



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Experiment 9

Student Name: Aman Gokul

UID: 20BCS5449

Branch: BE CSE

Section/Group: 607/A

Semester: 6th

Date of Performance: 05/05/2023

Subject Name: CC-2 Lab

Subject Code: 20CSP-351

1. Aim/Overview of the practical:

Binary Watch

A binary watch has 4 LEDs on the top to represent the hours (0-11), and 6 LEDs on the bottom to represent the minutes (0-59). Each LED represents a zero or one, with the least significant bit on the right.

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

2. Apparatus / Simulator Used:

- Windows 7 or above
- Google Chrome

3. Objective:

- To understand the concept of Backtracking.

Backtracking is an algorithmic technique for solving problems recursively by trying to build a solution incrementally, one piece at a time, removing those solutions that fail to satisfy the constraints of the problem at any point in time (by time, here, is referred to the time elapsed till reaching any level of the search tree). Backtracking can also be said as an improvement to the brute force approach. So basically, the idea behind the backtracking technique is that it searches for a solution to a problem among all the available options. Initially, we start the backtracking from one possible option and if the problem is solved with that selected option then we return the solution else we backtrack and select another option from the remaining available options. There also might be a case where none of the options will give you the solution and hence we understand that backtracking won't give any solution to that particular problem. We can also say that backtracking is a form of recursion. This is because the process of finding the solution from the various option available is repeated recursively until we don't find the solution or we reach the final state. So we can conclude that backtracking at every step eliminates those choices that cannot give us the solution and proceeds to those choices that have the potential of taking us to the solution.

4. Code:

```
class Solution {
    public List<String> readBinaryWatch(int num) {
        List<String> result = new ArrayList<>();
        for (int hh = 0; hh < 12; hh++)
            for (int mn = 0; mn < 60; mn++)
                if (aux(hh, mn, num))
                    if (mn < 10)
                        result.add(String.format("%d:0%d", hh, mn));
                    else
```

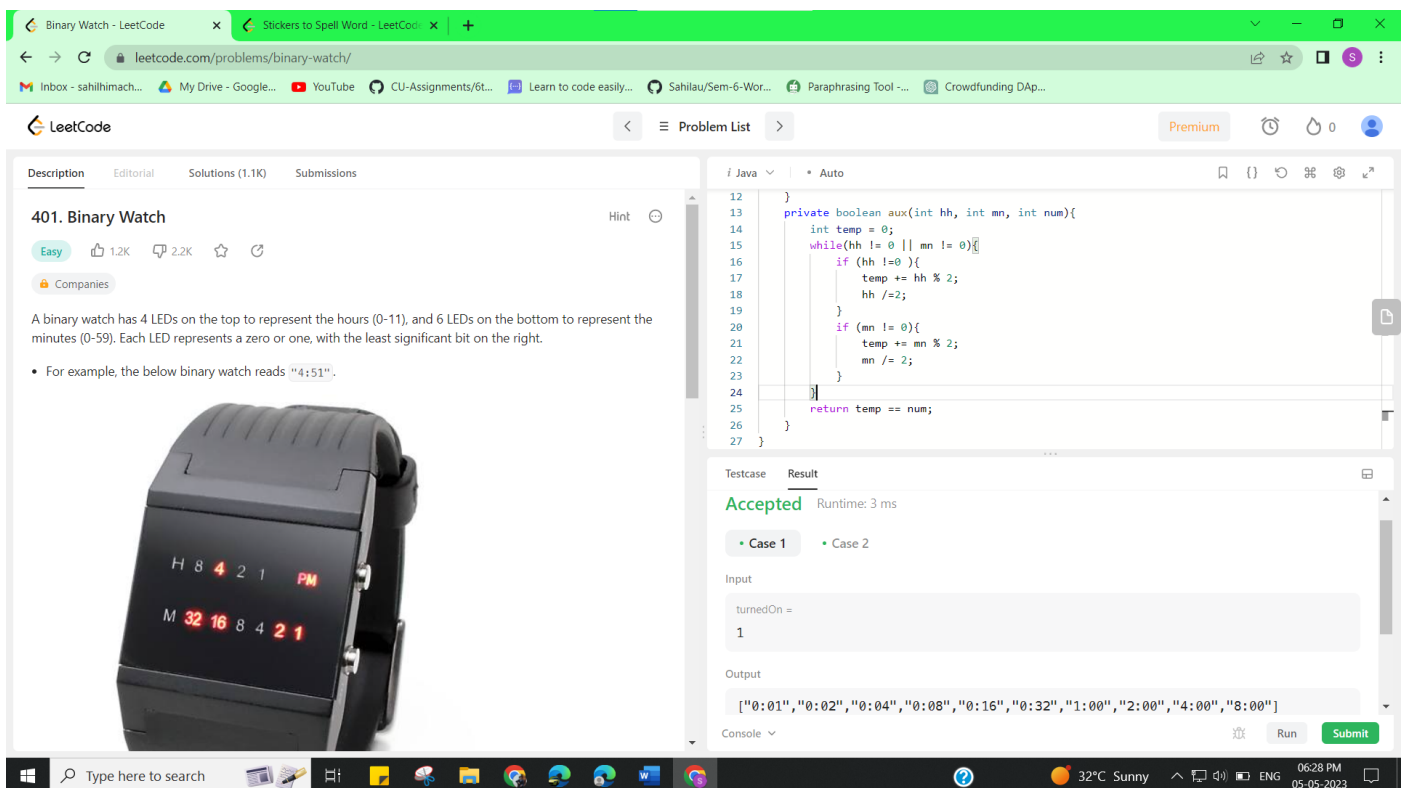
```

        result.add(String.format("%d:%d", hh, mn));
    }
    return result;
}

private boolean aux(int hh, int mn, int num){
    int temp = 0;
    while(hh != 0 || mn != 0){
        if (hh != 0){
            temp += hh % 2;
            hh /= 2;
        }
        if (mn != 0){
            temp += mn % 2;
            mn /= 2;
        }
    }
    return temp == num;
}
}

```

4. Result/Output/Writing Summary:



The screenshot displays the LeetCode interface for problem 401, "Binary Watch". The problem description states: "A binary watch has 4 LEDs on the top to represent the hours (0-11), and 6 LEDs on the bottom to represent the minutes (0-59). Each LED represents a zero or one, with the least significant bit on the right." An example shows a binary watch displaying "H 8 4 2 1 PM" and "M 32 16 8 4 2 1".

The Java code in the editor implements the solution:

```

12 }
13 private boolean aux(int hh, int mn, int num){
14     int temp = 0;
15     while(hh != 0 || mn != 0){
16         if (hh != 0){
17             temp += hh % 2;
18             hh /= 2;
19         }
20         if (mn != 0){
21             temp += mn % 2;
22             mn /= 2;
23         }
24     }
25     return temp == num;
26 }
27 }

```

The test results show "Accepted" with a runtime of 3 ms. The input is "turnedOn = 1" and the output is an array of time strings: ["0:01", "0:02", "0:04", "0:08", "0:16", "0:32", "1:00", "2:00", "4:00", "8:00"].



The screenshot shows a LeetCode submission for the problem 'Binary Watch' (problem ID 944953638). The submission is marked as 'Accepted'. The user's profile is 'sahilhimachal1806', dated May 05, 2023, 18:29. The submission is in Java. Performance metrics are displayed: Runtime 7 ms, Beats 66.60%, Memory 42.7 MB, and Beats 44.2%. A distribution chart is shown for the 'Beats' metric. The code is as follows:

```
class Solution {
    public List<String> readBinaryWatch(int num) {
        List<String> result = new ArrayList<>();
        for (int hh = 0; hh < 12; hh++)
            for (int mn = 0; mn < 60; mn++)
                if (aux(hh, mn, num))
                    if (mn < 10)
                        result.add(String.format("%d:%02d", hh, mn));
    }
}
```

Experiment 9.2

1. Aim/Overview of the practical:

Stickers to Spell Word

We are given n different types of stickers. Each sticker has a lowercase English word on it. You would like to spell out the given string target by cutting individual letters from your collection of stickers and rearranging them. You can use each sticker more than once if you want, and you have infinite quantities of each sticker.

Return the minimum number of stickers that you need to spell out target. If the task is impossible, return -1.

<https://leetcode.com/problems/stickers-to-spell-word/>

2. Apparatus / Simulator Used:

- Windows 7 or above
- Google Chrome

3. Objective:

- To understand the concept of Backtracking.

Backtracking is an algorithmic technique for solving problems recursively by trying to build a solution incrementally, one piece at a time, removing those solutions that fail to satisfy the constraints of the problem at any point in time (by time, here, is referred to the time elapsed till reaching any level of the search tree). Backtracking can also be said as an improvement to the brute force approach. So basically, the idea behind the backtracking technique is that it searches for a solution to a problem among all the available options. Initially, we start the backtracking from one possible option and if the problem is solved with that selected option then we return the solution else we backtrack and select another option from

the remaining available options. There also might be a case where none of the options will give you the solution and hence we understand that backtracking won't give any solution to that particular problem. We can also say that backtracking is a form of recursion. This is because the process of finding the solution from the various option available is repeated recursively until we don't find the solution or we reach the final state. So we can conclude that backtracking at every step eliminates those choices that cannot give us the solution and proceeds to those choices that have the potential of taking us to the solution.

4. Code:

```
class Solution {
public:
    int minStickers(vector<string>& stickers, string target) {
        vector<vector<int>>> sticker_counts(stickers.size(), vector<int>(26));
        unordered_map<string, int> dp;
        for (int i = 0; i < stickers.size(); ++i) {
            for (const auto& c : stickers[i]) {
                ++sticker_counts[i][c - 'a'];
            }
        }
        dp[""] = 0;
        return minStickersHelper(sticker_counts, target, &dp);
    }

private:
    int minStickersHelper(const vector<vector<int>>>& sticker_counts, const string& target,
        unordered_map<string, int> *dp) {
        if (dp->count(target)) {
            return (*dp)[target];
        }
        int result = numeric_limits<int>::max();
        vector<int> target_count(26);
        for (const auto& c : target) {
            ++target_count[c - 'a'];
        }
        for (const auto& sticker_count : sticker_counts) {
            if (sticker_count[target[0] - 'a'] == 0) {
                continue;
            }
            string new_target;
            for (int i = 0; i < target_count.size(); ++i) {
                if (target_count[i] - sticker_count[i] > 0) {
                    new_target += string(target_count[i] - sticker_count[i], 'a' + i);
                }
            }
            if (new_target.length() != target.length()) {
                int num = minStickersHelper(sticker_counts, new_target, dp);
                if (num != -1) {
                    result = min(result, 1 + num);
                }
            }
        }
        (*dp)[target] = (result == numeric_limits<int>::max()) ? -1 : result;
        return (*dp)[target];
    }
};
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

}};

5. Result/Output/Writing Summary:

Stickers to Spell Word - LeetCode

leetcode.com/problems/stickers-to-spell-word/

691. Stickers to Spell Word

Hard 964 77

Companies

We are given n different types of stickers. Each sticker has a lowercase English word on it.

You would like to spell out the given string `target` by cutting individual letters from your collection of stickers and rearranging them. You can use each sticker more than once if you want, and you have infinite quantities of each sticker.

Return the minimum number of stickers that you need to spell out `target`. If the task is impossible, return `-1`.

Note: In all test cases, all words were chosen randomly from the 1000 most common US English words, and `target` was chosen as a concatenation of two random words.

Example 1:

Input: `stickers = ["with","example","science"], target = "thehat"`
Output: 3
Explanation: We can use 2 "with" stickers, and 1 "example" sticker. After cutting and rearrange the letters of those stickers, we can form the target "thehat". Also, this is the minimum number of stickers necessary to form the target string.

```
private:
int minStickersHelper(const vector<vector<int>>& sticker_counts, const string& target,
                    unordered_map<string, int> *dp) {
    if (dp->count(target)) {
        return (*dp)[target];
    }
    int result = numeric_limits<int>::max();
    vector<int> target_count(26);
    for (const auto& c : target) {
        ++target_count[c - 'a'];
    }
    for (const auto& sticker_count : sticker_counts) {
        if (sticker_count[target[0] - 'a'] == 0) {
            continue;
        }
        string new_target;
```

Testcase Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

stickers =
["with","example","science"]

target =
"thehat"

Console Run Submit

Stickers to Spell Word - LeetCode

leetcode.com/problems/stickers-to-spell-word/submissions/944955422/

Accepted

Next question

692. Top K Frequent Words

More challenges

383. Ransom Note

All statuses All languages

Accepted a few seconds ago C++

sahilhimachal1806 May 05, 2023 18:33

C++

Runtime 52 ms Beats 78.23% Memory 12 MB Beats 69%

Click the distribution chart to view more details

Notes

Write your notes here

Related Tags

Select tags 0/5

```
class Solution {
public:
    int minStickers(vector<string>& stickers, string target) {
        vector<vector<int>> sticker_counts(stickers.size(), vector<int>(26));
        unordered_map<string, int> dp;
        for (int i = 0; i < stickers.size(); ++i) {
            for (const auto& c : stickers[i]) {
                sticker_counts[i][c - 'a']++;
            }
        }
        return minStickersHelper(sticker_counts, target, &dp);
    }
};
```

Console Run Submit

Learning outcomes (What I have learnt):



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

- Learned the concept of Backtracking.