0; i < n; i++) {
    while (k >= 2 && !ccw( /* lower hull */
        ps[ans[k-2]].fi, ps[ans[k-1]].fi, ps[i].fi, col)) {
          k--; }
    ans[k++] = i;
  }
  if (k == n) {
    ans.resize(n);
    for (auto &i : ans) i = ps[i].second;
    return ans; }
  for (int i = n-2, t = k+1; i >= 0; i--) {
    while (k >= t && !ccw( /* upper hull */
        ps[ans[k-2]].fi, ps[ans[k-1]].fi, ps[i].fi, col)) {
          k--; }
    ans[k++] = i;
  }
  ans.resize(k-1);
  for (auto &i : ans) i = ps[i].second;
  return ans;
}
```

4 Graph

4.1 Prim MST

Time	Space
$\mathcal{O}(E \log V)$	$\mathcal{O}(V + E)$

```
vi par(N, -1);
vector<ll> d(N, oo);
vector<vector<wv>> mst(N);

ll prim(int s) {
  ll sum = 0;
  priority_queue<wv, vector<wv>, greater<wv>> Q;
  auto add = [&](int v, ll x, int p) {
    if (x < d[v]) {
      Q.emplace(d[v]=x, v);
      par[v] = u;
    }};
  add(s, 0, s);
  while (Q.size()) {
    auto [w, v] = Q.top(); Q.pop();
    if (vis[u]) continue;
    vis[u] = true;
    if (par[u] != -1) {
      int p = par[v]; ll w = d[v];
      mst[v].emplace_back(w, u);
      mst[u].emplace_back(w, v);
    }
    sum += w;
    for (auto [c, u] : g[v])
      if (!vis[u])
        add(u, c, v);
  }
  return sum;
}
```

4.2 Dijkstra SSSP

Time	Space
$\mathcal{O}((V + E) \log V)$	$\mathcal{O}(V^2)$

```
vi d(MAXN, oo);

void dijkstra(int s) {
  priority_queue<wv, vector<wv>, greater<wv>> pq;
  pq.emplace(d[s] = 0, s);
  add(s, 0);
  while (!pq.empty()) {
    auto [w, v] = pq.top(); pq.pop();
    if (w > dist[v]) continue;
    for (auto [x, u] : g[v])
      if (w+x < d[u])
        pq.emplace(d[u]=w+x, u);
  }
}
```

4.3 Graph Check

Time/Space	Usage
$\mathcal{O}(V + E)$	`graphCheck(firstVertex, -1)`

```
int UNVISITED = -1, EXPLORED = 0, VISITED = 1;
vector<vi> adj(M);
vi tin;

void graphCheck(int v, int p) { //vertex, parent
  tin[v] = EXPLORED;
  for (auto u: adj[v]) {
    if (tin[u] == UNVISITED) { //tree edge
      graphCheck(u, v);
    } else if (tin[u] == EXPLORED) {
      if (u == p)
        ; //two way edge u <-> v
      else
        ; //back edge v -> u
    } else if (tin[u] == VISITED) {
      ; //forward/cross edge u-v
    }
  }
  tin[v] = VISITED;
}
```

4.4 Articulations and Bridges

Time/Space	Usage
$\mathcal{O}(V + E)$	`dfs(src, -1)`

```

int tk = 0;
vi tin(M, -1);
```

```
vector<vi> adj(M);

void dfs(int v, int p) {
  tin[v] = low[v] = tk++;
  int children = 0;
  for (auto u: adj[v]) {
    if (u == p) continue;
    else if (tin[u] == -1) {
      ++children;
      dfs(u, v);
      if (low[u] >= tin[v] && p != v)
        ; //articulation point
      if (low[u] > tin[v])
        ; //bridge u-v
      low[v] = min(low[v], low[u]);
    } else {
      low[v] = min(low[v], tin[u]);
    }
  }
}
```

4.5 Euler Tour

Time	Usage
$\mathcal{O}(E^2)$	`tour(cyc.begin(), start_vertex)`

```
list<int> cyc;
vector<vi> adj(M);
vector<vector<bool>> traversed(M, vector<bool>(M, false));

//euler tour (list for fast insertion)
void tour(list<int>::iterator i, int v) {
  for (auto u: adj[v]) {
    if (!traversed[v][u]) {
      traversed[v][u] = true;
      for (auto t: adj[u])
        if (t == v && !traversed[u][t]) {
          traversed[u][t] = true;
          break;
        }
      tour(cyc.insert(i, v), u);
    }
  }
}
```

4.6 FFEK MaxFlow

Time	Space
$\mathcal{O}(VE^2)$	$\mathcal{O}(V+E)$

```

vector<vi> capacity(M, vi(M, 0)), adj(M);
vector<ii> mc; //mincut edges

int bfs(int s, int t, vi &par) {
  fill(all(par), -1);
  par[s] = -2;
  queue<ii> q; q.push({s, inf});
  while (!q.empty()) {
    int v = q.front().first,
      flow = q.front().second;
    q.pop();
    for (auto u: adj[v])
      if (par[u] == -1 && capacity[v][u]) {
        par[u] = v;
        int new_flow = min(flow, capacity[v][u]);
        if (u == t) return new_flow;
        q.push({u, new_flow});
      }
  }
  return 0;
}

int maxflow(int s, int t) {
  int flow = 0;
  vi par(M);
  int new_flow;
  while ((new_flow = bfs(s, t, par))) {
    flow += new_flow;
    int v = t;
    while (v != s) {
      int p = par[v];
      capacity[p][v] -= new_flow;
      capacity[v][p] += new_flow;
      v = p;
    }
  }
  return flow;
}

void mincut(int s, int t) {
  maxflow(s, t);
  stack<int> st;
  vector<bool> visited(n, false);
  vector<ii> ans;
  st.push(s); // changed from 0 to s
  while (!st.empty()) {
    int v = st.top(); st.pop();
    if (visited[v]) continue;
    visited[v] = true;
    for (auto u: adj[v])
      if (capacity[v][u] > 0)
        st.push(u);
      else
        ans.push_back({v, u});
  }
  mc.clear();
  for (auto &[v, u] : ans)
    if (!visited[u])
      mc.push_back({v, u});
}
```

4.7 Heavy-Light Decomposition

Query	Setup	Update
$\mathcal{O}(\log^2 n)$	define `oper(a,b)` for query	`rmq.upd(pos[x],v)`

Queries on edges: assign values of edges to child node, then change `pos[x]` to `pos[x]+1` in query (see `!!!`)

```
vi g[MAXN];
int wg[MAXN],par[MAXN],h[MAXN]; // subtree size,father,height
void dfs1(int x){
  wg[x]=1;
  for(int y:g[x])if(y!=par[x]){
    par[y]=x;h[y]=h[x]+1;dfs1(y);
    wg[x]+=wg[y];
  }
}
int curpos,pos[MAXN],head[MAXN]; // head = representante
void hld(int x, int c){
  if(c<0)c=x;
  pos[x]=curpos++;head[x]=c;
  int mx=-1;
  for(int y:g[x])if(y!=par[x]&&(mx<0||wg[mx]<wg[y]))mx=y;
  if(mx>=0)hld(mx,c);
  for(int y:g[x])if(y!=mx&&y!=par[x])hld(y,-1);
}
void hld_init(){par[0]=-1;h[0]=0;dfs1(0);curpos=0;hld(0,-1);}
int query(int x, int y, stree& rmq){
  int r=NEUT;
  while(head[x]!=head[y]){
    if(h[head[x]]>h[head[y]])swap(x,y);
    r=oper(r,rmq.query(pos[head[y]],pos[y]+1));
    y=par[head[y]];
  }
  if(h[x]>h[y])swap(x,y); // now x is lca
  r=oper(r,rmq.query(pos[x],pos[y]+1)); // !!!
  return r;
}
```

4.8 Kahn's topological sort

Time	Space
$\mathcal{O}(VE)$	$\mathcal{O}(V+E)$

```cpp
vector<vi> adj(M);
vi sorted;

void kahn(int n) {
  vi indeg(n, 0);
  vector<bool> valid(n, true);
  priority_queue<int> pq;

  for (int v = 0; v < n; ++v)
    for (auto u: adj[v])
      indeg[u]++;
  for (int v = 0; v < n; ++v)
    if (!indeg[v]) pq.push(v);

  while (!pq.empty()) {
    int v = pq.top(); pq.pop();
    sorted.push_back(v);
    valid[v] = false;
    for (auto u: adj[v])
      if (valid[u] && !(--indeg[u]))
        pq.push(u);
  }
}
```

4.9 MCBM (Kunh Matching)

$\mathcal{O}(VE)$ time

Max Cardinality Bipartite Matching

`g`: arcos do lado esquerdo pro lado direito do grafo

```cpp
vi mat;
vector<bool> vis;

bool match(int v) {
  if (vis[v]) return false;
  vis[v] = true;
  for (int u: g[v])
    if (mat[u] < 0 || match(mat[u])) {
      mat[u] = v;
      return true;
    }
  return false;
}

mat.assign(n, -1);
int mcbm = 0; // num matched vertices
for (int i = 0; i < n; ++i) {
  vis.assign(n, false);
  mcbm += try_kuhn(i);
}
// match: mat[i] -> i (i é do lado direito)
```

4.10 LCA - Binary lifting

$\mathcal{O}(n \log n)$ time	$\mathcal{O}(n \log n)$ space

```cpp
int L = //log2(n)

void dfs(int v, int p) { // uso: dfs(raiz, raiz)
  up[v][0] = p;
  for (int l = 1; l <= L; ++l)
    up[v][l] = up[up[v][l-1]][l-1];
  for (int u : g[v])
    if (u != p) dfs(u, v);
}

int lca(int a, int b) {
  if (dep[b] >= dep[a]) { swap(a, b); }
  int diff = dep[a] - dep[b];
  for (int l = L; l >= 0; l--) if (diff & (1 << l))
    a = up[a][l];
  if (a == b) { return a; }
  for (int l = L; l >= 0; l--) if (up[a][l] != up[b][l])
    a = up[a][l], b = up[b][l];
  return up[a][0];
}
```

4.11 Tarjan Strongly Connected Component

Time/Space	Usage
$\mathcal{O}(V+E)$	`Tarjan(n, adj)`

```cpp
vi tin(M, -1), low(M, -1);
vector<bool> vis(M);
vector<vi> adj(M);
stack<int> S;
int tk = 0;

void dfs(int v) {
  low[v] = tin[v] = tk++;
  S.push(v);
  vis[v] = true;
  for (auto u: adj[v]) {
    if (tin[u] == -1)
      dfs(u);
    if (vis[u])
      low[v] = min(low[v], low[u]);
  }
  if (low[v] == tin[v])
    while (true) {
      int u = S.top(); S.pop(); vis[u] = false;
      if (u == v) break;
    }
}
```

4.12 LCA - Euler Path

$\mathcal{O}(n \log n)$ time	$\mathcal{O}(n)$ space

```cpp
vi idx(n);
int tk = 1;

void dfs(int v, int d) { // call with dfs(root, 0);
  for (auto u : adj[v]) {
    st.update(tk, {d, v});
    tk++;
    dfs(u, d+1);
  }
  idx[v] = tk;
  st.update(tk, {d, v});
  tk++;
}

int lca(int v, int u) {
  int l = idx[v], r = idx[u];
  return st.minquery(l, r).second; // .first is depth
}
```

4.13 Kosaraju SCC

Time/Space	Usage
$\mathcal{O}(V+E)$	`kosaraju()`

`rep`: representante do componente de cada vtx

`scc`: 2a dfs, processa os vtx do componente c

```cpp
vi S, rep(MAXN);

void dfs(int v) {
  vis[v] = true;
  for (int u: g[v])
    if (!vis[u]) dfs(u);
  S.push_back(v);
}
```

```cpp

void scc(int v, int c) {
  vis[v] = true;
  rep[v] = c;
  for (int u: gi[v])
    if (!vis[u]) scc(u, c);
}

void kosaraju() {
  for (int i = 0; i < n; ++i)
    if (!vis[i]) dfs(i);
  vis.assign(n, false);
  reverse(all(S));
  for (int v: order)
    if (!vis[v]) scc(v, v);
}
```

4.14 Bellman-Ford SSSP

Time	Space
$\mathcal{O}(VE)$	$\mathcal{O}(V+E)$

```cpp
const int inf = 0x3f3f3f3f;
vector<vector<ii>> adj(M);
vi dist(M, inf);

void bellmanFord(int n) {
  for (int i = 0; i < n-1; ++i)
    for (int v = 0; v < n; ++v)
      for (auto &[u, w]: adj[v])
        if (dist[v] != inf)
          dist[u] = min(dist[u], dist[v]+w);
}

//check if there are negative cycles
bool cycle(int n) {
  bool ans = false;
  for (int v = 0; v < n; ++v)
    for (auto &[u, w]: v)
      ans |= dist[v] != inf && dist[u] > dist[v]+w;
}
```

4.15 Kruskal MST

Time	Space
$\mathcal{O}(E \log V)$	$\mathcal{O}(V+E)$

```cpp
using edge = tuple<ll, int, int>; // peso, u, v
vector<edge> edges;
UnionFind muf;

pair<ll, vector<edge>> kruskal(int n) { // n = #vertices
  vector<edge> mst;
  ll cost = 0; sort(all(edges));
  for (auto [w, u, v] : edges)
    if (!muf.isSameSet(u, v)) {
      mst.emplace_back(w, u, v);
      cost += w;
      muf.unionSet(u, v);
    }
  return {cost, mst};
}
```

4.16 Floyd Warshall APSP

Time	Space	Usage
$\mathcal{O}(V^3+E)$	$\mathcal{O}(V^2+E)$	`FloydWarshall(n, edges)`

```cpp
vector<vi> w(MAXN, vi(MAXN, oo));
void fw(int n) {
  for (int m = 0; m < n; ++m)
    for (int u = 0; u < n; ++u)
      for (int v = 0; v < n; ++v)
        if (max(w[u][m], w[m][v]) < oo)
          w[u][v] = min(w[u][v], w[u][m]+w[m][v]);
}
```

5 Math

5.1 Sieve of Eratosthenes

Time	Space
$\mathcal{O}(n \log\log n)$	$\mathcal{O}(n)$

```cpp

bitset<11234567> pr;
vi factors(M, 0);
vi primes;

void sieve(int n) {
  pr.set();
  for (int i = 2; i*i <= n; ++i)
    if (pr[i]) { //factors[i] == 0
      primes.push_back(i);
      for (int p = i*i; p <= n; p += i) {
        pr[p] = false;
        factors[p]++;
      }
    }
}

// O(1) for small n, O(sieve_size) else
bool isPrime(int n) {
  int sieve_size = 11234567;
  if (n <= sieve_size) return pr[n];
  for (auto p: primes) // only works if n <= primes.back()^2
    if (!(n%p)) return false;
  return true;
}
```

5.2 Prime Factors w/ Optimized Trial Divisions

Time	Space
$\mathcal{O}(\pi(\sqrt{n}))$	$\mathcal{O}(n)$

```cpp

vi primes;
vector<ii> factors;

void pf(int n) {
  for (auto p: primes) {
    if (p*p > n) break;
    int i = 0;
    while (!(n%p)) {
      n /= p;
      i++;
    }
    factors.push_back({p, i});
  }
  if (n != 1) factors.push_back({n, 1});
}
```

5.3 Extended Euclid for Linear Diophantines

Time	Usage for **a,b**
$\mathcal{O}(\log \min(a,b))$	`int x, y; gcd(a, b, x, y);`

```cpp
int gcd(int a, int b, int& x, int& y) {
  if (!b) {
    x = 1;
    y = 0;
    return a;
  }
  int x1, y1;
  int d = gcd(b, a % b, x1, y1);
```

```cpp
  x = y1;
  y = x1 - y1 * (a / b);
  return d;
}
```

5.4 Floyd's algorithm cycle-finding

$\mathcal{O}(V)$ time

```cpp
int findCycle(int x) {
  int a, b;
  a = succ(x);
  b = succ(succ(x));
  while (a != b) {
    a = succ(a);
    b = succ(succ(b));
  }
  a = x;
  while (a != b) {
    a = succ(a);
    b = succ(b);
  }
  int first = a; // first element in cycle
  b = succ(a);
  int length = 1;
  while (a != b) {
    b = succ(b);
    length++;
  }
}
```

6 Paradigm

6.1 Coordinate Compression

Normalize vector access; can also be done with map/set but high constant. $\mathcal{O}(n \log n)$ time

```cpp
vi v, vals, cv; // all same size, cv = compressed v
vals = v;
sort(all(vals));
vals.erase(unique(all(vals)), vals.end());
for (int i = 0; i < n; ++i)
  cv[i] = lower_bound(all(vals), v[i]) - vals.begin();
```

6.2 128 Bit Integers

```cpp
// cout, cerr, etc; may over/underflow
ostream& operator<<(ostream& out, __int128 x) {
    if (x == 0) return out << 0;
    string s; bool sig = x < 0; x = x < 0 ? -x : x;
    while(x > 0) s += x % 10 + '0', x /= 10;
    if (sig) s += '-';
    reverse(all(s));
    return out << s;
}
// cin, etc; may over/underflow
istream& operator>>(istream& in, __int128& x) {
    char c, neg = 0; while(isspace(c = in.get()));
    if(!isdigit(c)) neg = (c == '-'), x = 0;
    else x = c - '0';
    while(isdigit(c = in.get())) x = (x << 3) + (x << 1) -
          '0' + c;
    x = neg ? -x : x; return in;
}
```

6.3 Prefix AND

$\mathcal{O}(\log N)$ bitwise and on array, $\mathcal{O}(N \log N)$ build

```cpp
vector<vi> ps(n, vi(32));
const int L = 32; // teto(log2(max(v[i])))

void build(vi &v) {
  int n = v.size();
  for (int i = 0; i < n; ++i) // build
    for (int b = 0; b < L; ++b) {
      if (i > 0) ps[i][b] = ps[i-1][b];
      if (v[i] & (1ll << b)) ps[i][b]++;
    }
}

int qry(int l, int r) {
  int num = 0;
  for (int b = 0; b < L; ++b)
    // ligado em todos
    if (ps[r][b] - (l ? ps[l-1][b] : 0) == r-l+1)
      num |= (1ll << b);
  return num;
}
```

6.4 Binary Search

```cpp
// std
int l = 0, r = n-1;
while (l <= r) {
  int m = l+(r-l)/2;
  if (array[m] == x) // found
  if (array[m] > x) r = m-1;
  else l = m+1;
}
// nice - binary steps
int k = 0;
for (int b = n/2; b > 0; b /= 2)
  while (k+b < n && array[k+b] <= x)
    k += b;
if (array[k] == x) // found
```

7 Problems

7.1 Tree Distances I

Given a tree of n nodes, determine for each node its maximum distance to another node

```cpp
vi d(maxn);

void dfs1(int v, int p=-1) { // d[v] = max dist below
  d[v] = 0;
  for (int u : g[v])
    if (u != p) {
      dfs1(u, v);
      d[v] = max(d[v], 1+d[u]);
    }
}

void add(int &a, int &b, int c) {
  if (c > a)
    tie(a, b) = make_pair(c, a);
  else if (c > b)
    b = c;
}

void dfs2(int v, int a, int p=-1) { // include dist above
  // a,b = max and 2nd max subtrees
  int b = 0;
  for (int u : g[v])
    if (u != p)
      add(a, b, 1+d[u]);
  for (int u : g[v])
    if (u != p) {
      int dist = (1+d[u]) == a ? b : a;
      dfs2(u, dist+1, v);
    }
  d[v] = max(d[v], a);
}

signed main() {
  dfs1(src);
  dfs2(src, 0);
  rep(i,0,n)
    cout << d[i] << '\n';
```

```
}
```

7.2 Coin Combinations II

Given money system of n coins $c_i > 0$. How many distinct **ordered** ways to produce sum x using these coins? (use each one as many time as you want, order doesn't matter i.e. $2+2+5 = 2+5+2$)

```
int n, x;
cin >> n >> x;
vector<int> c(n);

for (auto &i: c)
  cin >> i;

vector<int> dp(x+1);
const int M = 1000000007;

dp[0] = 1;
for (int k: c)
  for (int j = k; j <= x; ++j)
    dp[j] = (dp[j] + dp[j-k]) % M;

cout << dp[x] << '\n';
```

7.3 Kth smallest sum

Given an array A, sum at least one element, may use the same element multiple times. Find K-th smallest possible sum

$1 \le N \le 10, 1 \le K \le 2\times10^5, 1 \le A_i \le 10^9$

```
int n, k;
cin >> n >> k;
vector<int> a(n);
priority_queue<int, vector<int>, greater<int>> pq;
set<int> seen;
for (int &x : a) {
  cin >> x;
  if (seen.find(x) == seen.end()) {
    pq.push(x);
    seen.insert(x);
  }
}
int cnt = 0;
while (pq.size()) {
  int t = pq.top(); pq.pop();
  if (++cnt == k) {
    cout << t << '\n';
    break;
  }
  for (int x : a)
    if (seen.find(t+x) == seen.end()) {
      pq.push(t + x);
      seen.insert(t+x);
    }
}
```

7.4 Art Gallery on Graph

Given graph with $N \le 2\times10^5$ vertices, $M \le 2\times10^5$ edges.

$K \le N$ security guards are on some vertices, each with a different stamina h_i.

Vertex v is guarded if there is a guard i with distance $\le h_i$

List all guarded vertices

```
priority_queue<pair<int, int>> pq;
vector<int> dist(n, -1);

for (int i = 0; i < k; ++i) {
  int p, h;
  cin >> p >> h;
  --p;
  pq.emplace(dist[p] = h, p);
}

while (pq.size()) {
  auto [h, p] = pq.top(); pq.pop();
  if (h != dist[p])
    continue;
  for (int &q : g[p])
    if (h-1 > dist[q])
      pq.emplace(dist[q] = h-1, q);
}

vector<int> ans;
for (int i = 0; i < n; ++i)
  if (dist[i] >= 0)
    ans.push_back(i+1);
```

7.5 Coin Combinations I

Given money system of n coins $c_i > 0$. How many ways to produce sum x using these coins? (use each one as many time as you want, order matters i.e. $2+2+5 \neq 2+5+2$)

```
int n, x;
cin >> n >> x;
vector<int> c(n);
for (auto &i : c)
  cin >> i;
const int M = int(1e9)+7;
vector<int> ans(x+1);
ans[0] = 1;
for (int i = 1; i <= x; ++i)
  for (int ci : c)
    if (ci <= i)
      ans[i] = (ans[i] + ans[i-ci]) % M;
cout << ans[x] << '\n';
```

8 String

8.1 Rolling hash (linear)

$\mathcal{O}(n)$ time

Let $h_{i..j} = \text{hash}(s_{i..j})$.

$h_{i..j} \times p^i = h_{0..j} - h_{0..i-1}$. Instead of finding the multiplicative inverse of p^i, you can multiply this term by p^{n-i} (so every hash is compared multiplied by p^n).

```
ll hash(string const& s) {
  const int p = 31; // ~alphabet size (31 for lowercase, 53
      for uppercase)
  const int M = 1e9 + 9;
  ll h = 0;
  ll p_pow = 1; // precompute for performance
  for (char c : s) {
    h = (h + (c - 'a' + 1)*p_pow) % M;
    p_pow = (p_pow * p) % M;
  }
  return h;
}
```

8.2 Prefix Function (KMP)

$\mathcal{O}(n)$ time

To find ocurrences of `s` in `t`, use the string `s+%+t`, then look for `pi[i] = s.length()` on the "`t` side"

```
vi prefix(string s) {
  int n = s.length();
  vi pi(n, 0); // can be optimized if you know max prefix
      length
  for (int i = 1; i < n; ++i) {
    int j = pi[i-1];
    while (j > 0 && s[i] != s[j])
      j = pi[j-1];
```

```cpp
    if (s[i] == s[j])
      j++;
    pi[i] = j;
  }
  return pi;
}
```

8.3 Suffix Array

Build	Query
$\mathcal{O}(n \log n)$	$\mathcal{O}(\log n)$

To find whether p is a substring of s (and where this ocurrence starts), you can build the suffix array A of s. Since A is sorted, you can binary search for p as a prefix of all suffixes of s. **Complexity (besides construction):** $\mathcal{O}(|p| \log(|s|))$.

```cpp
// sort p by the values in c (stable) (O(|alphabet| + n))
void count_sort(vi &p, vi &c) {
  int n = p.size();
  int alphabet = 256; // ascii range
  vi cnt(max(alphabet, n));
  for (auto x : c)
    cnt[x]++;

  vi pos(max(alphabet, n));
  pos[0] = 0;
  for (int i = 1; i < max(alphabet, n); ++i)
    pos[i] = pos[i-1] + cnt[i-1];

  vi p_sorted(n);
  for (auto x : p) {
    p_sorted[pos[c[x]]++] = x;
  }

  p = p_sorted;
}

// build suffix array
vi suffix_array(string s) {
  s += "$";
  int n = s.size();
  // at k = 2^0, sort strings of length 1
  vi p(n), c(n); // suffix start position, equivalence class
  for (int i = 0; i < n; ++i) {
    p[i] = i;
    c[i] = s[i];
  }
  // at first c is just a hack to sort p, it's not really equiv. class
  count_sort(p, c);
  // but then it is
  c[p[0]] = 0;
  for (int i = 1; i < n; ++i) {
    c[p[i]] = c[p[i-1]];
    if (s[p[i]] != s[p[i-1]])
      c[p[i]]++;
  }
  int k = 1;
  while (k < n) {
    // transition from k to k+1
    for (int i = 0; i < n; ++i)
      p[i] = (p[i] - k + n) % n;
    count_sort(p, c);
    // recalculate equiv.
    vi c_upd(n);
    c_upd[p[0]] = 0;
    for (int i = 1; i < n; ++i) {
      ii prev = {c[p[i-1]], c[(p[i-1] + k)%n]};
      ii curr = {c[p[i]], c[(p[i] + k)%n]};
      c_upd[p[i]] = c_upd[p[i-1]];
      if (curr != prev)
        c_upd[p[i]]++;
    }
    c = c_upd;
    k <<= 1;
  }
  return p;
}
```

9 Structure

9.1 MUF with distance to root

`par[i].se` tem a distancia, nesse caso só precisa da paridade então usamos xor. Se quiser dist toda, usa soma (!!!)

`merge` retorna 1 se o componente for bipartido

```cpp
struct MUF {
  int n;
  vector<ii> par;
  vi rk, bip; // bipartite check

  MUF(int n) : n(n), par(n), rk(n, 0), bip(n, 1) {
    rep(i,0,n)
      par[i] = {i,0};
  }

  ii find(int i) {
    if (par[i].fi != i) {
      int p = par[i].se;
      par[i] = find(par[i].fi);
      par[i].se ^= p; // !!!
    }
    return par[i];
  }

  int merge(int a, int b) {
    int x, y;
    tie(a, x) = find(a);
    tie(b, y) = find(b);
    if (a == b)
      bip[a] &= x != y;
    if (rk[a] < rk[b])
      swap(a, b);
    par[b] = {a, x^y^1}; // !!!
    bip[a] &= bip[b];
    rk[a] += rk[a] == rk[b];
    return bip[a];
  }

  bool same(int i, int j) {
    return find(i) == find(j);
  }
};
```

9.2 Merge/Disjoint Union-Find

Time	Space	Usage
$\mathcal{O}(A \times n)$	$\mathcal{O}(n)$	`muf(n)`

```cpp
struct MUF {
  int n;
  vi par, rk, sz;

  MUF(int n) : n(n), par(n), rk(n, 0), sz(n, 1) {
    rep(i,0,n)
      par[i] = i;
  }

  int find(int i) {
    return par[i] == i ? i : (par[i] = find(par[i]));
  }

  int merge(int a, int b) {
    a = find(a); b = find(b);
    if (a != b)
      sz[a] = sz[b] = (sz[x] + sz[y]);
    if (rk[a] < rk[b])
      swap(a, b);
    par[b] = a;
    rk[a] += rk[a] == rk[b];
    return sz[x];
```

```
  }

  bool same(int i, int j) {
    return find(i) == find(j);
  }
};
```

9.3 Bottom-Up Segment Tree

Build	Query	Update	Usage
$\mathcal{O}(n)$	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$	`seg(n)`

Uses less space than top-down $4n$ segtree ($2n$ here)

```
struct seg {
  int n;
  vi t;

  seg(vi v) : n(v.size()), t(2*n) {
    for (int i = 0; i < n; ++i)
      upd(i, v[i]);
  }
  seg(int sz) : n(sz), t(2*n) {}

  int query(int a, int b) {
    int ans = 0;
    for (a += n, b += n; a <= b; ++a /= 2, --b /= 2) {
      if (a%2 == 1) ans += t[a];
      if (b%2 == 0) ans += t[b];
    }
    return ans;
  }

  void upd(int p, int x) {
    t[p += n] = x;
    while (p /= 2) t[p] = t[p<<1] + t[(p<<1)+1];
  }
};
```

9.4 Segment Tree

Build	Query	Modify	Usage
$\mathcal{O}(n \log n)$	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$	`stree(n)`

```
struct stree {
  int n;
  vi st, v;

  stree(vi v): n(v.size()), st(4*n), v(v) {
    build(1, 0, n-1); }
  }

  int left(int i) { return i<<1; }
  int right(int i) { return (i<<1)+1; }

  void build(int p, int pl, int pr) {
    if (pl == pr) {
      st[p] = v[pl];
      return;
    }
    int m = (pl+pr)/2;
    build(left(p), pl, m);
    build(right(p), m+1, pr);
    st[p] = min(st[left(p)], st[right(p)]);
  }

  int query(int p, int pl, int pr, int ql, int qr) {
    // same params as update, except [ql..qr] is the query
        range
    if (qr < pl || ql > pr) return inf;
    if (ql <= pl && pr <= qr) return st[p];
    int m = (pl+pr)/2;
    int query_left = query(left(p), pl, m, ql, qr);
    int query_right = query(right(p), m+1, pr, ql, qr);
    return min(query_left, query_right);
  }

  int query(int ql, int qr) { return query(1, 0, n-1, ql,
      qr); }

  void update(int p, int pl, int pr, int i, int x) {
    // p = st idx, corresponds to range [pl..pr]
    if (i < pl || i > pr) return;
    if (pl == pr) {
      st[p] = x;
      return;
    }
    int m = (pl+pr)/2;
    update(left(p), pl, m, i, x);
    update(right(p), m+1, pr, i, x);
    st[p] = min(st[left(p)], st[right(p)]);
  }

  void update(int i, int x) { update(1, 0, n-1, i, x); }
};
```

10 Extra

10.1 C++ structs

```
struct st {
  vi a;
  vector<bool> b = vector<bool>(5); // default value
  int i;
  st(int _i) : a(_i), i(_i) {};
  bool operator< (st& e) const { return i < e.i; }
};

st e = st(3); st f(3);

struct matrix {
  vector<vi> m;
  matrix(int n) m(n, vi(n)) {};
  matrix operator * (const matrix &b) {
    matrix c = matrix();
    for (int i = 0; i < m.size(); ++i)
      for (int j = 0; j < m.size(); ++j)
        for (int k = 0; k < m.size(); ++k)
          c.m[i][j] = c.m[i][j] + 1LL*m[i][k]*b.m[k][j];
    return c;
  }
};.
```

10.2 cmp

```
// upper_bound: 1st > x, lower_bound: 1st >= x
// last <= x: up-1, first >= x: lo
priority_queue<int, vector<int>, greater<int>> pq;
struct {
  bool operator()(const int& a, const int& b) const {
    return a < b;
  }
} cmp;
priority_queue<int, vector<int>, cmp> pq2;
sort(all(v), cmp);
```

10.3 Vim

```
set et ts=2 sw=2 ai si cindent sta is tm=50 nu noeb sm "cul
sy on
```

10.4 Generator

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

int main(int argc, char *argv[]) {
  cin.tie(0); ios_base::sync_with_stdio(0);
  if (argc < 2) {
    cout << "usage: " << argv[0] << " <seed>\n";
    exit(1);
  }
```

```
  srand(atoi(argv[1]));
  // use rand() for random value
}
```

10.5 Makefile

```
# p3: pypy3 -m py_compile
CXX = g++
CXXFLAGS = -Wall -Wconversion -Wfatal-errors -g -O2
-std=gnu++20 -fsanitize=address,undefined -Wshadow
-fno-omit-frame-pointer -Wno-unused-result
-Wno-sign-compare -Wno-char-subscripts #-fuse-ld=gold
```

10.6 C++ Template

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
using namespace std;
using namespace __gnu_pbds;
#define fi first
#define se second
#define rep(i,a,b) for (int i = (a); i < (b); ++i)
#define all(x) begin(x), end(x)
#define endl '\n'
#define int long long
using vi = vector<int>;
using ll = long long;
using ii = pair<int, int>;
using wv = pair<ll, int>;
typedef tree<int,null_type,less<int>,rb_tree_tag,
tree_order_statistics_node_update> indexed_set;

signed main() {
  cin.tie(0)->sync_with_stdio(0);
}
```

10.7 Stress

```
for (( I=0; I < 5; I++ )); do
  ./gen $I > z.in
  ./brute < z.in > expected.txt
  ./prog < z.in > output.txt
  if diff -u expected.txt output.txt; then : ; else
    echo "--> input (z.in):"; cat z.in
    echo "--> expected output:"; cat expected.txt
    echo "--> received output:"; cat output.txt
    break
  fi
  echo -n .
done
```