# November JDCC Solutions + Comments

## **Problem A** (Euler):

Solution: We iterate over the list of people and keep track of the one with the earliest discovery date. Note that we do not need to store the whole list of names.

Time Complexity: O(N)

Space Complexity: O(1)

## **Problem B** (Reddit At Work):

Solution: The first challenge in this problem was to parse the input. It is important to know how to parse integers out of strings. In Java, we could use s.split(“:”) to create an array which uses “:” as a delimiter to split s into tokens.

After we parse the input, we iterate over the input, keeping track of the start and end of the current block of meetings. If we run into a meeting that starts before our end time, we update our end time with the max of the two meetings. Otherwise, we add end – start to our total meeting time and update the start/end time. In this way, we correctly process meeting overlap, acheiving the correct answer.

Time Complexity: O(N)

Space Complexity: O(1)

## **Problem C** (Painting Cost):

Solution: We read in the input into a 2D array, then we iterate over it and check the number of unique characters in each 2x2 block. There are many ways of doing this, of which we will present two:

1) Place the 4 colours in an array and sort it. Then iterate over an array and count the number of times a[i] != a[i+1].

2) Place the 4 colours into a set data structure, which automatically removes duplicates.

Time Complexity: $O(RC)$

Space Complexity: $O(RC)$

## **Problem D** (Guess My Age):

Solution: This is a fairly simple dynamic programming problem, however it is tricky to see the underlying reccurence that gives us a good run time.

The key insight here is to notice that whether you know the person's age is between 1 and M or between N and N+M-1, the remaining cost that you will incur is the same, as the size of both ranges is the same. The actual indices at which the range lies doesn't matter, only the size.

Hence, we can make our dp state to be the size of a range. The transition then becomes clear: iterate over all elements in the range and try asking the "Are you over X years old?" question there. Hence, we get a one-dimensional DP with a O(N) transition time, giving us the $O(N^2)$ run time needed to solve the problem.

Some further analysis shows that this problem could likely be solved with a ternary search, which would reduce the runtime to O(N log N).

Time Complexity: $O(N^2)$

Space Complexity: $O(N)$

## **Problem E** (Catchy Music):

Solution: This problem falls in the category of problems that ask you to find properties of substrings in O(N) time. The most famous example of such a problem is the “find if S is a substring of T” problem. These problems are notoriously difficult to implement and are best learned through practice.

One way to solve this problem would be to find the longest prefix of S present in S, which would give you a period of the melody. An algorithm such as KMP could be used to solve this in O(N) time.

Another way to solve this problem would have been to use regular expressions. The one we used to find duplicates was:

```
"(.+?)\1+$"
```

It is definitely not necessary for you to know regexes, however if you are interested in such problems then they are an interesting field to explore.

Time Complexity: O(N)
Space Complexity: O(N)