C++ Cheatsheet

Soumitra Das

October 24, 2021

1 Initial Template

Uncomment line 5-9 if external library is needed.

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  #define FASTio ios::sync_with_stdio(false);cin.tie(
       NULL);
4  #define DECI fixed<<setprecision(5)
5  // #include <ext/pb_ds/assoc_container.hpp>
6  // #include <ext/pb_ds/tree_policy.hpp>
7  // using namespace __gnu_pbds;
8  // typedef tree<int,null_type,less<int>,rb_tree_tag,
       tree_order_statistics_node_update> indexed_set;
9  // typedef tree<int,null_type,less_equal<int>,
       rb_tree_tag,tree_order_statistics_node_update>
       indexed_multiset;
10 typedef long long ll;
11 typedef unsigned long long ull;
12 typedef long double ld;
13 typedef vector<int> vi;
14 typedef vector<vector<int>> vvi;
15 typedef pair<int,int> pii;
16 typedef priority_queue<int> pqi;
17 typedef deque<int> di;
18 #define pb(k) push_back(k)
19 #define mp(a,b) make_pair(a,b)
20 #define B begin();
21 #define E end();
22 #define nl cout<<"\n"
23 #define DB(x) {static int testInt=1000;if((testInt--)
       >0)cout<<"(LINE "<<__LINE__<<": VALUE "<<x<<")\t"
       ;}
24 #define LB {static int testIntx=0;if(testIntx<1000)
       cout<<"(LINE "<<__LINE__<<","<<testIntx+1<<")\t";
       else break;testIntx++;}
25 #define TA(arr) {int* lLe=(int*)(&arr+1);for(int* xTe=
       arr;xTe!=lLe;xTe++) cout<<*xTe<<" ";nl;}
26 #define nax 1000000007
27 /*********************************/
28
29 int main() {
30   FASTio
31   int t; cin >> t; while(t--) {
32     LB
33   }
34   return 0;
35 }
```

2 STL Library

2.1 Containers

vector

deque

list

forward_list

map

unordered_map

multimap

unordered_multimap

set

unordered_set

multiset

unordered_multiset

stack

queue

priority_queue

pair

tuple

tree

2.2 Algorithms

sort

reverse

max_element

min_element

accumulate

count

find

binary_search

lower_bound

upper_bound

next_permutation

prev_permutation

partition

stable_partition

rotate

min

max

swap

__gcd

__builtin_popcount

3 Algorithms

3.1 Fibonacci numbers

if F_n is the n'th Fibonacci number, where $F_0 = 0$ and $F_1 = 1$, then

$$F_{n+k} = F_k F_{n+1} + F_{k-1} F_n$$

for any $n, k \in \mathbb{N}$.

3.2 Geometric Transformation of points

Point (x, y, z) can be transformed by matrix multiplication

$$\begin{bmatrix} x & y & z & 1 \end{bmatrix} \times \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} = \begin{bmatrix} x' & y' & z' & 1 \end{bmatrix}$$

Where (x', y', z') is our answer. If we call the 4×4 matrix as X, then for shifting x by a co-ordinate, y by b and z by c co-ordinate,

$$X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ a & b & c & 1 \end{bmatrix}$$

Instead of shifting, for scaling

$$X = \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

And finally, for rotating θ degrees around the x axis following the right-hand rule (counter-clockwise direction)

$$X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

For 2D rotation of (x, y) by θ degree counterclockwise,

$$\begin{bmatrix} x & y \end{bmatrix} \times \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} x' & y' \end{bmatrix}$$

Where (x', y') is our answer.

3.3 Extended Euclidean Algorithm

Returns the gcd of a and b with $ax + by = \gcd(a, b)$.

```
int gcd(int a,int b,int& x,int& y){
    if (b==0){
        x=1;
        y=0;
        return a;
    }
    int u,v;
    int d=gcd(b,a%b,u,v);
    x=v;
    y=u-v*(a/b);
    return d;
}
```

3.4 Binary Search

Returns the index of x in array a.

```
int bin_search(int a[],int n,int x) {
  int l=0,r=n-1;
  while(l<=r){
    int k=(l+r)/2;
    if(a[k]==x){
      return k;
    }
    if(a[k]>x) r=k-1;
    else l=k+1;
  }
  return -1;
}
```

3.5 Processing All Subset

Processes subset of array a. Initially $k = 0$ and s empty.

```
void all_subset(int a[],int n,int k,vector<int> s) {
  if(k==n){
    // process subset
  }
  else{
    all_subset(a,n,k+1,s);
    s.push_back(a[k]);
    all_subset(a,n,k+1,s);
    s.pop_back();
  }
}
```

3.6 Processing All Permutation

Processes all permutation of array a, all element should be distinct. bm is an boolean array with length n, initially all element is $false$.

```
void all_permutation(int a[],int n,vector<int> p,int
    bm[]) {
  if(p.size()==n) {
    // process permutation
  }
  else {
    for(int i=0;i<n;i++) {
      if(bm[i]) continue;
      bm[i] = true;
      p.push_back(a[i]);
      all_permutation(a,n,p,bm);
      bm[i] = false;
      p.pop_back();
    }
  }
}
```

4 Useful Results

4.1 Finding directed path with fixed length

Create the adjacency matrix and raise it's power to k, cell (u, v) will give the number of distinct path with length k connecting vertex u and v (direction from u to v).