
Headstart
Coding Club, IIT Guwahati
< WEEK 2 >

CONTENTS

Note:

- The articles mentioned contain theoretical concepts and should be read by everyone. The codes might be given in C++, but we can correspondingly develop the python code from the theory given.
- The articles from USACO Guide can also be accessed in both python and C++ by changing the language using an option on the webpage
- Python users can checkout our **GitHub repo** which contains the code of some useful functions to be covered ahead.

STL

(Expected Time: 1-2 days)

The **Standard Template Library (STL**) is a set of C++ template classes to provide common programming **data structures** and functions such as lists, stacks, arrays, etc.

At the core of the C++ Standard Template Library are the following three well-structured components – **containers, algorithms, and iterators**. Basic knowledge of STL is quite handy, and in fact, necessary for competitive programming.

Resources:

- **Standard Template Library in C++ - Great Learning**
- **The C++ Standard Template Library (STL) - GeeksforGeeks**
- **Watch this video to learn the basics of STL**
- **C++ STL - Luv Competitive Programming** (in Hindi)
- Read Ch-4 of **CPH** (pg 35-pg 43)

Frequently used containers:

- **Array**: Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value. Vector is used more frequently compared to array since the former is dynamic in nature. However, **higher dimensional arrays** are simpler to declare than vectors.

- **Vector**: One of the most powerful and frequently used STL containers. Vectors are the same as dynamic arrays with the ability to resize themselves automatically when an element is inserted or deleted. Study these functions only:
  push_back(), pop_back(), sort(), upper_bound() , lower_bound()* , begin(), end(), rbegin(), size(), resize(), accumulate(), binary_search()* clear() and find()*.

- **Pair:** Pair is a container that contains two values. It is used to combine two values and associate them even if they are of different types. The first and second values of a pair 'p' can be accessed using *p.first* and *p.second* .

- **Set**: A container that keeps a **unique** copy of every element in **sorted** order. Study these functions only:
  insert(), erase(), begin(), end(), rbegin(), size(), lower_bound(), upper_bound(), find() and clear()

- **Multiset**: Similar to a set, but it can also store multiple copies of an element.

- **Map**: Maps are containers that store elements in a mapped fashion. Each element has a key value and a mapped value. Study these functions only:
  insert(), erase(), begin(), end(), rbegin(), find(), clear(), size()
  lower_bound(), upper_bound(), and the operator []

*The use of these containers and their functions will be explained in the upcoming weeks

Not-so-frequently used containers:

- **Stack**: Stacks are a type of container adaptor that operate in a Last In First Out (**LIFO**) type of arrangement. Elements are added at one end (top) and are removed from that end only.

- **Queue**: Queues are a type of container that operates in a First In First Out (**FIFO**) type of arrangement, unlike Stack. Elements are inserted at the back (end) and are deleted from the front.

- **Priority Queue**: Priority queues are queues such that the first element is either the greatest or the smallest of all elements in the queue and elements are in nonincreasing order.

- **Deque**: Double-ended queues are a special case of queues where insertion and deletion operations are possible at both ends

- **MultiMap**: Multimap is similar to a map with the addition that multiple elements can have the same keys

STL EQUIVALENT IN PYTHON

Frequently used containers:

- **List**: Lists are used to store multiple values(they may not be of the same data type) in a single variable, instead of declaring separate variables for each value.

- **Dictionary**: It is equivalent to an unordered map of C++ STL. Maps are containers that store elements in a **mapped** fashion. Each element has a key value and a mapped value. It does not store elements in a sorted manner like C++ Map.

- **Sets**: A set is an unordered collection of items. Every set element is unique (no duplicates). Sets can also be used to perform mathematical set operations like union, intersection, etc.

Some more containers:

- **Deque**: It can be used to implement **queue** and **deque** similar to those given in C++ STL.

- **Heap**: The property of this data structure is that each time the smallest heap element is popped. It is the python equivalent of a priority queue.

Frequently used Modules:

- **Math**: It contains all the basic Math operations ranging from ceil, and floor to factorial, pow, etc.

- **Sys**: It's generally used for some specific functions revolving around the std input and output. Eg: *sys.stdin.readline()* (which takes input faster than the normal input function), *sys.stdout.flush()* (used in interactive problems) , *sys.setrecursionlimit(int)* (to increase the recursion limit) , etc

- **Bisect**: Has 2 functions, *bisect_right(list, key)* which finds and returns the index of the minimum element in the list greater than the key, and *bisect_left(list, key)* which finds and returns the index of the maximum element in the list lesser than key both in *O(log(len(list))* complexity.

- **Random**: Has its use in some of the probability-based questions and in question generation. Some common functions of random:
  Random.randint(a,b) which generates a random integer between a and b .
  Random.shuffle(list) returns a shuffled version of a given list in O(len(list)) .

Following are some practice problems:-

- **Distinct Numbers (C++Solution) (Python-soln)**
- **Remove Duplicates (C++Solution) (Python-soln)**
- **Registration System (C++Solution) (Python-soln)**
- **T-Shirt Buying? (C++Solution) (Python-soln)**
- **Subarray Sums (C++Solution)(Python-soln)**
- **Scope (C++Solution)(Python-soln)**
- **Movie Festival (C++Solution) (Python-soln)**

SORTING

Sorting means arranging elements (in a data structure) in a particular order.
There are several sorting algorithms. Some basic sorting algorithms are as following to give you an idea of sorting techniques:

1. **Insertion Sort**
   a. **Article**
   b. **Video**

2. **Merge Sort**
   a. **Article**
   b. **Video**

But, why so many algorithms!!? **Read here**

While doing CP problems, we don't write any sorting algorithm. C++ STL provides a function to sort elements in any order in **O(N log N)** time.

SYNTAX - C++

Array : *sort(arr, arr+arr.size());*
Vector : *sort(vect.begin(), vect.end());*

Formally, *sort(it1, it2)* will sort elements in ascending order in the range [it1, it2) .

Custom sort (Ignore *qsort*).

Further Read (Optional)

- **Sorting a vector of pairs - 1**
- **Sorting a vector of pairs - 2**

SYNTAX - Python

List : *list.sort(reverse = True/False)*
Dictionary/Set : *sorted(iterable,reverse=True/False)*

Sorting a list of pairs can be done by the normal sort function itself. Let's say a list, a= [(4,5),(2,3),(2,1),(4,4)] then simply calling ***a.sort()*** will give a = [(2,1),(2,3),(4,4),(4,5)] (sorting based on the first element of each tuple). You can learn more about **custom sort** using lambda function.

Practice Problems

- **Smallest Pair**
- **Bear and Extra Number**
- **Incinerate**
- **Swiss System Tournament**
- **Stick Lengths**
- **Divisibility by 2^n**
- **Bombs (tough)**

Additional Questions

- **Nested Ranges Check**
- **Blue Red Permutation**
- **Weight of the System of Nested Segment**
- **Delete Them**

GREEDY ALGORITHM

A greedy algorithm, as the name suggests, always makes the choice that seems to be the best at that moment. This means that it makes a locally-optimal choice in the hope that this choice will lead to a globally-optimal solution. Problem list is larger because greedy can be learned only through problem solving.

Resources

- **Watch this lecture on Algomaniacs**
- **How to prove greedy algorithm is correct (not so important for CP)**

Practice Problems

- **Different Differences**
- **Kathmandu**
- **Berland Music**
- **Closing the Gap**
- **Paprika and Permutation**
- **Tea with Tangarines**
- **Zero Array**
- **Make it Good**
- **Rudolf and the Another Competition**
- **The Vowel Matrix**
- **Negative Prefixes**
- **M-arrays**
- **Knapsack**
- **Particles (tough)**

Additional Questions

- **Problems by Algomaniacs**

COMBINATORICS

(Expected Time: 0-1 days)

Remember your JEE days? Well guess what... the stuff you have learnt for JEE doesn't go for waste, it has its own importance in CP too!

CALCULATING

Naïve method

You must have learnt the formula ${}^{n}C_{r} = {}^{n-1}C_{r} + {}^{n-1}C_{r-1}$. This essentially means that you could recursively calculate the value of ${}^{n}C_{r}$ through the previously calculated ${}^{n-1}C_{r}$ and ${}^{n-1}C_{r-1}$. The code for this part can be accessed in the **USACO** guide. However, the complexity of this naïve approach is **O(2^n)** and this can be **optimized** to **O(n^2)** using a technique called **dynamic programing** that we will go through in detail in the next week.

Optimal method

You must have also learnt an alternate formula for nCr which is **n!/(r!*(n-r)!)**. This can be used to find out the binomial coefficients too.

Also we know that factorials tend to be pretty large which can overflow the integer size (and in python even though you can store integers of very long size, it takes up a lot of time giving TLE), so we generally we calculate it modulo some number (which is generally prime). This can be done in **O(n+log(mod))** complexity by precomputing the factorials and inverse factorials individually and then using them in the problem as required. Code is available at **USACO**.

Practice Problems

- **Binomial Coefficients**
- **Creating Strings II**
- **Binary Strings are Fun**
- **Close Tuples**

Additional Questions

- **USACO Problemset**

Common Errors

Even if you try your best to write a flawless piece of code, you may still encounter some error. Don't worry it is a part of the journey. Refer this **blog** on codeforces which almost covers all possible types of common mistakes.

Some common verdicts and errors on online judges:

- **Compilation Error:** It occurs whenever your compiler fails to recognize your code as a legitimate one. Possible reasons are many like typos, syntax error etc.
- **Time Limit Exceeded (TLE):** It is one of the most common error verdict one may get. Possible reasons are missing terminating condition, writing wrong semantics, using for loop iterations more than 10e7 times.
- **Memory Limit Exceeded (MLE):** Again it is also a very common error verdict. Whenever you try to store objects like int or char more than 10e7 in an array or vector, you may face this.
- **Wrong Answer (WA):** This means that your code produced an incorrect output for at least one of the test cases.
- **Runtime Error (RE):** A runtime error in a program is an error that occurs while the program is running after being successfully compiled. Possible reasons are division by 0 or **segmentation fault**. Often debugging helps in this case. Runtime errors are commonly called referred to as "bugs" which you may have heard.
- **Presentation Error (PE):** This is given when the output format does not match the expected format, even though the actual content might be correct.
- **Partial Accepted (PA):** Some online judges provide this verdict, which means your code passed some but not all of the test cases.
- **Judgement Failed (JF):** This can be a generic error message indicating a problem with the judging system itself.

- **Output Limit Exceeded (OLE):** On platforms like AtCoder, when you submit a solution to a programming problem on AtCoder or similar online judges, your code is tested against a set of test cases. If your code produces more output than the expected output or exceeds the output limit specified in the problem statement, you will receive an "OLE" verdict. Possible reasons include printing too much output, buffer overflow, infinite loops etc.
- **Stack Overflow:** Occurs when a computer program tries to use more memory space in the call stack than has been allocated to that stack. During recursion you may face this if recursion call limit is exceeded.
- **Idleness Limit Exceeded:** It means that your program took too long to execute. Most competitive programming platforms have a time limit within which your code must produce the correct output. If your code exceeds this time limit, it will result in an error.

General Advice

- We hope that you all are following this course so far and also giving contests regularly. Problems of **Week 1** contest were all doable. Do give it a second try if you were not able to do it in the contest.
- Don't worry about about the **massive information** revealed to you this week in **STL**. Use **Google** whenever required. You will eventually learn it by practice like Rotational Dynamics during JEE :)
- Always try to solve that one problem you could not do in contest, after the contest ends. **Consistent up-solving** contributes significantly in improving CP skills.
- If you turn to reading editorial, read hints/partial solution and give it a try again. **Implementing solutions on your own** guarantees that you properly understood the problem and the solution.
- Maintain **healthy competition**, Discuss your solutions with your friends after contest ends. Doing CP with friends not only makes it more exciting and fun but also helps in discovering different ways to approach a problem.
- You will definitely have contests where you under-perform and your rating drops. Brush that negative feeling off as quick as possible. **You always have the next contest waiting for you :)**

Sneak Peek

Excited for the next week? So are we! Expect a lot of new thrilling topics and handful of challenges waiting to be conquered.
You will be introduced to Number Theory, Bit Manipulation and many more topics which are present in one form or other in almost all questions.
Solve plenty of problems in the mean time and cheers for coming this far!
See you next week!