Some Tips for Coding in C++ in Competitive Programming

press space to continue

IO

Improve the speed of `std::cin`

```
ios::sync_with_stdio(false);
cin.tie(nullptr);
```

(Or in oneline)

```
cin.tie(nullptr)->sync_with_stdio(false);
```

Note: can not use `scanf` after this

Output with `endl` is slow, use `\n` instead

Output with given precision

```
cout << fixed << setprecision(20);
cout << 1.0 / 3.0 << '\n';
```

`std::is_sorted`

check whether is sorted in non-descending order

```cpp
if (std::is_sorted(v.begin(), v.end())) {
  // do something
}
```

or non-ascending order

```cpp
if (std::is_sorted(v.begin(), v.end(), greater<>())
  // do something
}
```

Example: https://codeforces.com/contest/1670/problem/A

`cond ? x : y` can return a reference

```
vector<int> a(n), even, odd;
for (auto& x : a) cin >> x;
for (auto& x : a) ((x & 1) ? odd : even).push_back(x);
```

Example: https://codeforces.com/contest/1638/problem/B

`std::max` can receive more than 2 arguments

NO:

```
int ans = max(a, max(b, c));
```

YES:

```
int ans = max({a, b, c});
```

Same for `std::min`

Example:
https://codeforces.com/contest/1244/problem/B

`std::minmax`

`std::minmax` returns a pair of the minimum and maximum

```
auto [mini, maxi] = minmax(u, v);
```

It is pretty useful if you want to store an unordered pair in a set

```
set<pair<int, int>> s;
s.insert(minmax{u, v});
```

Be careful: this code does not work as one might expect

```
tie(u, v) = minmax(u, v);
```

Example: https://codeforces.com/contest/1700/problem/E

reverse

```
string s;
cin >> s;
string t(s.rbegin(), s.rend());
```

Example:
https://codeforces.com/contest/1758/problem/A

check whether contains repeated element

```
vector a = {1, 1, 4, 5, 1, 4};
if (set(a.begin(), a.end()).size() == a.size()) {
  // do something
}
```

Example:
https://codeforces.com/contest/1742/problem/B

iterate each segments with the same element

```
vector a = {1, 1, 1, 2, 2, 3} // [1, 1, 1], [2, 2], [3]
int n = int(a.size());

for (int i = 0, j; i < n; i = j){
    for (j = i; j < n && a[j] == a[i];) j++;
    // do something
}
// [i, j): [0, 3) [3, 5) [5, 6)
```

Example: https://codeforces.com/contest/1430/problem/D

lambda

```cpp
// avoid naming escape
int limit = [&]{
  int l = 0, r = n;
  while (l < r) {
    int m = (l + r) / 2;
    if (check(m)){
      r = m;
    } else {
      l = m + 1;
    }
  }
  return l;
}();
```

```cpp
// can be used in `if`
if ([&]{
  for (auto& x : a) {
    if (check(x)) return false;
  }
}()){
  cout << "YES\n";
} else {
  cout << "NO\n";
}
```

```cpp
// use as goto
[&] {
  for (auto& row : a) {
    for (auto& x : row) {
      if (check(x)) return;
    }
  }
  cout << "-1\n";
}();
```

```cpp
// can be passed around
auto multiplier = [](int x) { return x * x; };
auto adder = [](int x) { return x + x; };

auto solve = [&](auto operator){
  // ...
}

solve(f), solve(g);
```

```cpp
// recursion
function<void(int)> dfs = [&](int node){
  for (auto& child : g[node]) dfs(child);
};
```

`std::iota`

```
vector a = {1, 1, 4, 5, 1, 4};
vector<int> order(a.size());
iota(order.begin(), order.end(), 0);

// handy when sorting
// and wanting to reserve the original array
sort(order.begin(), order.end(), [&](int i, int j){
  return a[i] < a[j];
});
```

Example:
https://codeforces.com/contest/1689/problem/E

`std::lower_bound`

```
set<int> s = {1, 2, 3, 4, 5};
```

`s.lower_bound(x);` and `lower_bound(s.begin(), s.end(), x);` are different!

The first one is O(log n), while the second one is O(n).

`std::accumulate`

```
vector<int> a(100000, 100000);
long long sum = accumulate(a.begin(), a.end(), 0ll)
```

Be careful with the type of the initial value!

Be careful with overflow!

`floor_div`

```
int floor_div(int a, int b) { return a / b - ((a ^ b) < 0 && a % b != 0); }
```

Be careful with negative numbers!

Example: Kickstart 2022 Round E, Problem D

print `__int128`

```cpp
using ll = long long;
void print(__int128 x) {
  constexpr ll P = 1e18;
  if (x < P) {
    cout << ll(x) << "\n";
  } else {
    cout << ll(x / P) << setw(18) << setfill('0') << ll(x % P) << "\n";
  }
}
```

BFS

`` `std::queue` `` is slow...

```
vector<int> q = {0}, nq; // init
for (int step = 0; q.size(); swap(q, nq), nq.clear(), step++) {
  for (auto& x : q) {
    for (auto& y : g[x]) {
      // do something
      nq.push_back(y);
    }
  }
}
```