Devam Desai
IIT2022035
DAA Assignment 4

In this entire assignment, for plotting I have used the following driver code:

```
void solve(int n){
   // cout << n << endl;
   vector<int> arr;
   int maxi = 1e5;

   for (int i = 0;i < n;i++){
      arr.push_back(rand()%(maxi));
   }
   int ans=0;

   auto start = high_resolution_clock::now();
   ans=QUESTION_SOLUTION(arr,n);

   auto stop = high_resolution_clock::now();
   auto duration = duration_cast<microseconds>(stop - start);
   cout << (duration).count() << endl;
}
```

Q1. In the first Question , we are going to compare between 2 approaches

1. bruteforce: Takes time O(n^2), space O(1)

Code:

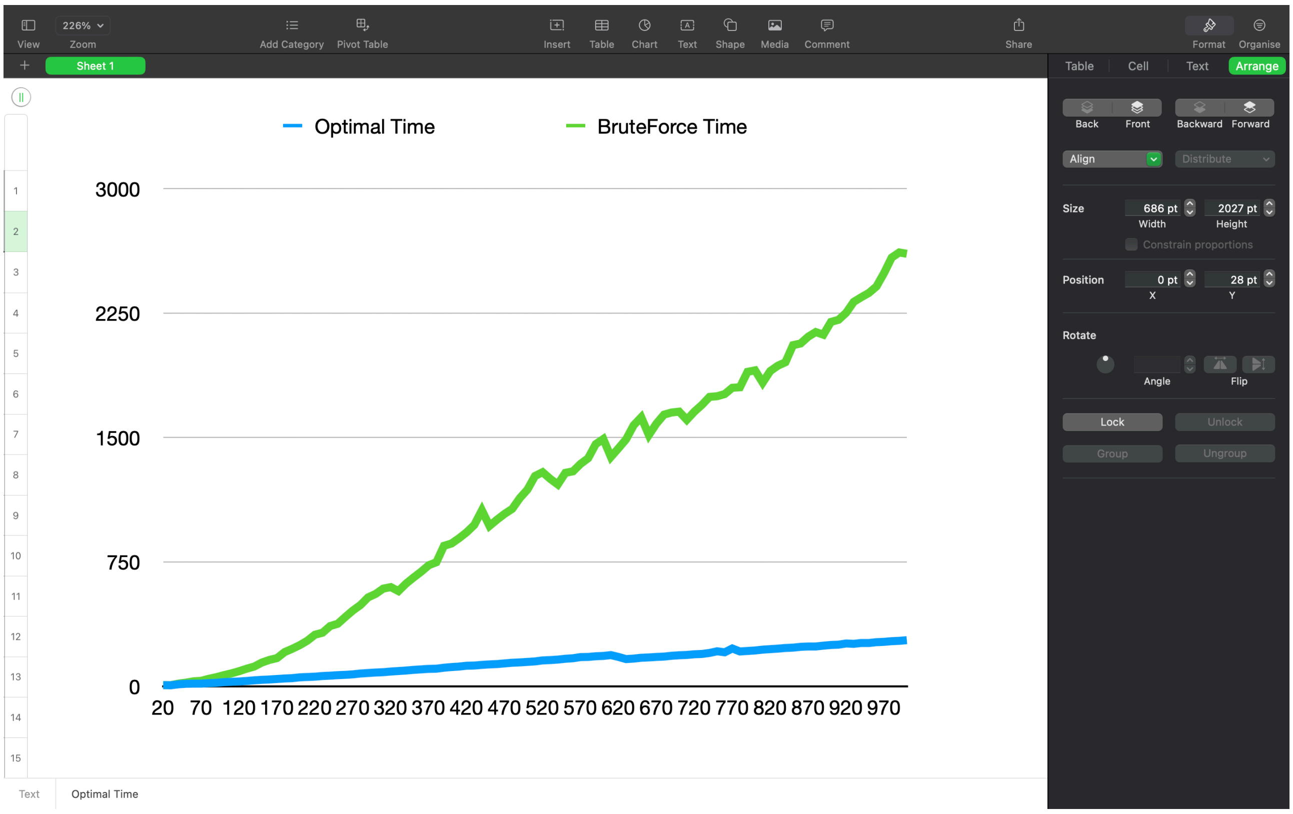
```
int countInversions(vint arr, int n){

   int res=0;
   for(int i=0;i<(n-1);i++){
      for(int j=i+1;j<n;j++){
         if(arr[i]>arr[j])
            res++;
      }
   }
   return res;
}
```

2. Using mergesort(Count and Merge)(Optimal): Time: O(nlog(n)) Space: O(n)

Code:

```
      int _mergeSort(vint &arr, vint &temp, int left, int right);
int merge(vint &arr, vint &temp, int left, int mid,
          int right);

// This function sorts the
// input array and returns the
// number of inversions in the array
int mergeSort(vint &arr, int array_size)
{
    vint temp(array_size);
    return _mergeSort(arr, temp, 0, array_size - 1);
}

int _mergeSort(vint &arr, vint &temp, int left, int right)
{
    int mid, inv_count = 0;
    if (right > left) {

        mid = (right + left) / 2;
        inv_count += _mergeSort(arr, temp, left, mid);
        inv_count += _mergeSort(arr, temp, mid + 1, right);

        inv_count += merge(arr, temp, left, mid + 1, right);
    }
    return inv_count;
}

int merge(vint &arr, vint &temp, int left, int mid,
          int right)
{
    int i, j, k;
    int inv_count = 0;

    i = left;
    j = mid;
    k = left;
    while ((i <= mid - 1) && (j <= right)) {
        if (arr[i] <= arr[j]) {
            temp[k++] = arr[i++];
        }
        else {
            temp[k++] = arr[j++];

            inv_count = inv_count + (mid - i);
        }
    }

    while (i <= mid - 1)
        temp[k++] = arr[i++];

    while (j <= right)
        temp[k++] = arr[j++];

    for (i = left; i <= right; i++)
        arr[i] = temp[i];

    return inv_count;
}
```

Graph of Analysis:



Size 686 pt Width 2027 pt Height

Constrain proportions

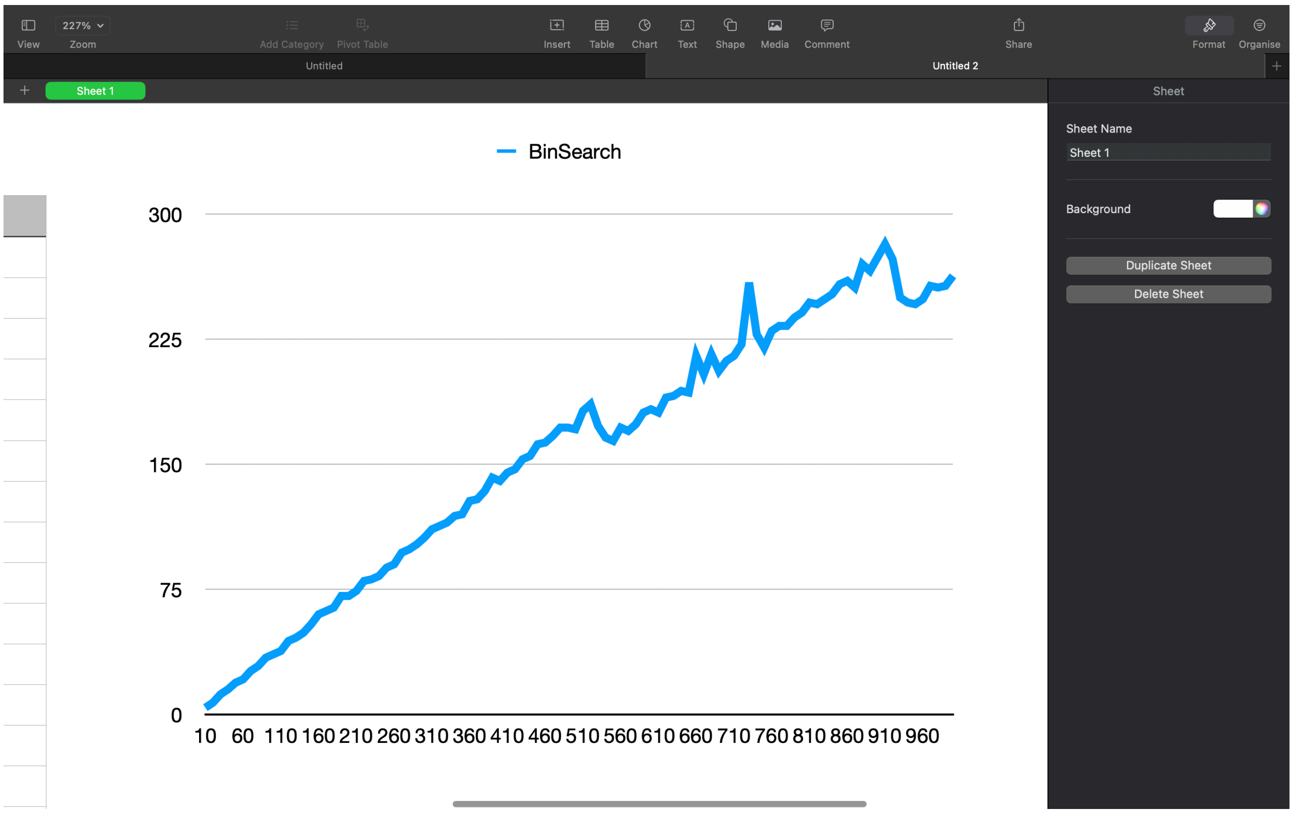
Position 0 pt X 28 pt Y

Q2.In the Second Question, I have used Binary Search to solve this question, we will binary search directly on the answer between 1 to 1e9,
Check function takes O(n) time and binary search is O(log(n)),
Hence final time complexity is O(nlog(n))
Space complexity: O(1)

Code:

```
bool check(vint &arr, int n, int m, int curr_min){
    int studentsRequired = 1;
    int curr_sum = 0;


    for (int i = 0; i < n; i++) {

        if (arr[i] > curr_min)
            return false;
        if (curr_sum + arr[i] > curr_min) {
            studentsRequired++;
            curr_sum = arr[i];

            if (studentsRequired > m)
                return false;
        }
        else
            curr_sum += arr[i];
    }
    return true;
}

int optimiseBooks(int n,int m,vint &arr){
    int lo=0,hi=1e9;
    while(lo<hi){
        int mid=lo+(hi-lo)/2;

        if(check(arr,n,m,mid)){
            hi=mid-1;
        }
        else{
            lo=mid+1;
        }
    }
    return hi;
}
```

Graph Analysis:



View
227%
Zoom
Add Category
Pivot Table
Insert
Table
Chart
Text
Shape
Media
Comment
Share
Format
Organise
Untitled
Untitled 2
Sheet 1
Sheet
Sheet Name
Sheet 1
Background
Duplicate Sheet
Delete Sheet
BinSearch
300
225
150
75
0
10 60 110 160 210 260 310 360 410 460 510 560 610 660 710 760 810 860 910 960