



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Experiment-3.2

Student Name: Nabha Varshney

UID: 20BCS4995

Branch: CSE

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

Semester: 6th

Date of Performance: 03rd May 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.

1) 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 )
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
{
    temp += to_string(hours) + ":";

    if(minute < 10)
    {
        temp += "0";
    }
    temp += to_string(minute);
    result.push_back(temp);
}

}

return result;
}
};
```

Output -

The screenshot shows a web browser with multiple tabs, including 'Binary Watch - LeetCode', 'Word Ladder II - LeetCode', and 'Firewall Authentication Keepalive'. The active page is the LeetCode submission page for the 'Binary Watch' problem. The page layout includes a top navigation bar with 'Problem List' and 'Premium' options. The main content area is divided into two columns. The left column shows the problem status as 'Accepted' with a green checkmark, and lists 'Next question' (402. Remove K Digits) and 'More challenges' (17. Letter Combinations of a Phone Number, 191. Number of 1 Bits). The right column displays the submission details for user 'DEATHTRADER', including a distribution chart showing 'Runtime 2 ms', 'Beats 47.67%', and 'Memory 6.4 MB'. Below the chart, there is a 'Notes' section and a 'Related Tags' section. At the bottom, a code editor shows a C++ solution snippet for the 'readBinaryWatch' function.

```
class Solution
{
public:
    vector<string> readBinaryWatch(int num)
    {
        vector<string> result;

        for(int hours = 0; hours < 12; hours++)
        {
            for(int minutes = 0; minutes < 60; minutes++)
            {
                if((hours * 60 + minutes) % 60 == num)
                {
                    result.push_back(to_string(hours) + ":" + to_string(minutes));
                }
            }
        }

        return result;
    }
};
```

2) 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;
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
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 –

The screenshot displays the LeetCode interface for the 'Word Ladder II' problem. The submission status is 'Accepted'. The distribution chart shows a runtime of 28 ms and memory usage of 9.4 MB. The C++ code for the solution is visible, featuring a BFS approach to find the shortest transformation sequence from 'beginWord' to 'endWord' using words in the dictionary.