Competitive Programming 1 (0 → 1200)

Low Jia Jun

May 2024 - June 2024

1 Introduction

This document documents my journey to becoming a competitive programmer from scratch. Prior to this journey, much of my programming experience was done without tight time constraints. The choice of programming language used for this journey is Java.

2 Getting Started [500-999]

It took me a while (the first 11 days) to be used to the Java syntax.

2.1 Error Handling

It was common for me to run into errors.

```
// Problem: FSQRT
    int output = Math.sqrt(n); // Narrow casting is done manually
    int output = (int) Math.sqrt(n);
```

Simple mathematical equations such as $s = \frac{d}{t}$ require working with doubles, which may not be safe if we are working with extremely small numbers.

```
// Problem: SPEEDTEST
    double epsilon = 1e-9; // Tolerance for zero
    if (Math.abs(as-bs) < epsilon) {
        System.out.println("EQUAL");
    }
```

Working with extremely big numbers sometimes require the use of BigInteger class in Java.

```
// Problem: FCTRL2
import java.math.BigInteger;
    public static BigInteger factorial(int n) {
        BigInteger res = BigInteger.ONE;
        for (int i = 0; i <= n; i++) {
            res = res.multiply(BigInteger.valueOf(i));
        }
    }

Time Complexity: O(n)

// Problem: FLOW018
// The constraints here is 0 <= N <= 20 hence using long will AC
// It is noteworthy that such an approach will fail when N>49.
```

In Python, programmers use "==", "!=" operators to compare strings. However, when we are comparing strings in Java, we must use the .equals() method.

```
// Problem: RECENTCONT
    for (int i = 0; i < n; i++) {
        String s = sc.next();
        if (s.equals("START38")) {
            st++;
        }
    }
```

2.2 Data Structures Recap

Input/output parsing was done with Java Scanner class for ease of readability.

```
// Problem: DPOLY (Array)
while (t-- > 0) {
    int n = sc.nextInt();
    int[] arr = new int[n]; // new array of integers of size n
    for (int i = 0; i < n; i++) {
        arr[i] = sc.nextInt();
    }
}
// It is also possible to use ArrayList<Integer> arr = new ArrayList<>();
```

```java
// Problem: PALL01 (Stack, Legacy Class)
while (t-- > 0) {
    String s = sc.next();
    Stack<Character> stack1 = new Stack<>(); // new stack1
    Stack<Character> stack2 = new Stack<>(); // new stack2

    for (char c : s.toCharArray()) {
        stack1.push();
    }
    for (int i = s.length() - 1; i > 0; i--) {
        stack2.push(s.charAt(i));
    }

    boolean isPalindrome = true;
    while (!stack1.isEmpty()) {
        if (stack1.pop() != stack2.pop()) {
            isPalindrome = false;
            break;
        }
    }
}

// Problem: WORDLE (StringBuilder)
while (t-- > 0) {
    String s = sc.next(); String t = sc.next();
    StringBuilder str = new StringBuilder(); // mutable string
    for (int i = 0; i < 5; i++) {
        if (s.charAt(i) == t.charAt(i)) {
            str.append('G');
        } else {
            str.append('B');
        }
    }
}

// Problem: TWORANGES (HashSet)
while (t-- > 0) {
    int a = sc.nextInt(); int b = sc.nextInt();
    int c = sc.nextInt(); int d = sc.nextInt();
    HashSet<Integer> set = new HashSet<>();
    for (int i = a; i <= b; i++) {
        set.add(i);
    }
    for (int i = c; i <= d; i++) {
        set.add(i);
    }
    System.out.println(set.size());

}
```

```java
// Problem: COOK82A (HashMap)
while (t-- > 0) {
    HashMap<String, Integer> hm = new HashMap<>();

    for (int i = 0; i < 4; i++) {
        hm.put(sc.next(), sc.nextInt());
    }

    boolean b = false;

    if (hm.get("RealMadrid") < hm.get("Malaga") && hm.get("Barcelona") >
         hm.get("Eibar")) {
        b = true;
    }

    System.out.println(b ? "Barcelona" : "RealMadrid");

}
```

I have learnt a great deal of data structures and its implementations, more specifically, methods and syntax. In Cook82A (HashMap), I have made learnt the use of ternary operator in Java which is useful for shortening my solution.

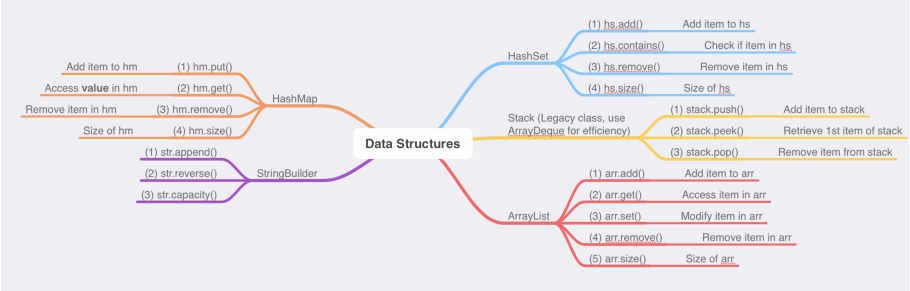

Figure 1: Summary of Data Structures

3 Practice [1000-1149]

3.1 HashMap

HashMap is a popular data structure due to its ability to store key-value pairs. $put(), get()$ methods were introduced in COOK82A but I will formally introduce more methods in this subsection.

```
// Problem: REMOVECARDS (HashMap)
while (t-- > 0) {
    int n = sc.nextInt();
    int[] cards = new int[n];
    HashMap<Integer, Integer> hm = new HashMap<>();

    for (int i = 0; i < n; i++) {
        cards[i] = sc.nextInt();
        hm.put(cards[i], hm.getOrDefault(cards[i], 0) + 1);
    }

    int max = Collections.max(hm.values());

    System.out.println(n - max);
}
```

Method Name	**Documentation**
put(K key, V value)	Adds a key-value pair to HashMap
get(Object key)	Returns the value to which specified key is mapped or 'null' if the map contains no mapping for the key
getOrDefault(Object key, V defaultValue)	Returns the value to which specified key is mapped or 'defaultValue'; if the map contains no mapping for the key
values()	Returns a 'Collection' view of the values contained in this map
Collections.max(hm.values())	Returns the maximum element of the HashMap
keySet()	Return a set of all keys of HashMap

```java
// Problem: EVENTUAL (HashMap)
while (t-- > 0) {
    int n = sc.nextInt();
    String s = sc.next();
    HashMap<Character, Integer> hm = new HashMap<>();
    for (char c : s.toCharArray()) {
        hm.put(c, hm.getOrDefault(c,0) + 1); // Important Step
    }
    boolean flag = true;
    for (int c : hm.values()) { // for-each loop
        if (c % 2 == 1) {
            flag = false;
            break;
        }
    }

    System.out.println(flag ? "YES" : "NO");

}

// Problem: GRPASSN (HashMap)
while (t-- > 0) {
    int n = sc.nextInt();
    int[] arr = new int[n];
    HashMap<Integer, Integer> hm = new HashMap<>();

    for (int i = 0; i < n; i++) {
        arr[i] = sc.nextInt();
        hm.put(arr[i], hm.getOrDefault(arr[i],0) + 1);
    }

    boolean flag = false;

    for (int key : hm.keySet()) {
    // hm.keySet() iterate through k and access v directly
        if (hm.get(key) % key != 0) {
            flag = true;
            break;
        }
    }

    System.out.println(flag ? "NO" : "YES");
}
```

3.2 TreeSet/HashSet

HashSet is much faster than TreeSet (constant-time versus log-time for most operations like add, remove and contains) but offers no ordering guarantees like TreeSet.

```
// Problem: VOTERS (Use of TreeSet to preserve the order of voters since
    HashSet does not guarantee ordering)

TreeSet<Integer> ans = new TreeSet<>();

for (int i = 0; i < (n1+n2+n3) - 1; i++) {
    if (arr[i] == arr[i+1]) {
        ans.add(arr[i]);
    }
}

System.out.println(ans.size());
for (int v : ans) {
    System.out.println(v);
}
```

We present a question where using a HashSet is appropriate in summary.

3.3 Strings

```
// Problem: ZEROSTRING (String Manipulation)
while (t-- > 0) {
    int n = sc.nextInt();
    String s = sc.next();

    int c0 = 0; // Count 0s
    int c1 = 0; // Count 1s

    for (int i = 0; i < n; i++) {
        c0 += (s.charAt(i) == '0') ? 1 : 0;
        c1 += (s.charAt(i) == '1') ? 1 : 0;
    }

    System.out.println(Math.min(c1,c0+1));
}

// Problem: PRIMEREVERSE (String Manipulation)
while (t-- > 0) {
    int n = sc.nextInt();
    String a = sc.next();
    String b = sc.next();

    char[] arr_a = a.toCharArray();
    char[] arr_b = b.toCharArray();

    Arrays.sort(arr_a); // O(n log n)
    Arrays.sort(arr_b);

    String s_a = new String(arr_a);
    String s_b = new String(arr_b);

    if (s_a.equals(s_b)) {
        ...
    }
} // Alternative is to count number of 1s and 0s of a and b respectively.

// Problem: SRTARR (String Manipulation)
//  Example: 01001010 -> 00010101 -> 00001011 -> -> 00000111 (sorted).
//      Intuition: Simply count the number of "10"s.

// Problem: ALTSTR (String Manipulation)
//  Simply count the number of 1s and 0s.
//      If n0 = n1, then we can have a perfect alternating sequence.
//      If n0 > n1, then start from 0101...000 (store extra 0s at end)
//      If n0 < n1, then start from 1010...111 (store extra 1s at end)
```

The table contains some common string methods used in this problem set, coupled with StringBuilder and ArrayList which are representations of mutable Strings and dynamic Arrays in Java.

String Methods	**Description**
s.charAt(i)	Returns the character at specified index
s.toCharArray()	Converts given s to a sequence of char
new String(...)	Converts sequence of char back to string
s.length()	Return length of s
Character.isDigit(c)	Determine whether a specified c is a digit
Character. getNumericValue(c)	Return numeric value of c
StringBuilder	**Description**
str.append()	append
ArrayList	**Description**
arr.add(s.charAt(i))	Add element to list
arr.set(idx, element)	Replace element at specified idx

3.4 Mathematics

1. In COUNTP, it can be proven that $sum(S_1)$ and $sum(S_2)$ must both be odd for their products to be odd and the sum $sum(S_1) + sum(S_2)$ must be even. Hence, a shortcut to solve this problem is to simply find an odd integer in A and check if the sum of A is even for a valid subsequence S_1 and S_2 to exist.

2. In ODDPAIRS, we make use of the fact that the sum of an even and odd integer is odd. If n is even, there are $\frac{n}{2}$ even integers and $\frac{n}{2}$ odd integers. If n is odd, there are $\frac{n+1}{2}$ even integers and $\frac{n-1}{2}$ odd integers. We use the basic principle of counting to determine that there are $2 * \frac{n}{2} * \frac{n}{2}$ or $2 * \frac{n+1}{2} * \frac{n-1}{2}$ pairs that satisfy the problem statement.

3. ```java
   // Problem: PRIME1
   public static boolean isPrime(int prime) {
       if (prime <= 1) {
           return false;
       }
       for (int i = 2; i <= Math.sqrt(prime); i++) {
           if (prime % i == 0) {
               return false;
           }
       }
       return true;
   }
   ```

4. In EQUALIZEAB, it can be proven that

   $$|b - a| \% (2 * x) = 0$$

   to make $A = B$, if $A \neq B$ initially.

5. ```java
   // Problem: RPD
   public static long digitSum(long num) {
       long sum = 0;
       while (num > 0) {
           sum += num % 10;
           num /= 10;
       }
       return sum;
   }

   // For the sake of simplicity, we make all methods public.
   ```

6. In MINMXOR, bitwise XOR (logical exclusive OR operation) is introduced. This subset of mathematics is known as boolean algebra.

3.5 Chess

In this subsection, we will cover ideas presented in POOK and KNIGHT2.

Statement 1 *The N Queen is the problem of placing N chess queens on an N×N chessboard so that no two queens attack each other.*

We make use of **Statement 1** to figure out that for $N \geq 4$, the solution is simply N pooks. It is not possible to place $\geq N + 1$ chess queens on an NxN chessboard (proof by contradiction).

Statement 2 *Given a* 8×8 *chessboard, in exactly* 100 *moves,* $(X_1, Y_1) \rightarrow (X_2, Y_2)$ *is possible if and only if* (X_1, Y_1) *and* (X_2, Y_2) *are of the same colour.*

Assume $(1, 1)$ is a black square, then we observe that $(1, 2)$ is a white square, and so on... We observe that $(X_i + Y_i)$ is either an odd or even integer, which is useful to denote whether (X_i, Y_i) is a black or white square for $i = 1, 2$.

3.6 Summary

We will summarise strings using CIELAB.

```
// Problem: CIELAB

// Approach: We wish to use convert the correct answer of A-B to a
    character array, and iterate from the last element of the array,
    changing the first nonzero element to zero. After, we will convert
    the character array back to a string and output the string.

   int a = sc.nextInt();
   int b = sc.nextInt();
   int ans = b - a; // ans > 0 is guaranteed

   String s = String.valueOf(ans); // Convert int -> str
   char[] arr = s.toCharArray(); // Convert str -> arr

   for (int i = s.length() - 1; i >= 0; i--) {
       if (arr[i] != '0') {
           arr[i] = (arr[i] == '1') ? '2' : '1';
           break;
       }
   }

   System.out.println(new String(arr)); // Convert arr -> str
```

We will summarise basic programming skills without data structures using DWNLD.

```
// Problem: DWNLD

while (t-- > 0) {
    int n = sc.nextInt(); // (t1,d1) , ... , (tn,dn)
    int k = sc.nextInt(); // first k minutes is free

    // next n lines contains (t1,d1) , ... , (tn,dn)

    int sol = 0;

    for (int i = 0; i < n; i++) {
        int t = sc.nextInt();
        int d = sc.nextInt();

        if (t < k) {
            k -= t; // keep updating k -> 0
        } else {
            sol += (t-k) * d; // keep updating sol
            k = 0; // k converges to 0 here
        }
    }

    System.out.println(sol);

}
```

We will summarise HashSet/HashMap using MATPAN.

Statement 3 *A pangram is a sentence containing every letter of the alphabet.*

```
// Problem: MATPAN

// Approach: We can use a HashSet to store alphabet-price pair, after we
    wish to iterate through the string, for each character, we wish to
    set the price of that alphabet in the HashSet to zero. After
    iterating through the string, we return the sum of all values in
    the HashSet.
```

```
// Problem: MATPAN

while (t-- > 0) {

    HashSet<Character, Integer> hs = new HashSet<>()

    for (char c = 'a'; c <= 'z'; c++) {
        hs.put(c, sc.nextInt());
    }

    String s = sc.next();

    for (char c : s.toCharArray) {
        if (hs.contains(c)) {
            hs.put(c,0);
        }
    }

    int ans = 0;

    for (int v : hs.values()) {
        ans += v;
    }

    System.out.println(ans);
}
```

4 Conclusion [1149-1200]

Date	Contest	Solves	Rating	Percentile
20 May 2024	Codeforces Round 946	3	784	84 (All Divs)
22 May 2024	CodeChef Starters 135	3	1098	67 (Div 4)
26 May 2024	Codeforces Round 948	2	1088	84 (All Divs)
30 May 2024	Codeforces Round 952	3	1188	61 (All Divs)
12 June 2024	CodeChef Starters 138	4	1359	96 (Div 4)

A competitive programmer once told me that the most challenging part of competitive programming is implementing the solution during the contest itself and it is OK as implementing the solution comes from experience. I have enjoyed myself throughout this month's adventure as I look forward to improving in the upcoming months.