

C++ Cheatsheet

Soumitra Das

October 27, 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 $ax + by = \text{gcd}(a, b)$.

```
1 int gcd(int a, int b, int& x, int& y) {
2     if (b==0) {
3         x=1;
4         y=0;
5         return a;
6     }
7     int u, v;
8     int d=gcd(b, a%b, u, v);
9     x=v;
10    y=u-v*(a/b);
11    return d;
12 }
```

3.4 Binary Search

Returns the index of x in array a .

```
1 int bin_search(int a[], int n, int x) {
2     int l=0, r=n-1;
3     while(l<=r) {
4         int k=(l+r)/2;
5         if(a[k]==x) {
6             return k;
7         }
8         if(a[k]>x) r=k-1;
9         else l=k+1;
10    }
11    return -1;
12 }
```

3.5 Processing All Subset

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

```
1 void all_subset(int a[], int n, int k, vector<int> s) {
2     if(k==n) {
3         // process subset
4     }
5     else {
6         all_subset(a, n, k+1, s);
7         s.push_back(a[k]);
8         all_subset(a, n, k+1, s);
9         s.pop_back();
10    }
11 }
```

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*.

```
1 void all_permutation(int a[], int n, vector<int> p, int
2     bm[]) {
3     if(p.size()==n) {
4         // process permutation
5     }
6     else {
7         for(int i=0; i<n; i++) {
8             if(bm[i]) continue;
9             bm[i] = true;
10            p.push_back(a[i]);
11            all_permutation(a, n, p, bm);
12            bm[i] = false;
13            p.pop_back();
14        }
15 }
```

3.7 Miller Rabin Primality Test

Checks whether n prime or not, n must be in *int* range. Time complexity $O(\log n)$

```
1 typedef long long ll;
2
3 int binpower(int base, int e, int mod) {
4     int result=1;
5     base%=mod;
6     while(e) {
7         if(e&1)
8             result=(ll)result*base%mod;
9         base=(ll)base*base%mod;
10        e>>=1;
11    }
12    return result;
13 }
14 bool check_composite(int n, int a, int d, int s) {
15     int x=binpower(a, d, n);
16     if(x==1 || x==n-1) return false;
17     for(int r=1; r<s; r++) {
18         x=(ll)x*x%n;
19         if(x ==n-1) return false;
20     }
21     return true;
22 }
```

```

22 };
23 bool MillerRabin(int n) {
24     if (n<2) return false;
25     int r=0;
26     int d=n-1;
27     while((d&1)==0){
28         d>>=1;
29         r++;
30     }
31     for(int a:{2,3,5,7}){
32         if(n==a) return true;
33         if(check_composite(n,a,d,r)) return false;
34     }
35     return true;
36 }

```

4 Useful Results

4.1 matrix

The element *val* of this struct contains the value of the elements of the matrix, where $val[i][j]$ represents the value in i 'th row and j 'th column.

```

1 struct matrix {
2     vector<vector<int>> val;
3     matrix(int n) {
4         vector<int> temp(n,0);
5         for(int i=0;i<n;i++) val.push_back(temp);
6     }
7     matrix operator+(matrix x) {
8         matrix t_matrix(val.size());
9         for(int i=0;i<val.size();i++) for(int j=0;j<val.size();j++) {
10             t_matrix.val[i][j]=val[i][j]+x.val[i][j];
11         }
12         return t_matrix;
13     }
14     matrix operator-(matrix x) {
15         matrix t_matrix(val.size());
16         for(int i=0;i<val.size();i++) for(int j=0;j<val.size();j++) {
17             t_matrix.val[i][j]=val[i][j]-x.val[i][j];
18         }
19         return t_matrix;
20     }
21     matrix operator*(matrix x) {
22         matrix t_matrix(val.size());
23         for(int i=0;i<val.size();i++) for(int j=0;j<val.size();j++) {
24             int temp=0;
25             for(int k=0;k<val.size();k++) temp+=val[i][k]*(x.val[k][j]);
26             t_matrix.val[i][j]=temp;
27         }
28         return t_matrix;
29     }
30 };

```

4.2 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).