DAA WEEK3 SUBMISSION

Name: Davasam Karthikeya   Section: AIMLB   Reg No: 230962326

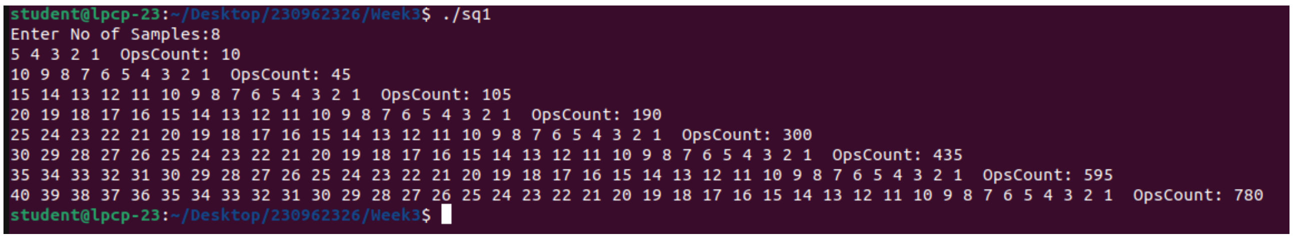
Date: 25/01/2025

1) Write a program to sort a set of integers using selection sort algorithm and analyze its time efficiency. Obtain the experimental result of order of growth. Plot the result for the best and worst case.

Code:

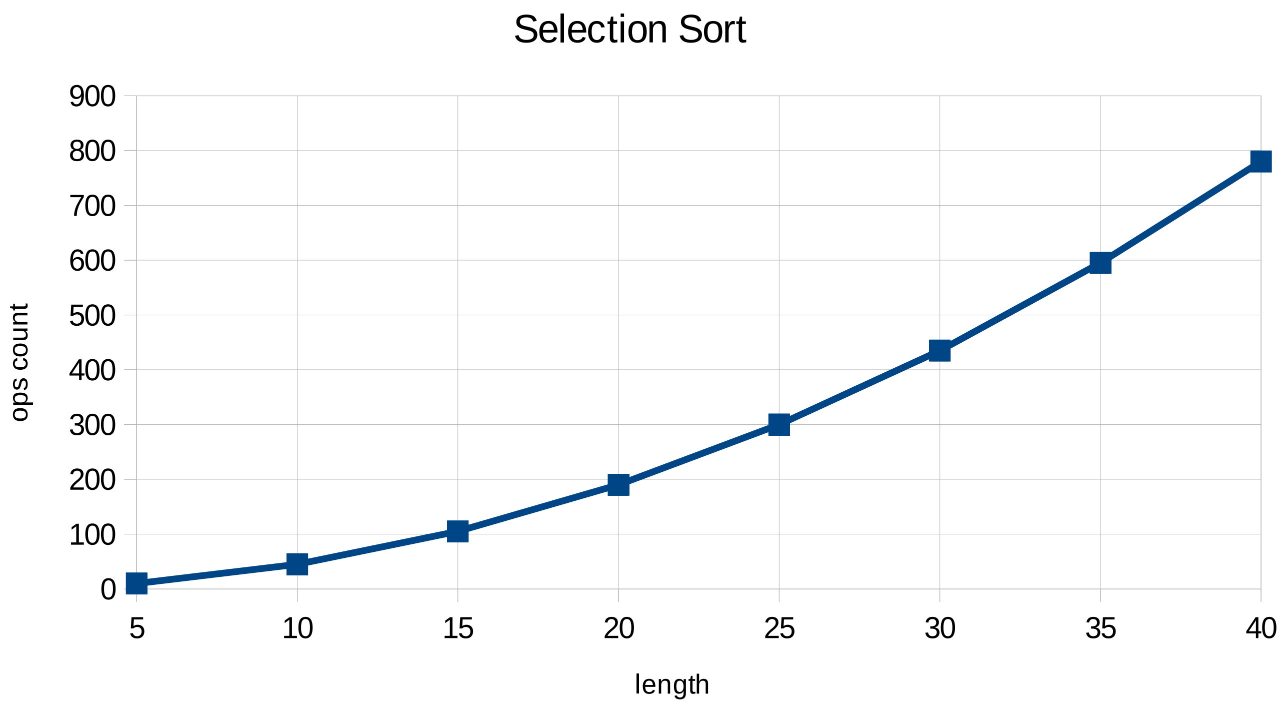
```
#include <stdio.h>
#include <stdlib.h>

void SelectionSort(int *Array,int length){
    int OpsCount = 0;
    for (int i = 0; i < length; i++)printf("%d ",Array[i]);
    for(int i = 0; i < length; i++){
        int small_index = i;
        for(int j = i+1; j < length; j++){
            if(Array[j]<Array[small_index])small_index=j;
            OpsCount++;
        }
        int temp = Array[i];
        Array[i] = Array[small_index];
        Array[small_index] = temp;
    }
    printf(" OpsCount: %d \n",OpsCount);
}


int main(){
    // int Arr[] = {5,4,3,2,1};
    // SelectionSort(&Arr[0],5);
    int N = 5;
    int NoOfSamples;printf("Enter No of Samples:");scanf("%d",&NoOfSamples);
    for(int i = 0; i < NoOfSamples;i++){
        int *Arr = (int*)malloc(N*sizeof(int));
        for(int j = 0;j<N;j++)Arr[j] = N-j;
        SelectionSort(Arr,N);
        N += 5;
    }

    return 0;
}
```

Sample Input/Output:



```
student@lpcp-23:~/Desktop/230962326/Week3$ ./sq1
Enter No of Samples:8
5 4 3 2 1  OpsCount: 10
10 9 8 7 6 5 4 3 2 1  OpsCount: 45
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1  OpsCount: 105
20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1  OpsCount: 190
25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1  OpsCount: 300
30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1  OpsCount: 435
35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1  OpsCount: 595
40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1  OpsCount: 780
student@lpcp-23:~/Desktop/230962326/Week3$
```

Graph:

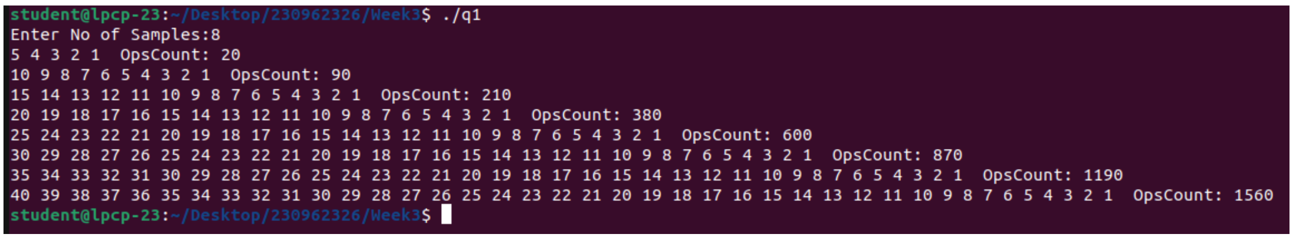


2) Write a program to sort set of integers using bubble sort. Analyze its time efficiency. Obtain the experimental result of order of growth. Plot the result for the best and worst case.

Code:

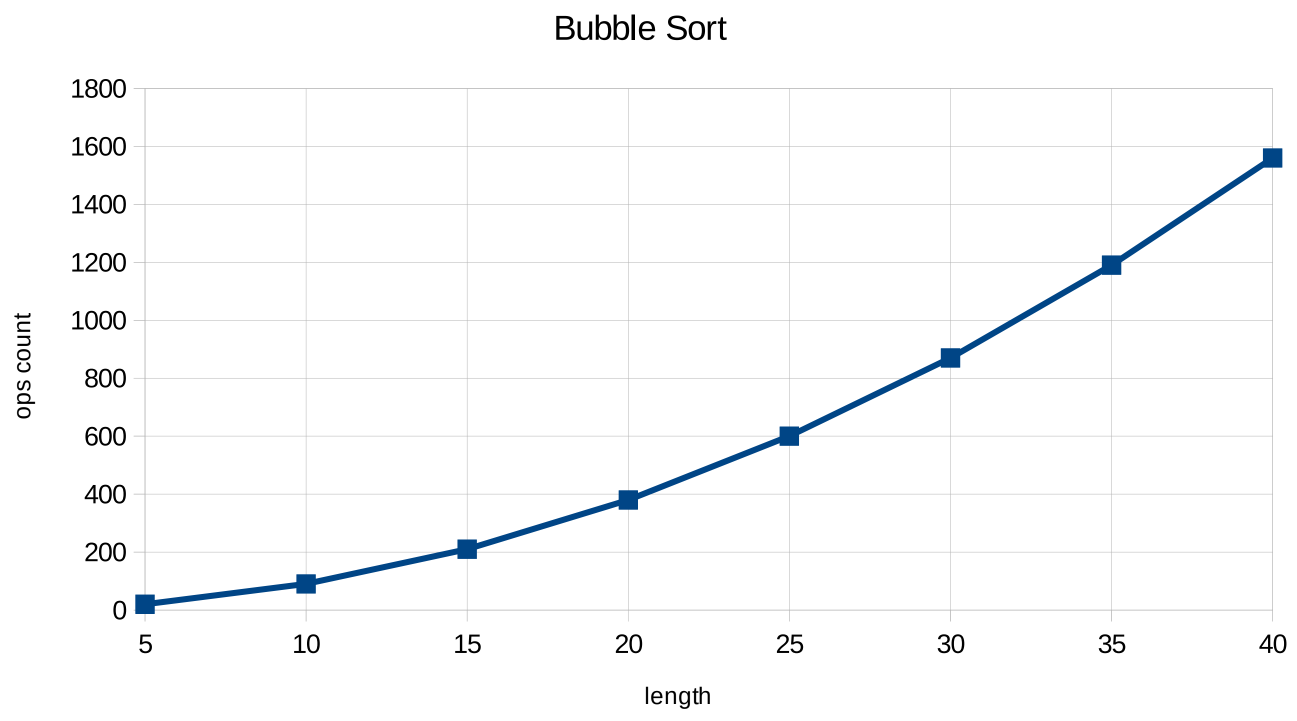
```
#include <stdio.h>
#include <stdlib.h>

void BubbleSort(int *Array,int length){
    int OpsCount = 0;
    for(int i = 0; i < length; i++)printf("%d ",Array[i]);
    for(int i = 0; i < length-1; i++){
        for (int j = 0; j < length-i-1; j++){
            OpsCount++;
            if(Array[j+1] < Array[j]){
                OpsCount++;
                int temp = Array[j+1];
                Array[j+1] = Array[j];
                Array[j] = temp;
            }
        }
    }
    printf(" OpsCount: %d \n",OpsCount);
}

int main(){
    int N = 5;
    int NoOfSamples;printf("Enter No of Samples:");scanf("%d",&NoOfSamples);
    for(int i = 0; i < NoOfSamples;i++){
        int *Arr = (int*)malloc(N*sizeof(int));
        for(int j = 0;j<N;j++)Arr[j] = N-j;
        BubbleSort(Arr,N);
        N += 5;
    }
    return 0;
}
```

Sample Input/Output:



```
student@lpcp-23:~/Desktop/230962326/Week3$ ./q1
Enter No of Samples:8
5 4 3 2 1  OpsCount: 20
10 9 8 7 6 5 4 3 2 1  OpsCount: 90
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1  OpsCount: 210
20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1  OpsCount: 380
25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1  OpsCount: 600
30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1  OpsCount: 870
35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1  OpsCount: 1190
40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1  OpsCount: 1560
student@lpcp-23:~/Desktop/230962326/Week3$
```

Graph:

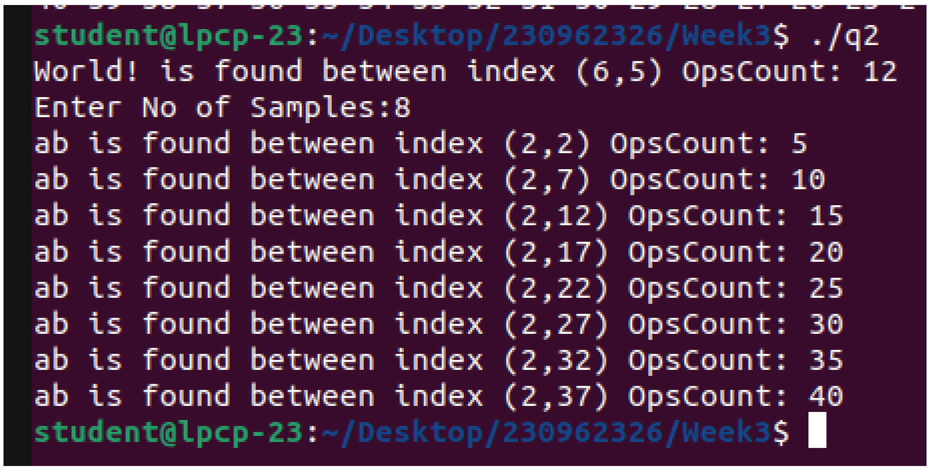


3) Write a program to implement brute-force string matching. Analyze its time efficiency.

Code:

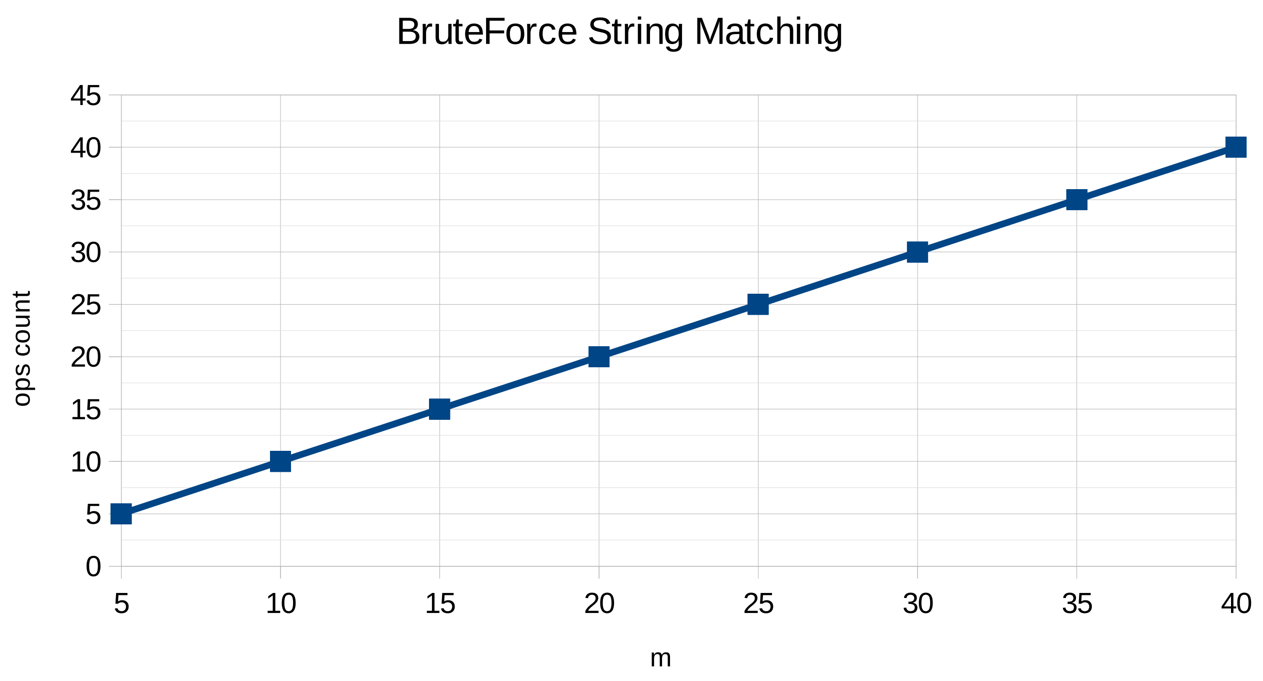
```
#include <stdio.h>
#include <string.h>

void BruteForceStringMatch(char string[],char search[]){
    int i = 0,j=0,OpsCount = 0;
    int len1 = strlen(string),len2 = strlen(search);
    while(i < len1 && j < len2){
        OpsCount++;
        if(string[i] == search[j]){
            i++;j++;
        }else{
            i++;
        }
    }
    if(j == len2)printf("%s is found between index (%d,%d) OpsCount: %d \
n",search,j,i-j-1,OpsCount);
    else printf("%s not found in %s \n",search,string);
}

int main(){
    int N = 5;
    int NoOfSamples;printf("Enter No of Samples:");scanf("%d",&NoOfSamples);
    for(int i = 0; i < NoOfSamples;i++){
        char string[N];
        for(int i = 0;i<N;i++)string[i] = 'a';
        string[N-1] = 'b';
        BruteForceStringMatch(string,"ab");
        N += 5;
    }
    return 0;
}
```

Sample Input/Output:



```
student@lpcp-23:~/Desktop/230962326/Week3$ ./q2
World! is found between index (6,5) OpsCount: 12
Enter No of Samples:8
ab is found between index (2,2) OpsCount: 5
ab is found between index (2,7) OpsCount: 10
ab is found between index (2,12) OpsCount: 15
ab is found between index (2,17) OpsCount: 20
ab is found between index (2,22) OpsCount: 25
ab is found between index (2,27) OpsCount: 30
ab is found between index (2,32) OpsCount: 35
ab is found between index (2,37) OpsCount: 40
student@lpcp-23:~/Desktop/230962326/Week3$
```

Graph:

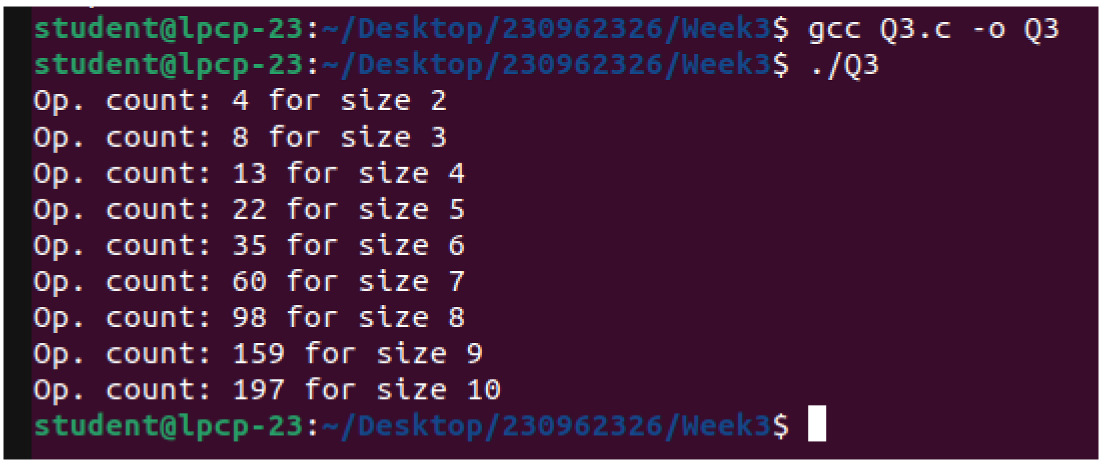


4) Write a program to implement solution to partition problem using brute-force technique and analyze its time efficiency theoretically. A partition problem takes a set of numbers and finds two disjoint sets such that the sum of the elements in the first set is equal to the second set. [Hint: You may generate power set]

Code:

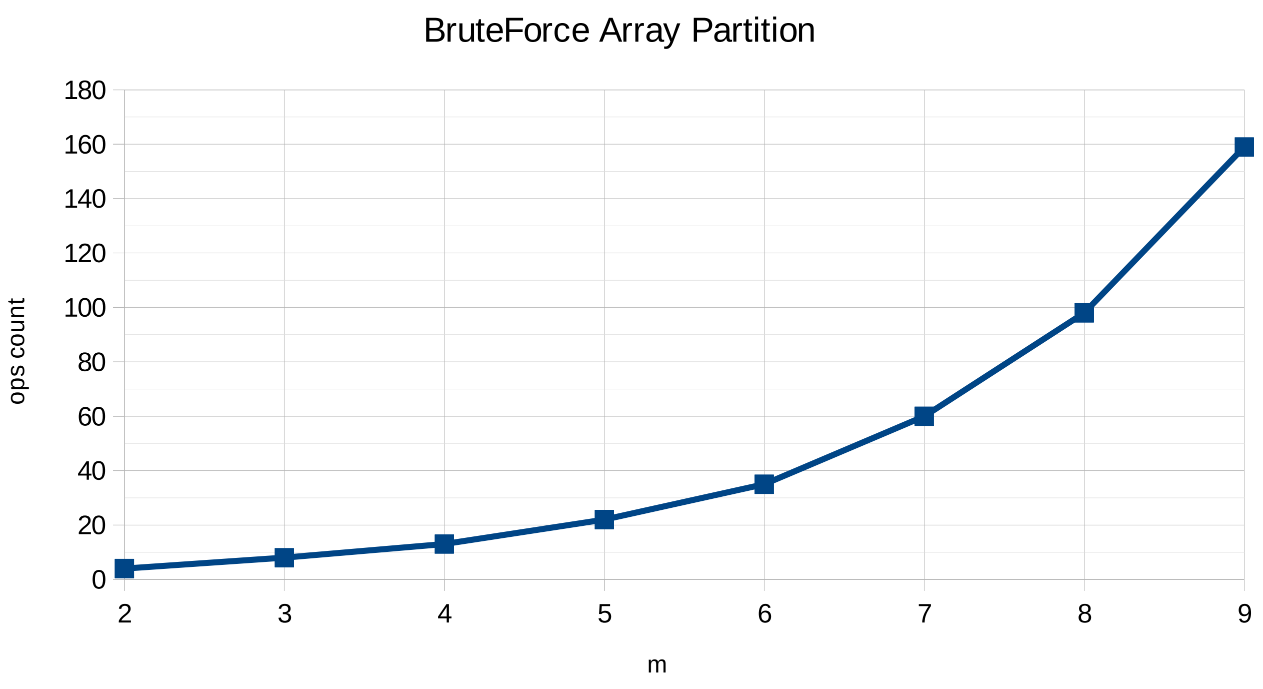
```
#include <stdio.h>
#include <stdlib.h>

char isPartition(int arr[], int len, int sum, int *opCount) {
    (*opCount)++;
    if(sum == 0) return 1;
    else if(len == 0) return 0;
    else if(arr[len - 1] > sum) return isPartition(arr, len - 1, sum, opCount);
    return isPartition(arr, len - 1, sum, opCount) || isPartition(arr, len - 1,
sum - arr[len - 1], opCount);
}
void main() {
    int opCount, sum, trCount = 50, *arr;
    for(int len = 2; len <= 10; len++) {
        opCount = 0;
        arr = calloc(len, sizeof(int));
        for(int trial = 0; trial < trCount; trial++) {
            sum = 0;
            for(int i = 0; i < len; i++) {
                arr[i] = rand() * trial * (i + 1);
                sum += arr[i];
            }
            for(int i = len - 1; i > 0; i--) {
                int w = rand() % i;
                int t = arr[i];
                arr[i] = arr[w];
                arr[w] = t;
            }
            isPartition(arr, len, sum / 2, &opCount);
        }
        free(arr);
        printf("Op. count: %d for size %d\n", opCount / trCount, len);
    }
}
```

Sample Input/Output:



```
student@lpcp-23:~/Desktop/230962326/Week3$ gcc Q3.c -o Q3
student@lpcp-23:~/Desktop/230962326/Week3$ ./Q3
Op. count: 4 for size 2
Op. count: 8 for size 3
Op. count: 13 for size 4
Op. count: 22 for size 5
Op. count: 35 for size 6
Op. count: 60 for size 7
Op. count: 98 for size 8
Op. count: 159 for size 9
Op. count: 197 for size 10
student@lpcp-23:~/Desktop/230962326/Week3$
```

Graph:



BruteForce Array Partition
ops count
m
0
20
40
60
80
100
120
140
160
180
2
3
4
5
6
7
8
9