ACM-ICPC Team Reference Document

Chattogram Polytechnic Institute

1. C++ Template

```
// pragmas
// #pragma GCC optimize("Ofast")
// #pragma GCC target("avx,avx2,fma")
// #pragma GCC optimization ("unroll-loops")
// #pragma GCC optimization ("strict-overflow")

#include<bits/stdc++.h>
#include <ext/pb_ds/tree_policy.hpp>
#include <ext/pb_ds/assoc_container.hpp>

using namespace std;
using namespace __gnu_pbds;

# define    sp " "
# define    ff first
# define    ss second
# define    ll long long
# define     lld long double
# define    mp make_pair
# define    siz(x) (int)x.size()
# define    ull unsigned long long
# define    all(v) v.begin(),v.end()
# define    allr(v) v.rbegin(),v.rend()
# define    torad(x) ((x) * ((2*acos(0))/180.0))
# define    todeg(x) ((x) * (180.0/(2*acos(0))))
# define     nl "\n"
# define     print(x) cout<<x<<nl

constexpr ll mod=1000000000+7;
constexpr ll INF=LLONG_MAX;
// constexpr double PI= acos(-1);
constexpr double eps=1e-9;

#define fastio()
ios_base::sync_with_stdio(false);cin.tie(NULL);cout.tie(
NULL)
# define fraction(a) cout.unsetf(ios::floatfield);
cout.precision(a); cout.setf(ios::fixed,ios::floatfield);
# define ordered_set tree<int, null_type,less<int>,
rb_tree_tag,tree_order_statistics_node_update>

#define debug(x) cout << #x<<" "; _print(x); cout << nl
#define ff first
#define ss second
#define pi
3.14159265358979323846264338327950288419 7
void _print(ll t) {cout << t;}
void _print(int t) {cout << t;}
void _print(string t) {cout << t;}
void _print(char t) {cout << t;}
void _print(lld t) {cout << t;}
void _print(double t) {cout << t;}
void _print(ull t) {cout << t;}
template <class T, class V> void _print(pair <T, V> p);
template <class T> void _print(vector <T> v);
template <class T> void _print(set <T> v);
template <class T, class V> void _print(map <T, V> v);
template <class T> void _print(multiset <T> v);
template <class T, class V> void _print(pair <T, V> p)
{cout << "{"; _print(p.ff); cout << ","; _print(p.ss); cout
<< "}";}
template <class T> void _print(vector <T> v) {cout << "[
"; for (T i : v) {_print(i); cout << " ";} cout << "]";}
template <class T> void _print(set <T> v) {cout << "[ ";
for (T i : v) {_print(i); cout << " ";} cout << "]";}
template <class T> void _print(multiset <T> v) {cout <<
"[ "; for (T i : v) {_print(i); cout << " ";} cout << "]";}
template <class T, class V> void _print(map <T, V> v)
{cout << "[ "; for (auto i : v);}


signed main(){
   fastio();
   int T;cin>>T;

   for(int TT=1;TT<=T;TT++){
   }
}
```

2. Python Template

```
import sys
sys.stdout = open('./output.txt', 'w')
sys.stdin = open('./input.txt', 'r')

input=sys.stdin.readline

for T in range(int(input())):
   n=int(input())
   a=list(map(str,input().split()))
   print(a)
```

3. CPP config

```
{
   "version": "2.0.0",
   "tasks": [
      {
         "label": "compile",
         "type": "shell",
         "command": "g++",
         "args": [
            "-std=c++17",
            "-o",
            "${fileBasenameNoExtension}",
            "${file}"
         ],
         "group": {
            "kind": "build",
            "isDefault": false
         }
      },
      {
         "label": "compile and run",
         "type": "shell",
         "command": "g++ -std=c++17 -o
${fileBasenameNoExtension} ${file} &&
./${fileBasenameNoExtension} < input.txt > output.txt",
            "group": {
                "kind": "build",
                "isDefault": true
            }
        }
    ]
}
```

4. Bellman Ford

```
vector<int> bellmanFord(
int V,
vector<vector<int>> &edges,
int src){
    vector<int> ans(V, 1e8);
    ans[src] = 0;
    for (int i = 1; i <= V; i++){
        for (vector<int> temp : edges){
            int u = temp[0], v = temp[1], w = temp[2];
            if (ans[u] != 1e8 and ans[v] > ans[u] + w){
                if (i == V){
                    return {-1};
                }
                ans[v] = ans[u] + w;
            }
        }
    }

    return ans;
}
```

6. Floyd Warshall

```
// mat is matrix where edges[i][j] repr weight of
i->j
void shortestDistance(vector<vector<int>> &mat){
    int n = mat.size();
    for (int i = 0; i < n; i++){
        for (int a = 0; a < n; a++){
            for (int b = 0; b < n; b++){
                if (mat[a][i] != -1 and mat[i][b] != -1 and
(mat[a][i] + mat[i][b] < mat[a][b] or mat[a][b] ==
-1)){
                    mat[a][b] = mat[a][i] + mat[i][b];
                }
            }
        }
    }
}
```

5. Dijstras

```
vector<int> dijkstra(
vector<vector<pair<int, int>>> &adj, int src){
    int n = adj.size();

    vector<int> ans(n, 1e9);

    // set<pair<int,int>>St;
    priority_queue<pair<int, int>, vector<pair<int,
int>>, greater<pair<int, int>>> pq;
    pq.push({0, src});

    // set<int>vis;

    while (pq.size() > 0){
        pair<int, int> cur = pq.top();
        pq.pop();

        // if(vis.find(cur.second)!=vis.end()){
        //     continue;
        // }
        // vis.insert(cur.second);

        if (cur.first >= ans[cur.second]){
            continue;
        }
        ans[cur.second] = cur.first;

        for (pair<int, int> c : adj[cur.second]){
            int cost = cur.first + c.second;
            if (cost < ans[c.first]){
                pq.push({cost, c.first});
            }
        }
    }

    return ans;
}
```

7. MST Prims

```
int spanningTree(
int V,
vector<vector<int>> adj[]){
    priority_queue<pair<int, int>, vector<pair<int,
int>>, greater<pair<int, int>>> pq; // weight,node

    pq.push({0, 0});

    vector<int> vis(V, 0);

    int cnt = 0;

    while (pq.size() > 0){
        auto cur = pq.top();
        pq.pop();

        int node = cur.second;

        if (vis[node]){
            continue;
        }

        vis[node] = 1;
        cnt += cur.first;

        for (auto brr : adj[node]){
            if (vis[brr[0]] == 0){
                pq.push({brr[1], brr[0]});
            }
        }
    }

    return cnt;
}
```

8. Z-function

```
vector<int> z_function(string s) {
    int n = s.size();
    vector<int> z(n);
    int l = 0, r = 0;
    for(int i = 1; i < n; i++) {
        if(i < r) {
            z[i] = min(r - i, z[i - l]);
        }
        while(i + z[i] < n && s[z[i]] == s[i + z[i]]) {
            z[i]++;
        }
        if(i + z[i] > r) {
            l = i;
            r = i + z[i];
        }
    }
    return z;
}
signed main(){
    string a,b;cin>>a>>b;
    //finding b inside a.
    b+="$"
    b+=a;
    z_function(b);
}
```

9. Toposort - khan's algo

```
vector<int> topologicalSort(
vector<vector<int>> &adj){
    int n = adj.size();

    vector<int> ind(n, 0);

    for (int i = 0; i < n; i++){
        for (int cur : adj[i]){
            ind[cur]++;
        }
    }

    queue<int> q;
    for (int i = 0; i < n; i++){
        if (ind[i] == 0){
            q.push(i);
        }
    }

    vector<int> ans;

    while (q.size() > 0){
        int cur = q.front();
        q.pop();
        ans.push_back(cur);

        for (int i : adj[cur]){
            ind[i]--;
            if (ind[i] == 0){
                q.push(i);
            }
        }
    }

    return ans;
}
```

10. MST_Kruskals

```
class DSU
{
   int *parent;
   int *rank;
   int *size;

public:
   DSU(int n){
      parent = new int[n];
      rank = new int[n];

      for (int i = 0; i < n; i++){
         parent[i] = -1;
         rank[i] = 1;
         size[i] = 1;
      }
   }

   // uses path compression - O(1)
   int find(int i){
      if (parent[i] == -1){
         return i;
      }
      // when we are connecting two nodes we are
connecting their parents.
      return parent[i] = find(parent[i]);
   }

   bool uniteByRank(int x, int y){
      int s1 = find(x);
      int s2 = find(y);

      if (s1 != s2){

         if (rank[s1] > rank[s2]){
            parent[s2] = s1;
         }
         else if (rank[s1] < rank[s2]){
            parent[s1] = s2;
         }
         else{
            parent[s2] = s1;
            rank[s1]++;
         }

         return true;
      }
      return false;
   }
bool uniteBySize(int x, int y){
      int s1 = find(x);
      int s2 = find(y);

      if (s1 != s2){

         if (size[s1] > size[s2]){
            parent[s2] = s1;
            size[s1] += size[s2];
         }
         else{
            parent[s1] = s2;
            size[s2] += size[s1];
         }

         return true;
      }
      return false;
   }
};

class Solution{
public:
   int spanningTree(int V, vector<vector<int>> adj[]){

      DSU dsu(V);

      vector<vector<int>> arr;

      for (int i = 0; i < V; i++){
         for (auto brr : adj[i]){
            vector<int> temp;
            temp.push_back(brr[1]);
            temp.push_back(i);
            temp.push_back(brr[0]);
            arr.push_back(temp);
         }
      }
      sort(arr.begin(), arr.end());

      int cnt = 0;

      // connecting more than v-1 edges will ultimately
result in a cycle so no need to check separately.
      for (int i = 0; i < arr.size(); i++){
         int w = arr[i][0], a = arr[i][1], b = arr[i][2];

         if (dsu.find(a) != dsu.find(b)){
            dsu.uniteBySize(a, b);
            cnt += w;
         }
      }

      return cnt;
   };
};
```

11. Manacher's algo

```
int manacher_algo(string &s){

   int n=s.size();
   int P[n];
   memset(P,0,sizeof(P));
   int C=0,R=0;
   int mx=0;
   for(int i=1;i<s.size()-1;i++){
       // the concept of mirroring is main thing that
reduce the time complexity from O(N^2) -> O(N)
       // as we don't have to calculate the length we
have calculated before.
       if(i<R){
           int mirr=(2*C)-i;
           //if i is center then we can at max expand it
till right boundary,it can happen that
           //mirror of i expand beyond the current
boundary range from left so thats why we are
checking it.
           P[i]=min(R-i,P[mirr]);
       }

       //check if we can expand further
       while(s[i+(1+P[i])]==s[i-(1+P[i])]){
           P[i]++;
       }

       //if we can only expand beyond cur right
boundary only then change it.
       if((i+P[i])>R){
           C=i;
           R=i+P[i];
       }

       mx=max(mx,P[i]);
   }

   return mx;
}
signed main(){
   string s="aaaa";
   string nw="$";
   for(int i=0;i<s.size();i++){
       nw+="#";
       nw+=s[i];
   }
   nw+="#";
   nw+="@";
print(manacher_algo(nw));
}
```

12. Rabin karp

```
vector<int> rabin_karp(string &p,string &t){

   int prime=31;
   int mod=1e9+9;

   int n=p.size(),m=t.size();
   vector<long long>p_pow(max(n,m));
   p_pow[0]=1;

   for(int i=1;i<p_pow.size();i++){
       p_pow[i]=(p_pow[i-1]*prime)%mod;
   }

   vector<long long>text_hash(m+1,0);
   for(int i=0;i<m;i++){

text_hash[i+1]=(text_hash[i]+((t[i]-'a'+1)*p_pow[i]))%m
od;
   }

   long long pattern_hash=0;
   for(int i=0;i<n;i++){

pattern_hash=(pattern_hash+((p[i]-'a'+1)*p_pow[i]))%m
od;
   }

   vector<int>occurances;

   for(int i=0;i+n<=m;i++){
       long long
hash_to_comp=(text_hash[i+n]+mod-text_hash[i])%mod
;
       // we are multiply and moduloing pattern hash again
to handle the index alignment issue.

if(hash_to_comp==(pattern_hash*p_pow[i]%mod)){
           occurances.push_back(i);
       }
   }

   return occurances;

}
```

13. ModInverse

```
const long long mod=1e9+7;
long long binpow(long long a, long long b, long
long m=mod) {
    a %= m;
    long long res = 1;
    while (b > 0) {
        if (b & 1)
            res = res * a % m;
        a = a * a % m;
        b >>= 1;
    }
    return res;
}
long long modInverse(
long long a,long long p=mod){
    //p must be prime.
    return binpow(a,p-2,p);
}
```

14. NCR

```
const int mod=1e9+7;
int binpow(int a, int b, int m=mod) {
    a %= m;
    int res = 1;
    while (b > 0) {
        if (b & 1)
            res = res * a % m;
        a = a * a % m;
        b >>= 1;
    }
    return res;
}
int modInverse(int a,int p=mod){
    //p must be prime.
    return binpow(a,p-2,p);
}
int nCr(int n,int r,int p=mod){
    if(n<r)return 0;
    if(r==0)return 1;
    vector<int>fact(n+1,0);
    fact[0]=1;
    for(int i=1;i<=n;i++){
        fact[i]=fact[i-1]*i;
        fact[i]%=p;
    }
    // fact[n]/(fact[r]*fact[n-r]);
    return
(fact[n]*modInverse(fact[r],p)%p*modInverse(fact
[n-r],p)%p)%p;
}
```

15. Binomial Cloef

//gives the number of ways we can choose a subset of k elements from a set of n elements.

```
int binomialCoeff(int n,int k){
    if(k > n){
        return 0;
    }else if(k == 0 || k == n){
        return 1;
    }

    return (binomialCoeff(n-1,k-1) +
binomialCoeff(n-1,k));
}

int main(){
    cout << "Binomial Coeff of 5,3 is : " <<
binomialCoeff(5,3);
    return 0;
}
```

16. Wheel Factorization

```
vector<long long> trial_division2(long long n) {
    vector<long long> factorization;
    while (n % 2 == 0) {
        factorization.push_back(2);
        n /= 2;
    }
    for (long long d = 3; d * d <= n; d += 2) {
        while (n % d == 0) {
            factorization.push_back(d);
            n /= d;
        }
    }
    if (n > 1)
        factorization.push_back(n);
    return factorization;
}
```

17. Prime Factorization

```
//Sqrt(N);
vector<long long> trial_division1(long long n) {
    vector<long long> factorization;
    for (long long d = 2; d * d <= n; d++) {
        while (n % d == 0) {
            factorization.push_back(d);
            n /= d;
        }
    }
    if (n > 1)
        factorization.push_back(n);
    return factorization;
}
```

18. Prime factorization

```
const ll N=(ll)1e7+10;
vector<ll>primes(N,1);
vector<ll>lp(N,0),hp(N,0);
void seive(){
   primes[0]=primes[1]=0;
   for(ll i=2;i<N;i++){
      if(primes[i]==1){
         lp[i]=hp[i]=i;
         for(ll j=i*i;j<N;j+=i){
            primes[j]=0;
            if(lp[j]==0)lp[j]=i;
            hp[j]=i;
         }
      }
   }
}
void solve(){
   int n;cin>>n;
   unordered_map<int,int>factors;
   //Log(N)
   while(n>1){
      int prime_factor=hp[n];
      while(n%prime_factor==0){
         n/=prime_factor;
         factors[prime_factor]++;
      }
   }
   for(auto it:factors){
      cout<<it.first<<" = "<<it.second<<nl;
   }
}
```

19.Floor Mod

```
//for modding negative numbers.
int floorMod(int a, int mod){
   int q = (int)floor((double)a / mod);
   return a - mod * q;
}
```

20.Custom Ceil

```
ll ceil_div(ll a,ll b){
   return (a+b-1)/b;
}
```

21.LCM

```
long long lcm(int a, int b){
   return (a / __gcd(a, b)) * b;
}
```

22.Formulas

- nPr=!n/!(n-r)
- nCr= nPr/!r = !n/(!(n-r)*!(r))
- nC(r-1)= !n/(!(n-r+1)*!(r-1))
- nCr=nC(r-1)*((n-r+1)/r))
- nC0+nC1+nC2...+nCN=2^N(if we can choose any number of items)
- সমান্তর ধারার সূত্র
  - প্রথম পদ a, সাধারণ অন্তর d হলে,
  - r তম পদ = a+(r-1)d
  - n সংখ্যক পদের সমষ্টি (S) = n{2a+(n-1)d}/2
  - n সংখ্যক স্বাভাবিক সংখ্যার সমষ্টি = n(n+1)/2
  - n সংখ্যক স্বাভাবিক সংখ্যার বর্গের সমষ্টি = n(n+1)(2n+1)/6
  - n সংখ্যক স্বাভাবিক সংখ্যার ঘন এর সমষ্টি = {n(n+1)/2}^2
  - গড় = শেষ পদ + ১ম পদ / ২
- গুণোত্তর ধারার সূত্র
  - ধারার প্রথম পদ a, সাধার অনুপাত r এবং পদ সংখ্যা n হলে,
  - n তম পদ = ar^n-1
  - n সংখ্যক পদের সমষ্টি, S = a(r^n-1)/(r-1) যেখানে r >1
  - আবার, s = a(1-r)^n/1-r যেখানে r <1
  - a, b এর গুণোত্তর মধ্যক G = √ab

$$(x+y)^0 = 1,$$
$$(x+y)^1 = x+y,$$
$$(x+y)^2 = x^2+2xy+y^2,$$
$$(x+y)^3 = x^3+3x^2y+3xy^2+y^3,$$
$$(x+y)^4 = x^4+4x^3y+6x^2y^2+4xy^3+y^4,$$
$$(x+y)^5 = x^5+5x^4y+10x^3y^2+10x^2y^3+5xy^4+y^5,$$
$$(x+y)^6 = x^6+6x^5y+15x^4y^2+20x^3y^3+15x^2y^4+6xy^5+y^6,$$
$$(x+y)^7 = x^7+7x^6y+21x^5y^2+35x^4y^3+35x^3y^4+21x^2y^5+7xy^6+y^7,$$
$$(x+y)^8 = x^8+8x^7y+28x^6y^2+56x^5y^3+70x^4y^4+56x^3y^5+28x^2y^6+8xy^7+y^8.$$

$$\sum_{k=0}^{n} k^2 = \frac{n(n+1)(2n+1)}{6},$$
$$\sum_{k=0}^{n} k^3 = \frac{n^2(n+1)^2}{4}.$$

Harmonic series

For positive integers n, the nth ***harmonic number*** is

$$\begin{aligned} H_n &= 1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\cdots+\frac{1}{n} \\ &= \sum_{k=1}^{n}\frac{1}{k} \\ &= \ln n + O(1). \end{aligned}$$

23.Radix sort

```
int getMax(vector<int>&arr){
    int mx=0;
    for(int cur:arr){
        mx=max(mx,cur);
    }
    return mx;
}

void counting_sort(vector<int>&arr,int exp){
    int n=arr.size();

    int count_arr[10]={0};
    for(int i=0;i<n;i++){
        count_arr[(arr[i]/exp)%10]++;
    }

    for(int i=1;i<10;i++){
        count_arr[i]+=count_arr[i-1];
    }

    vector<int>output(n);
    for(int i=n-1;i>=0;i--){
        int digit=(arr[i]/exp)%10;
        count_arr[digit]-=1;
        int ind=count_arr[digit];
        output[ind]=arr[i];
    }

    for(int i=0;i<n;i++){
        arr[i]=output[i];
    }
}

void radix_sort(vector<int>&arr){

    int mx=getMax(arr);

    for(int exp=1;(mx/exp)>0;exp*=10){
        counting_sort(arr,exp);
    }
}
```

24.Python config

```
{
    "version": "2.0.0",
    "tasks": [
        {
            "label": "Run Python file",
            "type": "shell",
            "command": "python3",  // Or "python"
depending on your OS
            "args": [
                "${file}"  // This will run the current file open
in the editor
            ],
            "group": {
                "kind": "build",
                "isDefault": true
            },
            "problemMatcher": [],
            "detail": "Runs the currently opened Python
file"
        }
    ]
}
```

25. Bucket sort

```
// This can be any sorting algo.
void insertionSort(vector<float>& bucket) {
    for (int i = 1; i < bucket.size(); ++i) {
        float key = bucket[i];
        int j = i - 1;
        while (j >= 0 && bucket[j] > key) {
            bucket[j + 1] = bucket[j];
            j--;
        }
        bucket[j + 1] = key;
    }
}

// Function to sort arr[] of size n using bucket sort
void bucketSort(float arr[], int n) {
    // 1) Create n empty buckets
    vector<float> b[n];

    // 2) Put array elements in different buckets
    for (int i = 0; i < n; i++) {
        int bi = n * arr[i];
        b[bi].push_back(arr[i]);
    }

    // 3) Sort individual buckets using insertion sort
    for (int i = 0; i < n; i++) {
        insertionSort(b[i]);
    }

    // 4) Concatenate all buckets into arr[]
    int index = 0;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < b[i].size(); j++) {
            arr[index++] = b[i][j];
        }
    }
}
```

26.Bucket sort

```
void bucketSort(vector<int>& arr) {
   int n = arr.size();
   if (n <= 0) return;

   // Find the minimum and maximum values in the
array
   int minValue = *min_element(arr.begin(),
arr.end());
   int maxValue = *max_element(arr.begin(),
arr.end());

   // Calculate the range of values
   int range = maxValue - minValue + 1;

   // Create `range` number of buckets
   vector<vector<int>> buckets(range);

   // Map array elements to buckets
   for (int i = 0; i < n; i++) {
      int bucketIndex = arr[i] - minValue; // Adjust
index for negative numbers
      buckets[bucketIndex].push_back(arr[i]);
   }

   // Concatenate all buckets into the original array
   int index = 0;
   for (int i = 0; i < range; i++) {
      for (int val : buckets[i]) {
         arr[index++] = val;
      }
   }
}
```

27.Merge sort

```
void merge(int low,int mid,int
high,vector<int>&arr){

   vector<int>temp;

   int l=low,r=mid+1;
   while(l<=mid and r<=high){
      if(arr[l]<=arr[r]){
         temp.push_back(arr[l]);
         l++;
      }else{
         temp.push_back(arr[r]);
         r++;
      }
   }
while(l<=mid){
      temp.push_back(arr[l]);
      l++;
   }
   while(r<=high){
      temp.push_back(arr[r]);
      r++;
   }


   for(int i=low,j=0;i<=high;i++,j++){
      arr[i]=temp[j];
   }
}

void merge_sort(int low,int high,vector<int>&arr){

   if(low>=high)return;
   int mid=(low+high)/2;
   merge_sort(low,mid,arr);
   merge_sort(mid+1,high,arr);
   merge(low,mid,high,arr);
}
```

28. Quick sort

```
int partition(vector<int>&nums,int low,int high){
   int pivot=nums[low];
   int l=low+1,r=high;
   while(l<=r){

      if(nums[l]>pivot and nums[r]<pivot){
         swap(nums[l],nums[r]);
         l++,r--;
      }
      if(nums[l]<=pivot){
         l++;
      }
      if(nums[r]>=pivot){
         r--;
      }
   }
   swap(nums[low],nums[r]);
   return r;
}


void qs(vector<int>&arr,int low,int high){
   if(low<high){
      int pt=partition(arr,low,high);
      qs(arr,low,pt-1);
      qs(arr,pt+1,high);
   }
}
```