Experiment-3.3

Student Name: Jaiswal Mukund **UID: 21BCS3407**
Branch: CSE **Section/Group: 602-B**
Semester: 5th **Date of Performance: 2/11/23**
Subject Name: AP-I **Subject Code: 21CSP-314**

Aim- Branch and Bound: Implement the problems based on Branch and Bound

Objectives- The objective of this experiment is to understand the concept of branch and bound.

Problem1: https://www.hackerrank.com/challenges/marcs-cakewalk/problem

Problem2:https://www.hackerrank.com/challenges/one-week-preparation-kit-grid-challenge/problem

Description-

Branch and bound is an algorithm design paradigm which is generally used for solving combinatorial optimization problems. These problems are typically exponential in terms of time complexity and may require exploring all possible permutations in worst case. The Branch and Bound Algorithm technique solves these problems relatively quickly.

This algorithm heavily depends on efficiently estimating the search space's lower and upper limits of branches. It degenerates to an exhaustive or thoroughgoing search if no limits are found. The branch and bound method solves these complex problems relatively faster. In most discrete optimization problems, the BnB method is a reliable choice to solve the issue.

Code:

1.

Change Theme Language C++14

```cpp
#include <bits/stdc++.h>

using namespace std;

int main(){
    int n;
    cin >> n;
    vector<int> calories(n);
    for(int calories_i = 0; calories_i < n; calories_i++){
       cin >> calories[calories_i];
    }
    // your code goes here
    sort(calories.begin(),calories.end());
    reverse(calories.begin(),calories.end());
    long long temp=1,ans=0;
    for(int i=0;i<n;i++)
    {
        ans+=calories[i]*temp;
        temp*=2;
    }
    printf("%lld\n",ans);
    return 0;
}
```

2.

Change Theme Language C++14

```cpp
#include <bits/stdc++.h>

using namespace std;

string ltrim(const string &);
string rtrim(const string &);

string gridChallenge(vector<string> grid) {
for (int i = 0; i < grid.size(); i++) {

        sort(grid[i].begin(), grid[i].end());
    }


    for (int j = 0; j < grid[0].size(); j++) {
        for (int i = 1; i < grid.size(); i++) {

            if (grid[i][j] < grid[i - 1][j]) {
                return "NO";
            }
        }
    }

    return "YES";
}

int main()
{
```

```
    }

    fout.close();

    return 0;
}

string ltrim(const string &str) {
    string s(str);

    s.erase(
        s.begin(),
        find_if(s.begin(), s.end(), not1(ptr_fun<int, int>(isspace)))
    );

    return s;
}

string rtrim(const string &str) {
    string s(str);

    s.erase(
        find_if(s.rbegin(), s.rend(), not1(ptr_fun<int, int>(isspace))).base(),
        s.end()
    );

    return s;
}

```

<u>Outcome-</u>

Problem 1 outcome

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Sample Test case 1

Input (stdin) Download

```
3
1 3 2
```

Your Output (stdout)

```
11
```

Expected Output Download

```
11
```

Problem 2 Outcome

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Sample Test case 1

Input (stdin) Download

```
1
5
eabcd
fghij
olkmn
trpqs
xywuv
```

Your Output (stdout)

```
YES
```

Learning Outcomes-

1. Learnt about the branch and bound.
2. Learnt about different algorithms to solve branch and bound problems.