873. Length of Longest Fibonacci Subsequence

A sequence $x_1, x_2, ..., x_n$ is *Fibonacci-like* if:

- $n \geq 3$
- $x_i + x_{i+1} = x_{i+2}, \forall\, i+2 \leq n$

Given a **strictly increasing** array `arr` of positive integers forming a sequence, return *the **length** of the longest Fibonacci-like subsequence of* `arr`. If one does not exist, return `0`.

A **subsequence** is derived from another sequence `arr` by deleting any number of elements (including none) from `arr`, without changing the order of the remaining elements. For example, `[3, 5, 8]` is a subsequence of `[3, 4, 5, 6, 7, 8]`.

Example 1:

```
Input: arr = [1,2,3,4,5,6,7,8]
Output: 5
Explanation: The longest subsequence that is fibonacci-like: [1,2,3,5,8].
```

Example 2:

```
Input: arr = [1,3,7,11,12,14,18]
Output: 3
Explanation: The longest subsequence that is fibonacci-like: [1,11,12], [3,11,14]
or [7,11,18].
```

Constraints:

- `3 <= arr.length <= 1000`
- `1 <= arr[i] < arr[i + 1] <=` 10^9

873. Length of Longest Fibonacci Subsequence

```
/*
   Brute force+Binary search
   Time complexity: O(n^2 logn)
   Space complexity:O(1)
*/
class Solution {
public:
   int lenLongestFibSubseq(std::vector<int>& arr) {
      int n=arr.size();
      if(n<3) return 0;

      int ans=0;
      // For each F_i(F_{i-1}) and F_j(F_{i-2})
      for(int i=0;i<n-2;++i){
         for(int j=i+1;j<n-1;++j){
            int len=2; // Default len

            // Determine the next Fibonacci term F_i
            int f0=arr[i];
            int f1=arr[j];
            int f=f0+f1;

            // Search it in the array
            int k=std::lower_bound(arr.begin()+2,arr.end(),f)-arr.begin();

            // While it exists
            while(k<n && arr[k]==f){
               len++; // Increment the length by 1

               // Pass to the next triplet
               f0=f1;
               f1=f;
               f=f0+f;
               k=std::lower_bound(arr.begin()+k,arr.end(),f)-arr.begin();
            }
            ans=std::max(ans,len);
         }
      }
      return ans>2?ans:0;
   }
};
```

873. Length of Longest Fibonacci Subsequence

Intuition

In a Fibonacci-like sequence, each number depends on the two numbers that came before it. This suggests that if we know the length of a Fibonacci-like sequence ending with two particular numbers, we can use that information to find longer sequences that might include these numbers. This aspect of building larger sequences from information collected from smaller ones suggests a dynamic programming approach.

873. Length of Longest Fibonacci Subsequence

```
/*
   DP+Two pointers
   Time complexity: O(n^2)
   Space complexity: O(n^2)
*/
class Solution {
public:
   int lenLongestFibSubseq(std::vector<int>& arr) {
      int n=arr.size();
      if(n<3) return 0;
      std::vector<std::vector<int>> dp(n,std::vector<int>(n,2));
      int ans=0;

      // For each sum at arr[i]
      for(int i=2;i<n;++i){
         // Use two pointers technique to find the elements arr[r]+arr[l]=arr[i]
         // in range[0,i-1]
         int l=0,r=i-1;
         while(l<r){
            int s=arr[l]+arr[r];
            if(s==arr[i]){
               // Add one the the precomputed result
               dp[i][r]=dp[r][l]+1;
               ans=std::max(ans,dp[i][r]);
               l++;
               r--;
            }
            else if(s>arr[i]) r--;
            else l++;
         }

      }
      return ans>2?ans:0;
   }
};
```