1. For finding the shortest sequence, the algorithm is:

```
MinSeq{
        n = j - i + 1
        D[1] = 0
        If i - j button exists:
                Return 1
        For i, 0 to n:
                For j, 0 to i -1:
                        dp[i] = min(dp[i], dp[i-j] + 1)
}

Count = 0
NumSeq(minSeq, current, final, len, b){
        if(len == minSeq and current == final)
                Count++
        if(len > minSeq)
                Print "NO"
        For i, 0 to b.length:
                if(current + b[i] <= final)
                len++;
                NumSeq(minSeq, current + b[i], final, len, b)
}
```

For finding the minimum seq, the best time complexity is:

= **O(1)**

For the worst time case, complexity will be:

= **$O(n^2)$**

Count is the number of sequences present.

2. The decision tree for this dynamic problem will be based on the red and green choices.At each step we have 2 choices either to take
    a. one red and two green cookies.
    b. or two red and one green cookies.

   Now at each step we will try for both the choices and reach to the optimal result.Now. Since we are trying both the possibility and trying to find the possibility of winning for one of the players this algorithm will give the correct results.

   Algorithm:

```
        Winner(red, green, player){
                If (red == 0 or green == 0)
                        return player^1
                if(red <= 1 or green <=1)
                        return player^1
                if(dp[red][green] != -1)
```

```
            return dp[red][green]
        dp[red][green] = winner(red-2, green-1, player^1) or winner(red-1, green-2,
        player^1)
        return dp[red][green]
    }
```

3. Let the input sequences be X[0..m-1] and Y[0..n-1] of lengths m and n respectively. And let L(X[0..m-1], Y[0..n-1]) be the length of LCS of the two sequences X and Y. Following is the recursive definition of L(X[0..m-1], Y[0..n-1]).
   If last characters of both sequences match (or X[m-1] == Y[n-1]) then
   L(X[0..m-1], Y[0..n-1]) = 1 + L(X[0..m-2], Y[0..n-2])
   If last characters of both sequences do not match (or X[m-1] != Y[n-1]) then
   L(X[0..m-1], Y[0..n-1]) = MAX (L(X[0..m-2], Y[0..n-1]), L(X[0..m-1], Y[0..n-2]))

   Examples:
   a. Consider the input strings "AGGTAB" and "GXTXAYB". Last characters match for the strings. So length of LCS can be written as:
      L("AGGTAB", "GXTXAYB") = 1 + L("AGGTA", "GXTXAY")
   b. Consider the input strings "ABCDGH" and "AEDFHR. Last characters do not match for the strings. So length of LCS can be written as:
      L("ABCDGH", "AEDFHR") = MAX (L("ABCDG", "AEDFHR"), L("ABCDGH", "AEDFH"))

   So the constrained LCS problem has optimal substructure property as the main problem can be solved using solutions to subproblems.

4. Suppose that T is a minimum weight spanning tree for G(V,E) that does not contain the edge e.
   Then Consider the graph T + e.
   This graph must contain a cycle C that contains the edge e.
   Let f be an edge of C different from e, and set
   $T^* = T + e - f$.
   Then T* is also a spanning tree for G, but
   $wt\ T^* = wt(T + e - f) = wt(T) + wt(e) - wt(f) < wt(T)$,
   contrary to T being a minimum weight spanning tree.
   Hence no such tree T (i.e., without e) can exist.
   Therefore, G(V,E) will always contain the edge e with least weight.