**Step 3: Get Familiar with Data Structures**

Again, please keep in mind our motive is not to make you memorize these Data Structures or Algorithms in the next step but to show you how can you implement these in real life problems.

We have also added some questions along with each topic so that you can get hands-on experience as to how to apply which data structure in which problem.

**Arrays and Vector**: A collection of similar data types is called an Array. Vectors are also like arrays but when combined with STL functions they prove to be far more useful than an array in Competitive Programming. Here are some great resources to understand the basics of Arrays and Vectors in C++. If you are going ahead with Java as the programming language, you can do a quick Google Search to find the equivalent Java resources as well.

Arrays and Vector Tutorials:

1. [GeeksforGeeks Array Data Structure](https://www.geeksforgeeks.org/array-data-structure/)
2. [GeeksforGeeks Vector in CPP](https://www.geeksforgeeks.org/vector-in-cpp-stl)

Problems on Arrays and Vector:

1. [Hackerrank - simple array sum problem](https://www.hackerrank.com/challenges/simple-array-sum/problem)
2. [Hackerrank - Circular array rotation](https://www.hackerrank.com/challenges/circular-array-rotation/problem)
3. <https://www.hackerrank.com/challenges/equality-in-a-array/problem>
4. <https://www.interviewbit.com/problems/wave-array/>
5. <https://www.interviewbit.com/problems/max-sum-contiguous-subarray/>

**Basic Maths**: Problems from basic mathematics and implementation are fairly common in contests as well as in interviews. Therefore, it is recommended that you should have an idea of the fundamental mathematics concepts.

Mathematical ProgrammingTutorials:

1. <https://www.geeksforgeeks.org/c-program-find-gcd-hcf-two-numbers/>
2. <https://www.geeksforgeeks.org/sieve-of-eratosthenes/>
3. <https://www.geeksforgeeks.org/modular-exponentiation-power-in-modular-arithmetic/>

 Questions on Mathematical Programming:

1. <https://www.spoj.com/problems/ADDREV/>
2. <https://www.spoj.com/problems/FCTRL2/>
3. <https://www.spoj.com/problems/FENCE1/>
4. <https://www.spoj.com/problems/PALIN/>
5. <https://www.spoj.com/problems/DIVFACT/>
6. <https://www.spoj.com/problems/AMR11E/>
7. <https://www.spoj.com/problems/TDPRIMES/>
8. <https://www.spoj.com/problems/TDKPRIME/>

**Strings**: They are collections of multiple characters and can be referred to as an array of characters. String problems are quite common in various programming contests and in fact string problems are among the favorite problems for tech interviewers.

Strings Tutorials:

1. <https://www.tutorialspoint.com/cplusplus/cpp_strings.htm>
2. <https://www.hackerearth.com/practice/algorithms/string-algorithm/basics-of-string-manipulation/tutorial/>

Problems on Strings:

1. <https://www.interviewbit.com/problems/palindrome-string/>
2. <https://www.hackerrank.com/challenges/two-characters/problem>
3. <https://www.hackerrank.com/challenges/gem-stones/problem>
4. <https://www.hackerrank.com/challenges/the-love-letter-mystery/problem>
5. <https://www.hackerrank.com/challenges/anagram/problem>
6. <https://www.interviewbit.com/problems/minimum-characters-required-to-make-a-string-palindromic/>

**Stack**: Stack is a linear data structure which follows a particular order in which the operations are performed. The order may be LIFO (Last In First Out) or FILO (First In Last Out). Stack follows LIFO.

Tutorials:

1. <https://www.geeksforgeeks.org/stack-in-cpp-stl/>
2. <https://www.hackerearth.com/practice/data-structures/stacks/basics-of-stacks/tutorial/>
3. <https://www.tutorialspoint.com/data_structures_algorithms/stack_algorithm.htm>

Problems:

1. <https://www.hackerearth.com/practice/data-structures/stacks/basics-of-stacks/practice-problems/algorithm/sniper-shooting/>
2. <https://www.interviewbit.com/problems/evaluate-expression/>
3. <https://www.hackerearth.com/practice/data-structures/stacks/basics-of-stacks/practice-problems/algorithm/monk-and-prisoner-of-azkaban/>
4. <https://www.hackerearth.com/practice/data-structures/stacks/basics-of-stacks/practice-problems/algorithm/a-game-of-numbers-1-5d3a8cb3/>
5. <https://www.interviewbit.com/problems/rain-water-trapped/>
6. <https://www.spoj.com/problems/STPAR/>
7. <https://www.spoj.com/problems/HISTOGRA/>
8. <https://www.spoj.com/problems/MMASS/>

**Queue**: A Queue is a linear structure which follows a particular order in which the operations are performed. The order is First In First Out (FIFO).

Tutorials:

1. <https://www.geeksforgeeks.org/queue-cpp-stl/>
2. <https://www.hackerearth.com/practice/data-structures/queues/basics-of-queues/tutorial/>
3. <https://www.w3schools.in/data-structures-tutorial/queue/>

Problems:

1. <https://www.hackerearth.com/practice/data-structures/queues/basics-of-queues/practice-problems/algorithm/monk-and-chamber-of-secrets/>
2. <https://www.hackerearth.com/practice/data-structures/queues/basics-of-queues/practice-problems/algorithm/chocolate-distribution-3-f9297a6e/>
3. <https://www.hackerearth.com/practice/data-structures/queues/basics-of-queues/practice-problems/algorithm/little-monk-and-goblet-of-fire/>
4. <https://www.spoj.com/problems/ADAQUEUE/>
5. <https://www.spoj.com/problems/LAZYPROG/>

**Map**: Map is by far one of the most useful Data Structures. It can be used to find, iterate, add, delete numbers, and is also one of the most widely used Data Structures.

Tutorials:

1. <https://www.geeksforgeeks.org/map-associative-containers-the-c-standard-template-library-stl/>
2. <https://www.tutorialspoint.com/cpp_standard_library/map.htm>
3. <https://www.studytonight.com/cpp/stl/stl-container-map>

Problems:

1. <https://www.hackerrank.com/challenges/cpp-maps/problem>
2. <https://www.spoj.com/problems/ADACLEAN/>
3. <https://www.interviewbit.com/problems/colorful-number/>
4. <https://www.interviewbit.com/problems/anagrams/>
5. <https://www.interviewbit.com/problems/substring-concatenation/>

**Segment Tree**:

1. <https://www.codechef.com/tags/problems/segment-tree>
2. <https://codeforces.com/blog/entry/22616>
3. Problems on (segment trees, range queries, interval trees, k-d trees, Binary index trees): <https://discuss.codechef.com/t/problems-on-segment-trees-range-queries-interval-trees-k-d-trees-binary-index-trees/3951>
   1. <http://www.spoj.com/problems/DQUERY/>
   2. [http://www.spoj.com/problems/KQUERY/](http://www.spoj/problems/KQUERY/)
   3. [http://www.spoj.com/problems/FREQUENT/](http://www.spoj/problems/FREQUENT/ )
   4. [http://www.spoj.com/problems/GSS1/](http://www.spoj/problems/GSS1/ )
   5. [http://www.spoj.com/problems/GSS2/](http://www.spoj/problems/GSS2/ )
   6. [http://www.spoj.com/problems/GSS3/](http://www.spoj/problems/GSS3/ 8)
   7. [http://www.spoj.com/problems/GSS5/](http://www.spoj/problems/GSS5/ 4)
   8. [https://www.spoj.com/problems/KGSS/](https://www.spoj.com/problems/KGSS/ )
   9. [http://www.spoj.com/problems/HELPR2D2/](http://www.spoj/problems/HELPR2D2/)
   10. <https://www.spoj.com/problems/INCSEQ/>
   11. <https://www.spoj.com/problems/INCDSEQ/>
   12. <https://www.spoj.com/problems/QTREE/>
   13. [http://www.spoj.com/problems/QTREE2/](http://www.spoj/problems/QTREE2/ 1)
   14. [http://www.spoj.com/problems/QTREE3/](http://www.spoj/problems/QTREE3/)
   15. [http://www.spoj.com/problems/BRCKTS/](http://www.spoj/problems/BRCKTS/)
   16. [http://www.spoj.com/problems/CTRICK/](http://www.spoj.pl/problems/CTRICK/)
   17. [http://www.spoj.com/problems/MATSUM/](http://www.spoj/problems/MATSUM/)
   18. [http://www.spoj.com/problems/RATING/](http://www.spoj/problems/RATING/)
   19. [http://www.spoj.com/problems/RRSCHED/](http://www.spoj/problems/RRSCHED/)
   20. [http://www.spoj.com/problems/SUPPER/](http://www.spoj/problems/SUPPER/)
   21. [http://www.spoj.com/problems/ORDERS/](http://www.spoj/problems/ORDERS/)
4. Notes: <https://codeforces.com/blog/entry/15890>

**Step 4: Get Familiar with Algorithms**

Algorithms are logics that are implemented on various Data Structures to achieve the desired output.

Time/Space Complexity: Every Algorithm has a Time and Space complexity which refers to the maximum amount of time an Algorithm will take and the maximum amount of memory an algorithm will require. While doing Competitive Programming these two will play a key role in determining the verdict of your solution.

Always try to think of the most optimal solution, that is, one which runs with least time complexity and occupies minimum space.

Sorting: You must have heard of a number of sorting techniques to sort but while doing Competitive Programming most of those techniques prove to be time-consuming hence the STL library comes to rescue, it offers a function sort() which sorts the array in the most optimal way.

Types of Algorithms:

* **Greedy:** A solution in which we move step by step towards our final goal if referred to as greedy algorithm.

Tutorials:

1. <https://www.hackerearth.com/practice/algorithms/greedy/basics-of-greedy-algorithms/tutorial/>
2. <https://www.topcoder.com/community/competitive-programming/tutorials/greedy-is-good/>

Problems:

1. <https://www.hackerrank.com/challenges/minimum-absolute-difference-in-an-array/problem>
2. <https://www.hackerrank.com/challenges/marcs-cakewalk/problem>
3. <https://www.spoj.com/problems/BUSYMAN/>
4. <https://www.spoj.com/problems/GERGOVIA/>
5. <https://www.spoj.com/problems/BAISED/>
6. <https://www.spoj.com/problems/BALIFE/>
7. <https://www.hackerrank.com/challenges/grid-challenge/problem>
8. <https://www.hackerrank.com/challenges/maximum-perimeter-triangle/problem>
9. <https://www.hackerrank.com/challenges/sherlock-and-the-beast/problem>

* **Divide and Conquer:** As the name suggests, in this we try to make the problem easier by dividing it into a number of subproblems and then solving them one at a time and then combining them all together in the end to give a final answer.

Tutorials:

1. [https://www.hackerearth.com/practice/algorithms/searching/binary-search/tutorial](https://www.hackerearth.com/practice/algorithms/searching/binary-search/tutorial/)
2. <https://www.hackerearth.com/practice/notes/power-of-binary-search/>
3. <https://www.topcoder.com/community/competitive-programming/tutorials/binary-search/>

Problems:

1. <https://www.spoj.com/problems/INVCNT/>
2. <https://www.codechef.com/problems/MRGSRT>
3. <https://www.spoj.com/problems/ABCDEF/>
4. <https://www.spoj.com/problems/AGGRCOW/>
5. Not done: <https://www.spoj.com/problems/PIE/>
6. Not done: <https://www.codechef.com/problems/STRSUB>
7. Not done: <https://www.spoj.com/problems/MKUHAR/>
8. <https://www.spoj.com/problems/SUBS/>

* **Recursion and backtracking:** Recursion in the type of algorithm in which a function calls itself again and again to achieve the final output. It makes use of stack data structure. Backtracking is an algorithmic paradigm that tries different solutions until finds a solution that “works”. It makes use of recursion.

Tutorials:

1. <https://www.geeksforgeeks.org/recursion/>
2. <https://www.topcoder.com/community/competitive-programming/tutorials/an-introduction-to-recursion-part-1/>
3. <https://www.geeksforgeeks.org/backtracking-algorithms/>

Problems:

1. <https://www.hackerearth.com/practice/basic-programming/recursion/recursion-and-backtracking/practice-problems/algorithm/simran-and-stairs/>
2. <https://www.interviewbit.com/problems/subset/>
3. <https://www.interviewbit.com/problems/kth-permutation-sequence/>
4. <https://www.hackerearth.com/practice/basic-programming/recursion/recursion-and-backtracking/practice-problems/algorithm/n-queensrecursion-tutorial/>
5. <https://www.hackerearth.com/practice/basic-programming/recursion/recursion-and-backtracking/practice-problems/algorithm/a-tryst-with-chess/>
6. <https://www.interviewbit.com/problems/sudoku/>

* **Dynamic Programming:** In DP we break a problem into a number of problems and then conquer them one by one and store the outcome of the previous subproblem to compute the output of the next one.

Tutorials:

1. <https://qr.ae/TUnXQo>
2. <https://nitkcccc.wordpress.com/2014/02/27/dynamic-programming-tutorial-level-1-easy-medium-part-12/>
3. <https://www.topcoder.com/community/competitive-programming/tutorials/dynamic-programming-from-novice-to-advanced/>

Problems:

1. <https://www.spoj.com/problems/COINS/>
2. <https://www.spoj.com/problems/EDIST/>
3. <https://www.codechef.com/problems/DELISH>
4. <https://www.codechef.com/problems/GRID>
5. <https://www.codechef.com/problems/MATRIX2>
6. <https://www.codechef.com/problems/DBOY/>
7. <https://www.codechef.com/JUNE13/problems/LEMOUSE>
8. <https://www.codechef.com/problems/STRSUB(also> in div and conquer)

* **Graph Theory:**

Tutorial:

1. <https://www.hackerearth.com/practice/notes/graph-theory-part-i/>
2. <https://www.geeksforgeeks.org/fundamentals-of-algorithms/#GraphAlgorithms>
3. <https://www.topcoder.com/community/competitive-programming/tutorials/introduction-to-graphs-and-their-data-structures-section-2/>

Problems:

1. <https://www.spoj.com/problems/PPATH/> (Breadth-first Search)
2. <https://www.spoj.com/problems/ONEZERO/> (Breadth-first Search)
3. <https://www.spoj.com/problems/PT07Z/> (Depth-first Search)
4. <https://www.spoj.com/problems/BUGLIFE/> (Depth-first Search)
5. <https://www.spoj.com/problems/SHPATH/> (Shortest path)
6. <https://www.spoj.com/problems/TRAFFICN/> (Shortest path)
7. <https://www.spoj.com/problems/SAMER08A/> (Shortest path)
8. <https://www.codechef.com/problems/DIGJUMP> (Shortest path)
9. <https://www.codechef.com/AMR14ROS/problems/AMR14B> (Shortest path)
10. <https://www.codechef.com/problems/SPSHORT> (Shortest path)

 For more information on Algorithms refer to this link: <https://www.geeksforgeeks.org/lmns-algorithms-gq/>

<https://careerhigh.in/Programming/Competitive%20Programming%20for%20Beginners%20Roadmap>