C++ Competitive Programming Cheat Sheet

# 1. Basic Syntax and I/O

#include <iostream>  
using namespace std;  
  
int main() {  
 int a;  
 cin >> a;  
 cout << "Value of a: " << a << endl;  
 return 0;  
}  
  
// Faster I/O  
ios\_base::sync\_with\_stdio(false);  
cin.tie(NULL);

# 2. Data Structures

## Arrays

int arr[100]; // Declaration

## Vectors

#include <vector>  
vector<int> v; // Dynamic array  
v.push\_back(1); // Add element  
v.pop\_back(); // Remove element

## Pairs

#include <utility>  
pair<int, int> p = {1, 2};  
cout << p.first << " " << p.second << endl;

# 3. Algorithms

## Sorting

#include <algorithm>  
vector<int> v = {4, 2, 3, 1};  
sort(v.begin(), v.end()); // Sort vector

## Binary Search

#include <algorithm>  
int idx = lower\_bound(v.begin(), v.end(), 3) - v.begin();

# 4. STL (Standard Template Library)

## Set

#include <set>  
set<int> s;  
s.insert(1);  
s.erase(1);

## Map

#include <map>  
map<int, int> m;  
m[1] = 2;  
cout << m[1] << endl;

# 5. Dynamic Programming

## Fibonacci Sequence

int fib(int n) {  
 int dp[n+1];  
 dp[0] = dp[1] = 1;  
 for(int i = 2; i <= n; ++i)  
 dp[i] = dp[i-1] + dp[i-2];  
 return dp[n];  
}

# 6. Graph Algorithms

## BFS

#include <queue>  
vector<int> adj[100];  
bool visited[100];  
  
void bfs(int start) {  
 queue<int> q;  
 q.push(start);  
 visited[start] = true;  
 while(!q.empty()) {  
 int node = q.front();  
 q.pop();  
 for(int neighbor : adj[node]) {  
 if(!visited[neighbor]) {  
 q.push(neighbor);  
 visited[neighbor] = true;  
 }  
 }  
 }  
}

# 7. Tree Algorithms

## DFS

void dfs(int node) {  
 visited[node] = true;  
 for(int neighbor : adj[node]) {  
 if(!visited[neighbor])  
 dfs(neighbor);  
 }  
}

# 8. Mathematics

## GCD

int gcd(int a, int b) {  
 return b == 0 ? a : gcd(b, a % b);  
}

# 9. Bit Manipulation

## Check if a Number is Power of 2

bool isPowerOfTwo(int n) {  
 return (n && !(n & (n - 1)));  
}

# 10. String Algorithms

## KMP Algorithm

vector<int> computeLPSArray(string pat) {  
 int M = pat.length();  
 vector<int> lps(M);  
 int len = 0;  
 lps[0] = 0;  
 int i = 1;  
 while (i < M) {  
 if (pat[i] == pat[len]) {  
 len++;  
 lps[i] = len;  
 i++;  
 } else {  
 if (len != 0) {  
 len = lps[len - 1];  
 } else {  
 lps[i] = 0;  
 i++;  
 }  
 }  
 }  
 return lps;  
}

# 11. Miscellaneous Tips

- Practice problem-solving regularly on platforms like Codeforces, LeetCode, and HackerRank.  
- Learn and use efficient data structures and algorithms.  
- Participate in coding contests to improve speed and accuracy.