DAA LAB

Shubham Singh
1906587

LAB-6

H.E.-4.2:- WAP to sort a given set of elements using the Quick sort method and determine the time required to sort the elements. Repeat the experiment for different value of n. Plot a graph time taken v/s n. The elements can be generated randomly.

Program:-
```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
void swap(int *a, int *b) {
      int t= *a;
      *a= *b;
      *b= t;
}
int partition (int arr[], int low, int high) {
    int pivot= arr[high];
    int i= (low -1);

    for(int j= low; j<=high -1; j++) {
        if(arr[j] < pivot) {
            i++;
            swap(&arr[i], &arr[j]);
        }
    }
    swap(&arr[i+1], & arr[high]);
    return (i+1);
}
void quickSort (int arr[], int low , int high) {
    if(low < high) {
        int pi= partition(arr, low, high);
        quickSort (arr, low, pi-1);
        quickSort (arr, pi+1, high);
    }
}
```

1906587

```
int main() {
    int i,n;
    clock-t start,end;
    double total-cputime;
    printf("Enter size: ");
    scanf("%d", &n);
    int a[n];

    start= clock();

    for(i=0; i<n; i++) {
        a[i]=(rand()%50000);
        printf("%d\t", a[i]);
    }
    printf("\n Elements generated randomly...\n");
    quickSort(a, 0, n-1);
    printf("\n Array after quick sort: \n");
    for(i=0; i<n; i++)
        printf("%d\t", a[i]);

    end= clock();

    printf("\n\n CPU Time calculation!! ");
    total_cputime= ((double)(end-start));
    printf(" \n Total CPU time: %f ms", total_cputime);
    total_cputime=((double)(end-start)/CLOCKS_PER_SEC);
    printf("\n Total CPU time: %f s", total-cputime);

    return 0;
}
```

1906587

L.E.-5.2:- WAP to implementation of fractional knapsack algorithm.

Program:-
```
#include<stdio.h>
Struct knap{
   int p, w;
   float r, x;
};
void print(struct knap l[], int n) {
     int i,j;
     printf("\n");
     for (i=0; i<n; i++)
        printf("|%d %d %0.2f %0.2f|\n", l[i].p, l[i].w,
                                   l[i].r, l[i].x);
 }
float knapsack(int n, int m) {
   int i, j;
   float profit = 0.0;
   struct knap k[10], t;
   for (i=0; i<n; i++) {
      printf("Enter price and weight of item %d: ", i+1);
      scanf("%d %d", &k[i].p , &k[i].w);
      k[i].r = (float)(k[i].p) / (float)(k[i].w);
   }
  printf(k,n);
  printf("At first array look like\n");
  for(i=0; i<n-1; i++){
     for(j=i; j<n; j++) {
        if(k[i].r < k[j].r) {
              t = k[i];
              k[i] = k[j];
              k[j] = t;
        }
     }
  }
```

Scanned by TapScanner

1906587

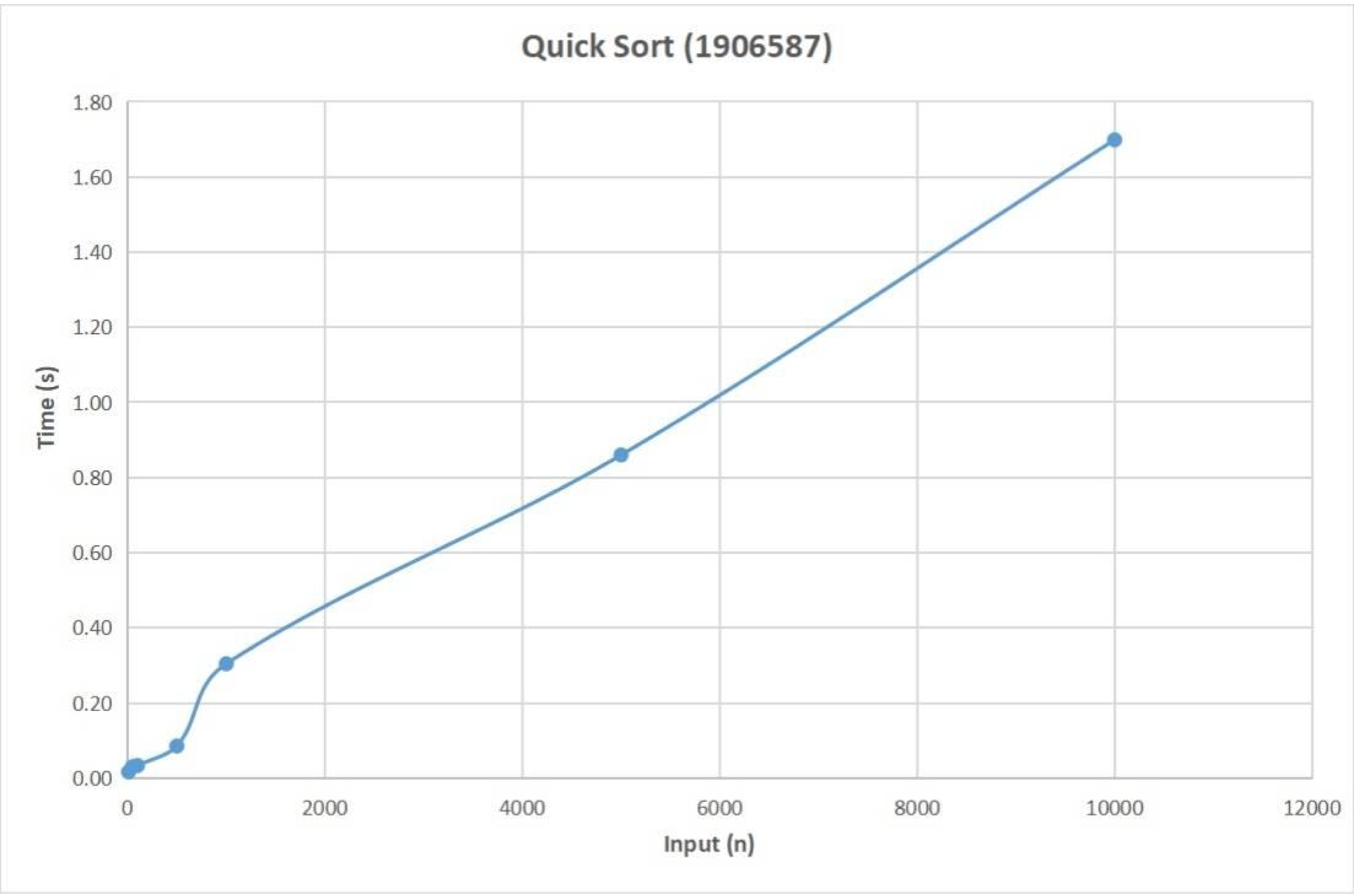
```
printf(k,n);
printf("After sorting the array w.r.t R \n");

for(i=0; i<n; i++){
   if(k[i].w<m){
       k[i].x=1;
       m-=k[i].w;
   }
   else{
     k[i].x= (float)(m) / (float)(k[i].w);
      m=0;
   }
   profit += (float)(k[i].p) * k[i].x;
 }
 printf(k,n);
 printf("finally the array looks like \n");

 return profit;
}

int main(){

  int n,m;
  printf("Enter Total Item: ");
  scanf("%d", &n);
  printf("Enter size of knapsack: ");
  scanf("%d", & m);
  printf("\n");
  printf("\n Maximum Profit will be: %.2f \n", ~~Knapsack(n,~~
                                              knapsack(n,m));
  return 0;
}
```


Quick Sort (1906587)
Time (s)
Input (n)
1.80
1.60
1.40
1.20
1.00
0.80
0.60
0.40
0.20
0.00
0
2000
4000
6000
8000
10000
12000