MICROSOFT PREPARATION

DAY : 13

Frog Jump :

Problem Link : https://leetcode.com/problems/frog-jump/

Test Cases Passed : 52 / 52

Time Used : 40.20 + Editorial (Read Only)

Difficulty Level : HARD

Approach Used :

- Creating a set containing all the nodes
- Creating a set of pairs according to which the positions with kth jump will failed

CanCross() :

- Traversing for all stones :
    - Inserting the stones/nodes into the main set
- Returning the dfs call to check if we can reach the stone from the 0th stone with 0th jump of k at start as dfs(0,0,stones)

DFS() :

- Check if we are already at the end stone :
    - Return true
- Check if we are failed at the particular position with k jump position :
    - Return false
- Check if the set doesn't contain the position we are at ie. no stone there :
    - Return false
- Traverse for all the possible jumps (ie. K-1,K,K+1) :
    - If jump is negative :
        - Continue as we don't want to go back in order to reach end
    - Else check if we can reach to the end using the dfs call with the element position, jump from position as dfs(position,jumpfromposition,stones) :
        - Return true if we can reach
- Insert the position and the jump in the failed set as we can not reach end in this case
- Return false // as we cannot reach end in this case

get_next_k():

- Return {k-1,k,k+1};

Solution :

```
set<pair<int, int>> failed;
   unordered_set<int> Set;
   bool canCross(vector<int>& stones) {
       for(auto x: stones)
       {
           Set.insert(x);
       }
       return dfs(0,0,stones);
   }
   bool dfs(int pos,int k,vector<int>&stones)
   {
       // we reached the last stone
       if(pos==stones.back())
       {
           return true;
       }
       // check if the particular stone failed
       if(failed.find({pos,k})!=failed.end())
       {
           return false;
       }
       // if we can see that pos is not in the unordered set
       if(Set.find(pos)==Set.end())
       {
           return false;
       }
       // checking for adjacent elements
       for(auto next_k:get_next_k(k))
       {
           if(next_k<=0)continue;
           if(dfs(pos+next_k,next_k,stones))
           {
               return true;
           }
       }
       failed.insert({pos,k});
       return false;
   }
   array<int, 3> get_next_k(int k)
   {
       return {k-1,k,k+1};
   }
```