Design and Analysis of Algorithms (CS206)

LAB Assignment – 7

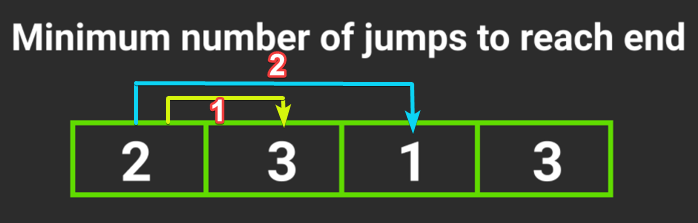
1.1. (T) Problem Statement:

**“Will Doraemon Help?”**

As Usual, **Nobita** woke Up *Late for School* and Got Ready in Hurry. But, His **Mom**, wanted to teach Nobita a Lesson, So with Help of Smart **Suneo**, Designed a Problem to Reach School.

The **Path** from **Nobita's Home** to **School**, is Broken Down into Consecutive **Spring Jumper Tiles**, Which has Power Capacity of '**p**'. If Person stands on this Tile, than He can jump to Any Tile from 1 to ‘p’ ahead of current Tile. [See Image for Better Understanding]



**Time Taken** to Move from Current Tile to Next Tile takes **1** second.

Since, Nobita is in Hurry, He came Crying to **Doraemon** [You] to solve this Problem, So that he can reach the School as **Early as Possible**.

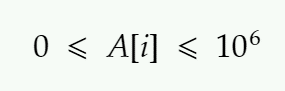
You need to Help Nobita, to Tell him if **He can Reach School or Not** (-1).

If He can **Reach School**, You need to inform him the **Minimum Time Taken** to Reach School.

Input Format:

Input File containing Integers, each representing the **Power** of Particular Tile

Constraints:



Output Format:

Print “-1”, If Nobita wont to be able to Reach School.

If it’s Possible to Reach School, Then Print the **Minimum** Time Taken to Reach School.

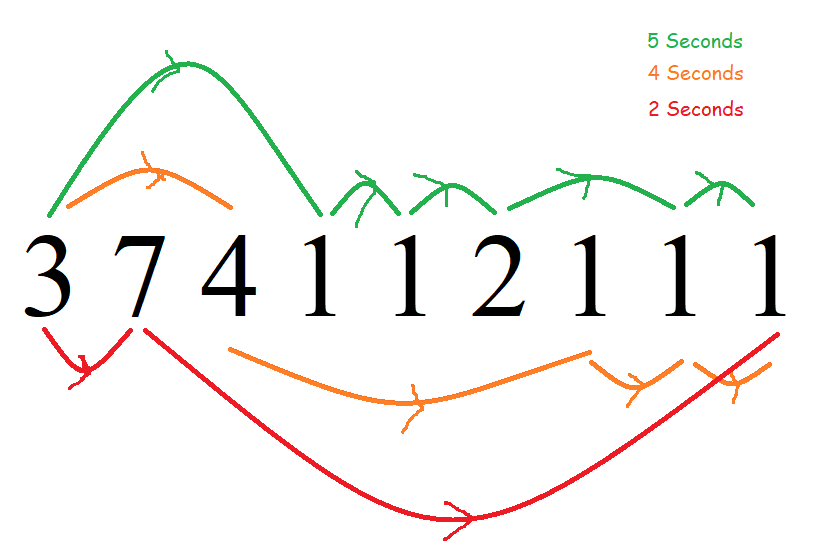
*Sample Input:*

3 7 4 1 1 2 1 1 1

*Sample Output:*

**2**

*Explanation:*



The Path {3 ->7->1} will take Minimum Time

*Sample Input 2:*

3 2 1 1 0 7 9 1

*Sample Output 2:*

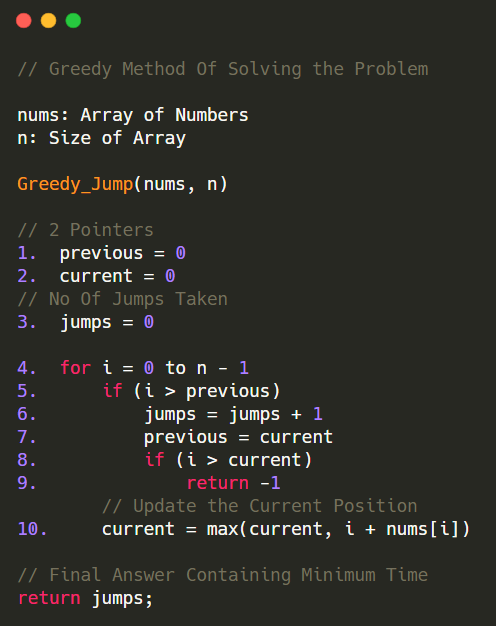
**-1**

*Explanation:*

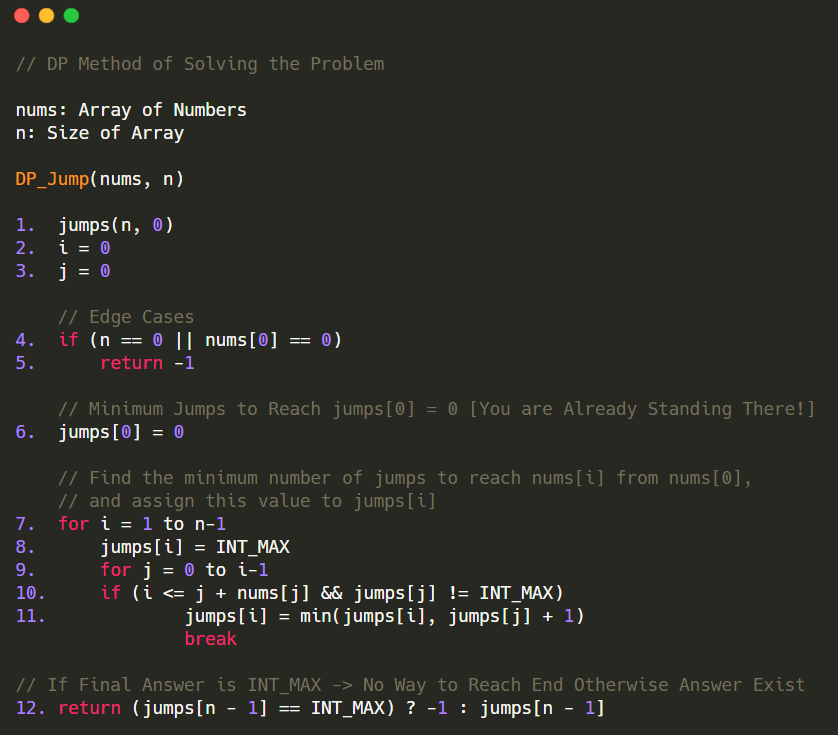
No Matter What You Try You Won’t to be able to Reach the End

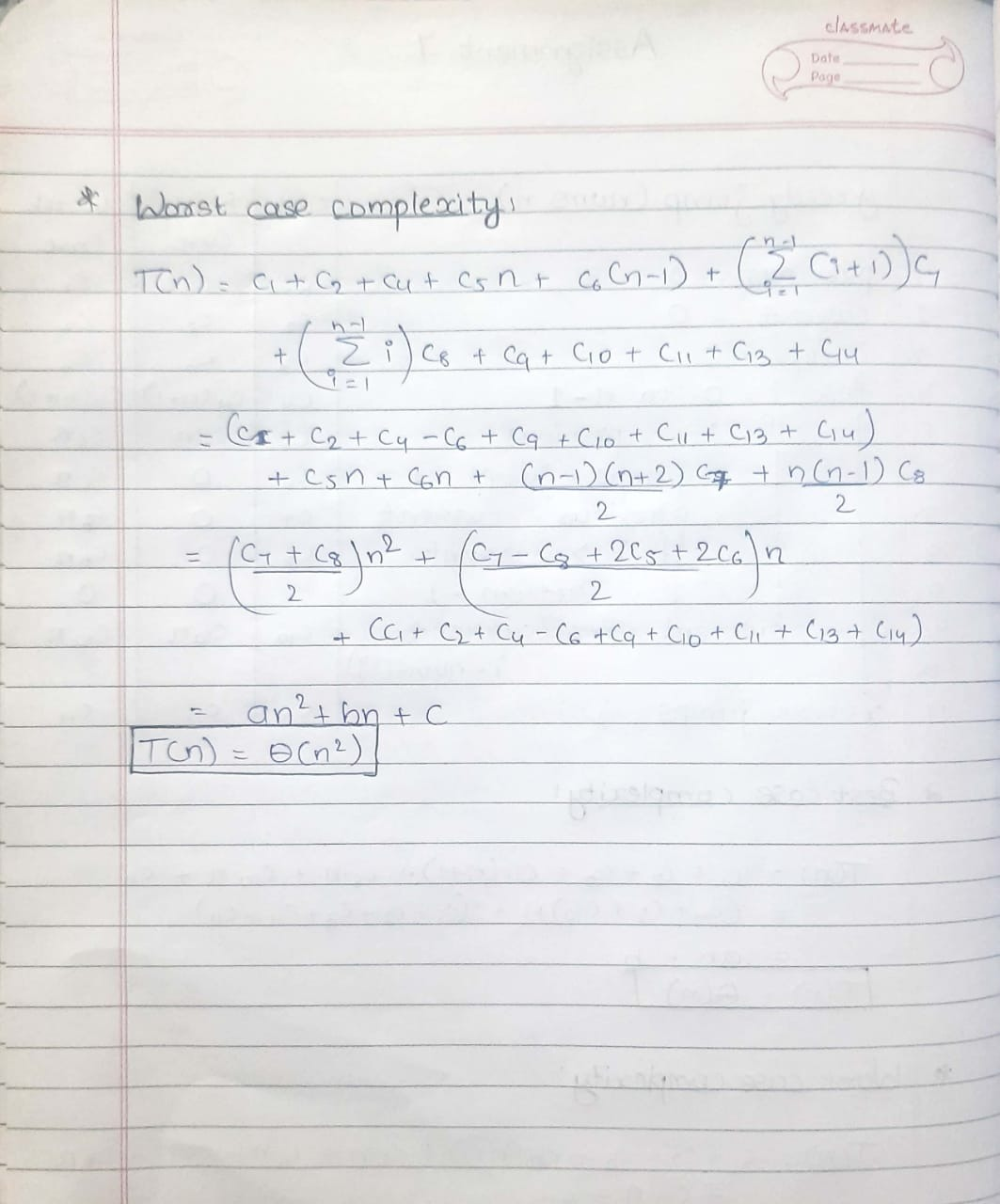
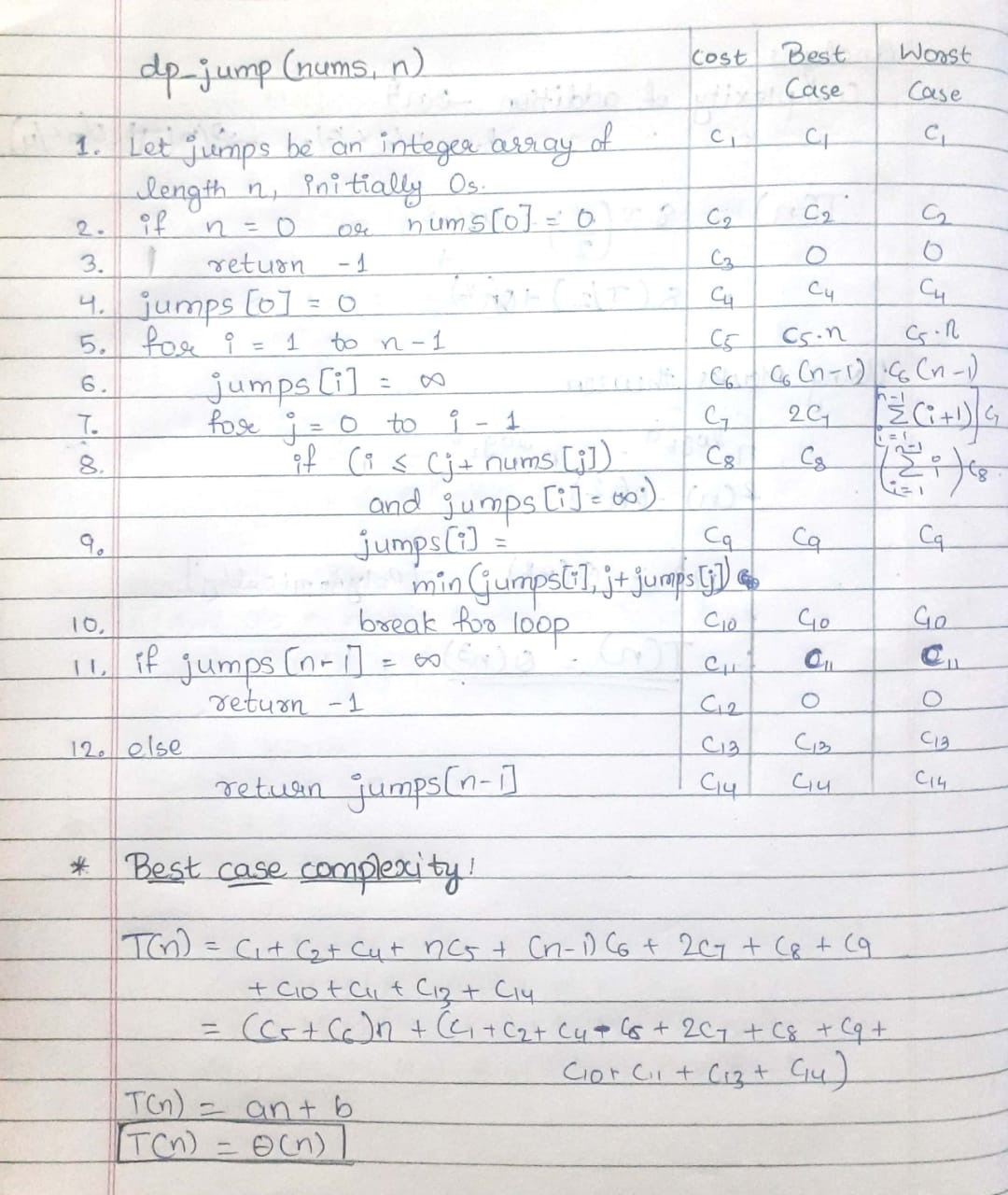
1.2. (T) Write pseudocodes to design the algorithms for the above mentioned computational problem using the Greedy approach as well as dynamic programming.

**Pseudo Code [Greedy]**

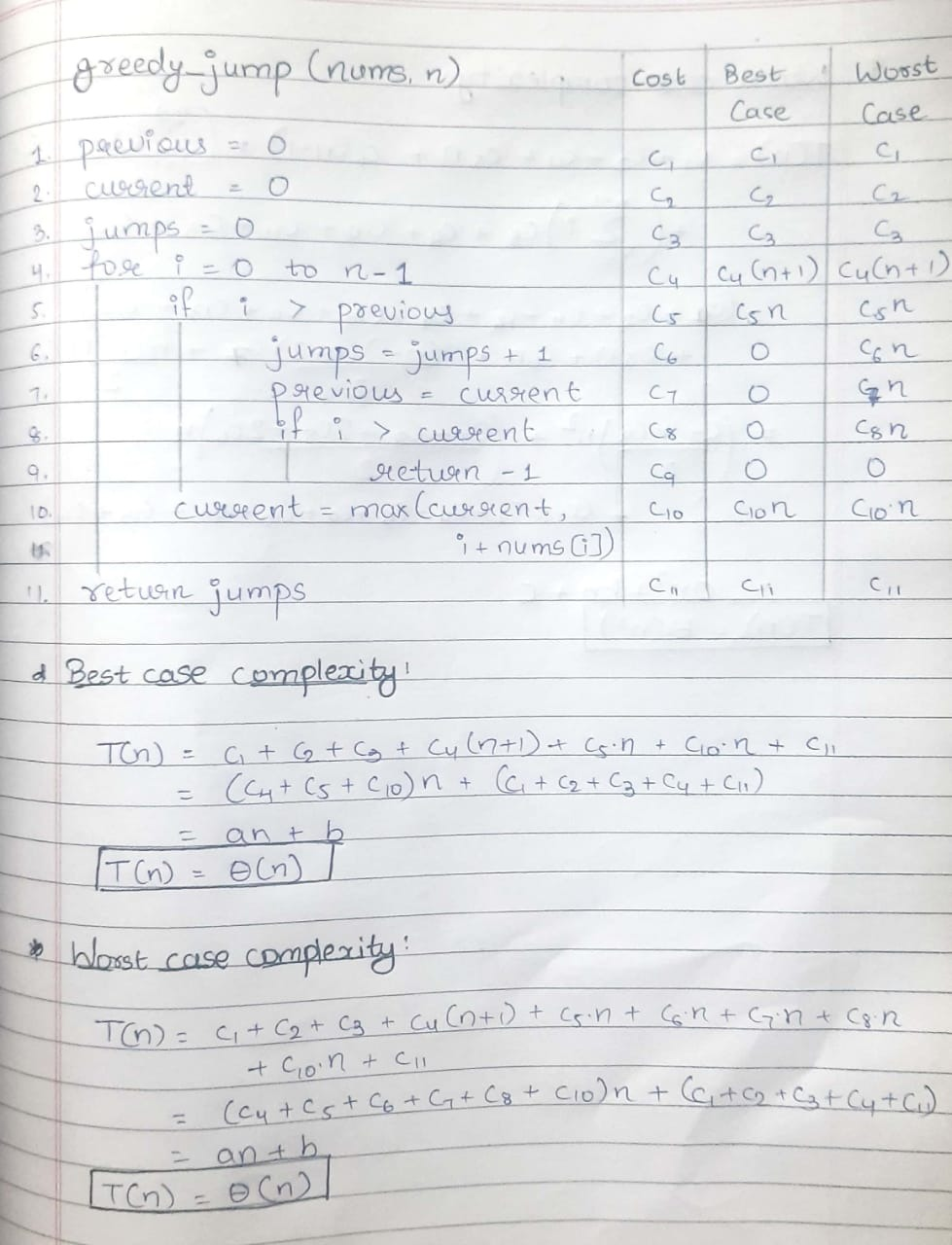


**Pseudo Code [DP]**



1.3. (T) Analyze the time complexity of above algorithms.

Greedy Approach:



1.4. (L) Provide the details of Hardware/Software you used to implement algorithms.

Hardware Details:

|  |  |
| --- | --- |
| PARAMETER | LAPTOP CONFIGURATION |
| Operating System | Microsoft Windows **10** .0.19042 |
| Processor | Intel(R) Core(TM) i5-10210U [Core **i5 10th Gen**] |
| CPU | **1.60GHz**, 2112 Mhz, **4** Core(s), 8 Logical Processor(s) |
| System Type | x64-based PC [**64 Bit**] |
| RAM | **8.00** GB |
| Hard Drive/SSD | 512 GB **SSD** |

Software Used:

|  |  |
| --- | --- |
| PARAMETER | LAPTOP CONFIGURATION |
| Code Editor | **Visual Studio** Code [Version 1.52] |
| Compiler | gcc (MinGW.org **GCC-8.2.0-5**) 8.2.0 |
| Time | Measured using **chrono** Library in C++ |
| Programming Language Used | **C++** |

1.5. (L) Implement the above algorithms and submit the code (complete programs).

Code:

*// HEADERS AND NAMESPACE*

*#include* <bits/stdc++.h>

*// INSTEAD OF ALL THESE*

*#include* <iostream>

*// For Creating File*

*#include* <fstream>

*#include* <vector>

*// For set - precision*

*#include* <iomanip>

*// For Time Calculation*

*#include* <chrono>

*// For File Name and Output File Name*

*#include* <string>

using namespace std;

using namespace std::chrono;

*// COMMONLY USED TYPES*

typedef long long ll;

typedef vector<ll> vll;

*#define* max(a, b) (a < b ? b : a)

*#define* min(a, b) ((a > b) ? b : a)

*// Greedy Method Of Solving the Problem*

ll Greedy\_Jump(vll nums, ll n)

{

    int previous = 0;

    int current = 0;

    int jumps = 0;

*for* (int i = 0; i < n; i++)

    {

*if* (i > previous)

        {

            jumps = jumps + 1;

            previous = current;

*if* (i > current)

*return* -1;

        }

        current = max(current, i + nums[i]);

    }

*return* jumps;

}

*// DP Method of Solving the Problem*

ll DP\_Jump(vll nums, ll n)

{

    vll jumps(n, 0);

    int i, j;

*// Edge Cases*

*if* (n == 0 || nums[0] == 0)

*return* -1;

*// Minimum Jumps to Reach jumps[0] = 0 [You are Already Standing There!]*

    jumps[0] = 0;

*// Find the minimum number of jumps to reach nums[i] from nums[0],*

*// and assign this value to jumps[i]*

*for* (i = 1; i < n; i++)

    {

        jumps[i] = INT\_MAX;

*for* (j = 0; j < i; j++)

        {

*if* (i <= j + nums[j] && jumps[j] != INT\_MAX)

            {

                jumps[i] = min(jumps[i], jumps[j] + 1);

*break*;

            }

        }

    }

*// If Final Answer is INT\_MAX -> No Way to Reach End Otherwise Answer Exist*

*return* (jumps[n - 1] == INT\_MAX) ? -1 : jumps[n - 1];

}

int main()

{

*// For Read & Write from "Input File" and  Return Output to "Output" File*

    freopen("output.txt", "w", stdout);

*// EDIT THIS FILE NUMBER , LIMIT and Number of Times File Runs*

    int file\_no = 1;

    int limit = 10;

    int each\_file\_runs = 2;

*for* (; file\_no <= limit; file\_no++)

    {

        string inp\_file = "File";

        string num = to\_string(file\_no);

        string ext = ".txt";

        inp\_file += num;

        inp\_file += ext;

        ifstream File;

        File.open(inp\_file);

        vector<ll> arr;

        ll number;

*while* (!File.eof())

        {

            File >> number;

            arr.push\_back(number);

        }

        ll DP\_Duration = 0;

        ll Greedy\_Duration = 0;

        auto start = high\_resolution\_clock::now();

        auto end = high\_resolution\_clock::now();

        auto time\_taken = duration\_cast<nanoseconds>(end - start);

        ll n = arr.size();

*for* (int f = 0; f < each\_file\_runs; f++)

        {

*// DP*

            start = high\_resolution\_clock::now();

*// Function Here*

            ll dp\_ans = DP\_Jump(arr, n);

*// Function Ends here*

            end = high\_resolution\_clock::now();

            time\_taken = duration\_cast<nanoseconds>(end - start);

            DP\_Duration += time\_taken.count();

*// Greedy*

            start = high\_resolution\_clock::now();

*// Function Here*

            ll greedy\_ans = Greedy\_Jump(arr, n);

*// Function Ends here*

            end = high\_resolution\_clock::now();

            time\_taken = duration\_cast<nanoseconds>(end - start);

            Greedy\_Duration += time\_taken.count();

*// For Checking Purpose*

*if* (dp\_ans != greedy\_ans)

            {

                cout << inp\_file << endl;

                cout << "WRONG ANSWER!" << endl;

            }

        }

        cout << "--------------------------------------------------------" << endl;

        cout << inp\_file << endl;

        cout << "TIME TAKEN (DP): ";

        double avg = (double)DP\_Duration / (double)each\_file\_runs;

        avg \*= 1e-9;

        cout << fixed << avg << setprecision(9);

        cout << " seconds" << endl;

        cout << "TIME TAKEN (GREEDY): ";

        double avg2 = (double)Greedy\_Duration / (double)each\_file\_runs;

        avg2 \*= 1e-9;

        cout << fixed << avg2 << setprecision(9);

        cout << " seconds" << endl;

    }

*return* 0;

}

1.6. (L) Analyze the performance of both the implemented algorithms (performance of algorithms on your computers). Plot a graph.

DP METHOD

|  |  |  |
| --- | --- | --- |
| Sr. No. | No of Elements | DP (ms) |
| 1 | 1024 | 0.00 |
| 2 | 2048 | 12.683500 |
| 3 | 4096 | 31.872000 |
| 4 | 8192 | 118.410500 |
| 5 | 16384 | 391.449000 |
| 6 | 32768 | 1561.782000 |
| 7 | 65536 | 5659.320500 |
| 8 | 131072 | 22958.085000 |
| 9 | 262144 | 96446.760500 |
| 10 | 524288 | 353511.590500 |

T(n)≈

1.2075296762154...×10−6 n2 + 0.0435634n − 750.681

GREEDY METHOD

|  |  |  |
| --- | --- | --- |
| Sr. No. | No of Elements | Greedy (ms) |
| 1 | 1024 | 0 |
| 2 | 2048 | 0 |
| 3 | 4096 | 0.2472 |
| 4 | 8192 | 0 |
| 5 | 16384 | 0 |
| 6 | 32768 | 0.513 |
| 7 | 65536 | 1.5814 |
| 8 | 131072 | 2.5466 |
| 9 | 262144 | 1.6814 |
| 10 | 524288 | 5.395835 |

T(n)≈

0.637577 n − 18720.3

1.7. (L) Comparatively Analyze the performance of above algorithms and plot a graph.

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No. | No of Elements | Greedy (ms) | Divide and Conquer  (ms) |
| 1 | 1024 | 0 | 0 |
| 2 | 2048 | 0 | 12.6835 |
| 3 | 4096 | 0.2472 | 31.872 |
| 4 | 8192 | 0 | 118.4105 |
| 5 | 16384 | 0 | 391.449 |
| 6 | 32768 | 0.513 | 1561.782 |
| 7 | 65536 | 1.5814 | 5659.3205 |
| 8 | 131072 | 2.5466 | 22958.085 |
| 9 | 262144 | 1.6814 | 96446.7605 |



The **Most Optimal Solution** = Nobita can ask Doraemon for **Bamboo Copter** [OR **Anywhere Door**]

SUBMITTED BY:

|  |  |
| --- | --- |
| Sr. No. | Admission No. |
| 1 | U19CS011 |
| 2 | U19CS012 |
| 3 | U19CS049 |
| 4 | U19CS080 |