# Experiment-3.2

## Student Name: Nabha Varshney UID: 20BCS4995

**Branch: CSE Section/Group: 20BCS-DM-704 (A)**

## Semester: 6th Date of Performance:03rdMay 2023 Subject Name: Competitive Coding II Subject Code: 20CSP- 351

**Aim** – To demonstrate the concept of backtracking

## Objective-

* The objective is to build problem solving capability and to learn the basic concepts of data structures.
* The implementation of binary watch using backtracking.
* The implementation of word ladder II.

## Binary Watch

<https://leetcode.com/problems/binary-watch/>

**Code –**

class Solution

{

public:

    vector<string> readBinaryWatch(int num)

    {

        vector<string> result;

        for(int hours = 0; hours < 12; hours++)

        {

            for(int minute = 0; minute < 60; minute++)

            {

                string temp = "";

                if(\_\_builtin\_popcount(hours) + \_\_builtin\_popcount(minute) == num )

                {

                    temp += to\_string(hours) + ":";

                    if(minute < 10)

                    {

                        temp += "0";

                    }

                    temp += to\_string(minute);

                    result.push\_back(temp);

                }

            }

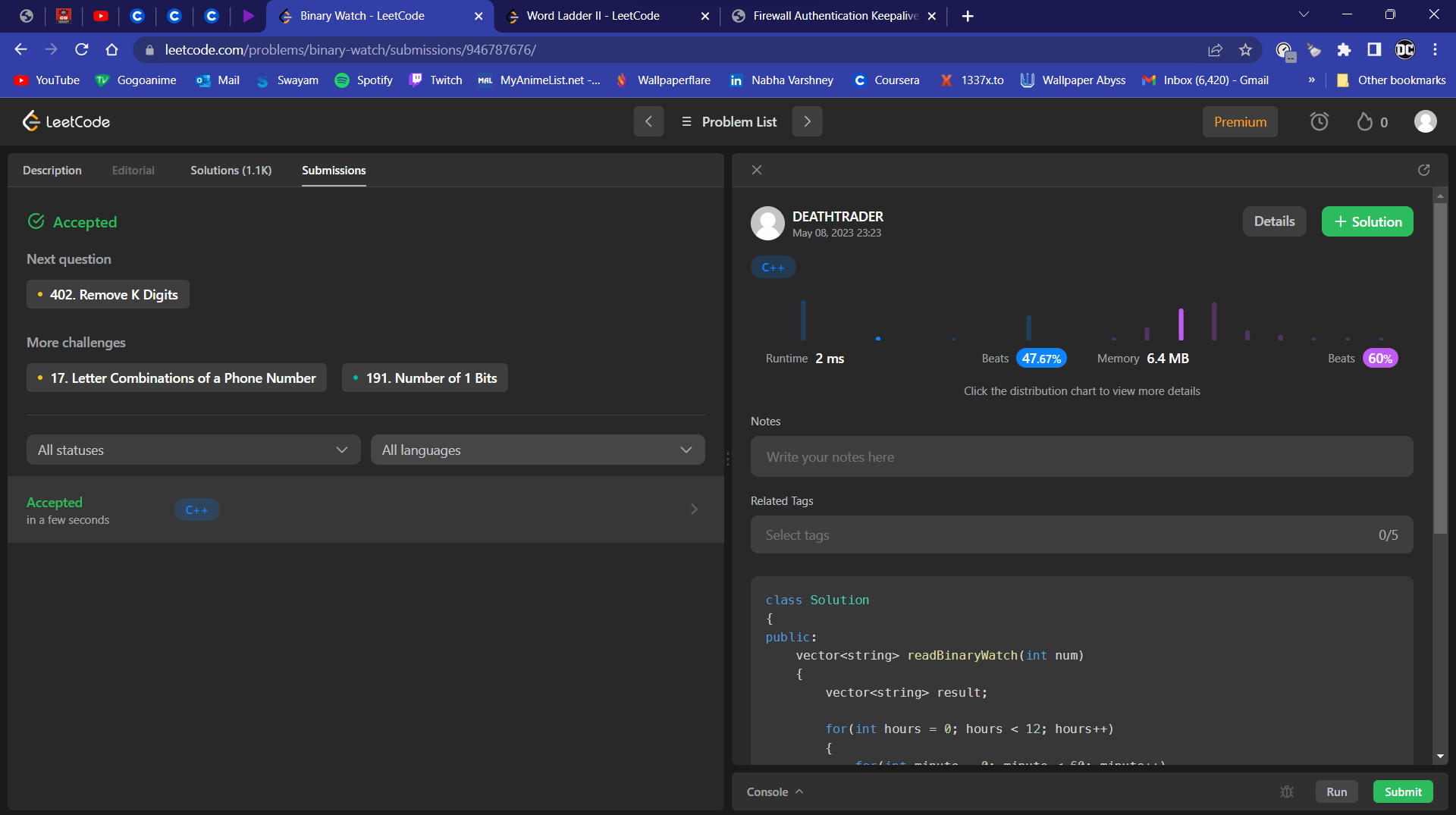
        }

        return result;

    }

};

## Output -

****

1. **Word Ladder II**

<https://leetcode.com/problems/word-ladder-ii/>

**Code -**

class Solution

{

public:

    vector<vector<string>> res;

    vector<string> te;

    unordered\_map<string, int> mp;

    string b;

    void dfs(string s)  // Step 2

    {

        te.push\_back(s);

        if (s == b)

        {

            vector<string> x = te;

            reverse(x.begin(), x.end());

            res.push\_back(x);

            te.pop\_back();

            return;

        }

        int cur = mp[s];

        for (int i = 0; i < s.size(); i++)

        {

            char c = s[i];

            for (char cc = 'a'; cc <= 'z'; cc++)

            {

                s[i] = cc;

                if (mp.count(s) && mp[s] == cur - 1)

                    dfs(s);

            }

            s[i] = c;

        }

        te.pop\_back();

        return;

    }

    vector<vector<string>> findLadders(string beginWord, string endWord, vector<string> &wordList)

    {

        unordered\_set<string> dict(wordList.begin(), wordList.end());

        b = beginWord;

        queue<string> q;

        int k = beginWord.size();

        q.push({beginWord});

        mp[beginWord] = 0;

        while (!q.empty())  // Step 1

        {

            int n = q.size();

            while (n--)

            {

                string t = q.front();

                q.pop();

                int x = mp[t] + 1;

                for (int i = 0; i < k; i++)

                {

                    string temp = t;

                    for (char ch = 'a'; ch <= 'z'; ch++)

                    {

                        temp[i] = ch;

                        if (!mp.count(temp) && dict.count(temp))

                            mp[temp] = x, q.push(temp);

                    }

                }

            }

        }

        if (mp.count(endWord))

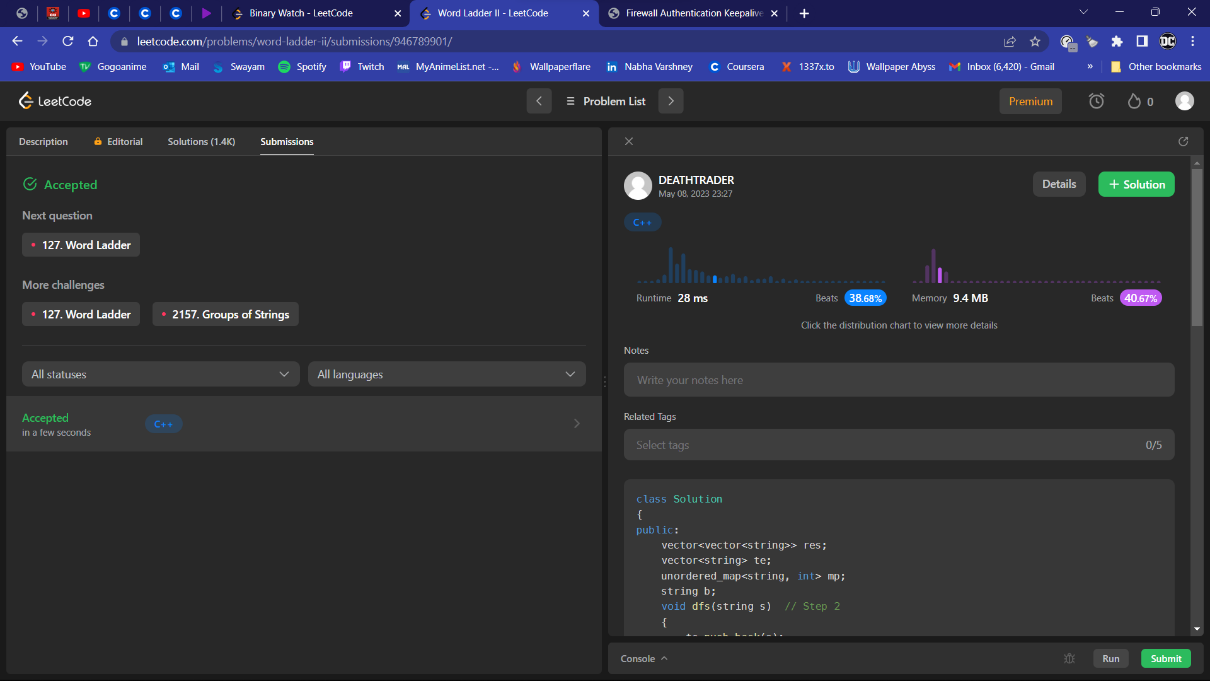
            dfs(endWord);

        return res;

    }

};

## Output –

****