Group 5 – BCS32
Morales, Lincoln
Reyes, Jerizsa Hannah
Papa, Dranel Allen
Punzalan, Jeff Joecel
Vergara, Philip Nicole

A Proposal on E-commerce Platform with Auction and Barter Systems for Online Shoppers and Sellers

CHOSEN ALGORITHMS

1. Binary Search
2. Merge Sort
3. Dijkstra's Algorithm

INTRODUCTION

1.1 Project Context

When people buy the item/s they need, it is sometimes a hassle to travel to the shop and back. Why go in line when you can go online. Amazon changed the online shopping idea by becoming the new 'norm'. People nowadays are experiencing the luxury of items being delivered straight at the comfort of their home. Anything can now be seen and bought right away as soon as you check out the items. Payment methods have been improved to provide a safer way of paying for your items. An Online Shopping System gives the benefit of not going outside and being able to use your time for something else like working, spending time with your friends or family, and getting the items you need to buy.

1.2 Purpose and Description

The aim of this research project is to create an effective system for online shopping. With the help of this system, customers can carry out smooth and easy transactions without going to stores and shopping districts. To use the system the customer would need to enter their log-in credentials which includes the registered email, password, and the customer's shipping address. Logging in ensures the security of the customer's information as well as saving their previously checked in products. It helps the customer by personalizing their storefront based on the customer's information and order history. The purpose of the site is to facilitate an e-commerce alternative like the likes of Shopee and Lazada with a focus on a more minimalistic design principle. The name of the website is "Minima" which is short for the word Minimalistic in line with the theme of the project.

1.3 Scope and Limitations

The scope of the research covers the basics of a functioning online shopping system as well as prototype views for auctioning and bartering. It features a website with a working user interface (Front-End) and a database system (Back-end). Organization of two groups were made to reduce workload and allow the team to specialize in their respective field. Entries for purchasing and selling of items are fully functional and automatically added into the database.

To accomplish the goals of the research in a timely manner, some features that are included in mainstream online shopping systems have not yet been included. One such feature is an online messaging system to facilitate buyer and seller communication. This can be useful to address customer feedback and complaints. Another feature that was to be added was vendor profiles. This would allow the sellers to customize their own shop complete with contact details, shipping locations, and authorized sellers. Customers could also leave reviews for their overall experience with the shop. Lastly, the barter and auction systems which, for now, are still only in the prototype and proposal stage of development. These features are avenues for improvement when it comes to the development of the application even after the research.

ALGORITHMS

1.) Binary Search

A. Pseudocode

Do while low <= high

```
        middle=( low +high) / 2
        If target=array(middle) then
                bfound=true
                Exit do
        Elseif target< array(middle) then
                high= (middle-1)
        Else
                low=(midlle+1)
        End if
Loop
```

B. Algorithm Code w/ Sample Run (C++)

```
// BinarySearch
#include <stdio.h>

 int Binary(int A[100],int n, int x)
 {
    int low =0,high=n-1;
    while (low<=high)
    {
       int mid=(low+high)/2;
       if(x==A[mid])
         return mid;
       else if(x<A[mid])
         high=mid-1;
       else
         low=mid+1;

    }
    return -1;
 }
int main()
{
int n,c,A[100];
   printf("Enter size of array\n");
   scanf("%d",&n);

   printf("Enter %d integers\n", n);

   for (c = 0; c < n; c++)
      scanf("%d",&A[c]);

  printf("Enter a number:");
  int x; scanf("%d",&x);
  int index = Binary(A,8,x);
```

```
 if (index !=-1)
   printf ("Number %d is at index %d",x,index+1);
   else printf("Number %d not found",x);

   return 0;
}
```

```
Enter 5 integers
10 20 30 40 50
Enter a number:50
Number 50 is at index 5
```

C. Relevance of the algorithm in proposed system/application

Binary search is a searching algorithm that looks for a given element in a sorted array by repeatedly dividing the search interval in half until it is found. The purpose of Binary Search in the application can be found in database indexing. This algorithm allows for quick access of records by their keys in a database. A single product in a large database can be found if their keys are already sorted. This is true if the number of keys is large. 32 key reads would be enough to find any single unique key within a collection of two billion sorted keys. Binary search works this way because every search attempt cuts the number of records to search in half.

2.) Merge Sort

A. Pseudocode

MergeSort(arr[], l, r)

```
If r > l
        middle m = (l+r)/2
        Call mergeSort(arr, l, m)
        Call mergeSort(arr, m+1, r)
        Call merge(arr, l, m, r)
```

MERGE (A, p, q, r)

```
n1 = q – p +1
n2 = r – q
let L [1.. n1 + 1 ] and L [1.. n2 + 1 ]  be new arrays
for i=1 to  n1
      L[ i ]  =  A [ p + i -1]
      for j=1 to n2
            R[ j ]= A[ q + j ]
            L [n1 + 1 ] =  ∞
            R [n2 + 1 ] =  ∞
i = 1
j = 1
for k = p to r
      if L[ i ] < R [ j ]
```

```
                A[ k ] = L[ i ]
                i = i + 1
        else  A[ k ] = R [ j ]
                j = j + 1
```

B. Algorithm Code w/ Sample Run (Python)

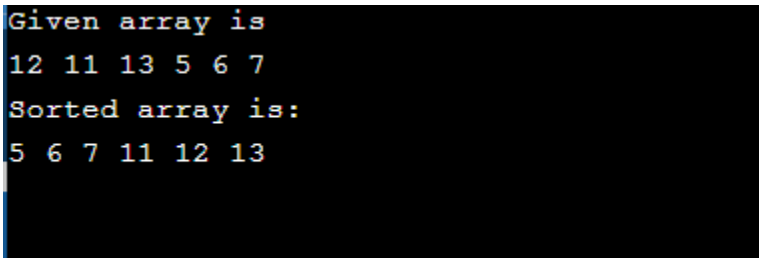
```
def mergeSort(arr):
    if len(arr) > 1:

        mid = len(arr)//2
        L = arr[:mid]
        R = arr[mid:]

        mergeSort(L)
        mergeSort(R)
        i = j = k = 0
        while i < len(L) and j < len(R):
            if L[i] < R[j]:
                arr[k] = L[i]
                i += 1
            else:
                arr[k] = R[j]
                j += 1
            k += 1

        while i < len(L):
            arr[k] = L[i]
            i += 1
            k += 1

        while j < len(R):
            arr[k] = R[j]
            j += 1
            k += 1
def printList(arr):
    for i in range(len(arr)):
        print(arr[i], end=" ")
    print()
if __name__ == '__main__':
    arr = [12, 11, 13, 5, 6, 7]
    print("Given array is", end="\n")
    printList(arr)
    mergeSort(arr)
    print("Sorted array is: ", end="\n")
    printList(arr)
```



C. Relevance of the algorithm in proposed system/application

Merge Sort is one of the most popular and used sorting algorithms that is based on the idea of Divide and Conquer Algorithm. Merge sort first divides the array into equal halves and then combines them in a sorted arrangement. The idea of keeping on dividing the list into equal halves until it can no more be divided is what made Merge sort the most used sorting algorithm. If it is only one element in the list, it is sorted. Then, merge sort combines the smaller sorted lists keeping the new list sorted too.

The purpose of Merge Sort in the proposed application is general product sorting. An example of this is arranging price ranges of products from highest to lowest and vice versa.

3.) Dijkstra's Algorithm

A. Pseudocode

```
function dijkstra(G, S)
   for each vertex V in G
      distance[V] <- infinite
      previous[V] <- NULL
      If V != S, add V to Priority Queue Q
   distance[S] <- 0

   while Q IS NOT EMPTY
      U <- Extract MIN from Q
      for each unvisited neighbour V of U
         tempDistance <- distance[U] + edge_weight(U, V)
         if tempDistance < distance[V]
            distance[V] <- tempDistance
            previous[V] <- U
   return distance[], previous[]
```

B. Algorithm Code w/ Sample Run

(C++)

```
#include<iostream>
#include<stdio.h>
using namespace std;
```

```
#define INFINITY 9999
#define max 5
void dijkstra(int G[max][max],int n,int startnode);
int main() {
  int G[max][max]={{0,1,0,3,10},{1,0,5,0,0},{0,5,0,2,1},{3,0,2,0,6},{10,0,1,6,0}};
  int n=5;
  int u=0;
  dijkstra(G,n,u);
  return 0;
}
void dijkstra(int G[max][max],int n,int startnode) {
  int cost[max][max],distance[max],pred[max];
  int visited[max],count,mindistance,nextnode,i,j;
  for(i=0;i<n;i++)
    for(j=0;j<n;j++)
  if(G[i][j]==0)
    cost[i][j]=INFINITY;
  else
    cost[i][j]=G[i][j];
  for(i=0;i<n;i++) {
    distance[i]=cost[startnode][i];
    pred[i]=startnode;
    visited[i]=0;
  }
  distance[startnode]=0;
  visited[startnode]=1;
  count=1;
  while(count<n-1) {
    mindistance=INFINITY;
    for(i=0;i<n;i++)
      if(distance[i]<mindistance&&!visited[i]) {
      mindistance=distance[i];
      nextnode=i;
    }
    visited[nextnode]=1;
    for(i=0;i<n;i++)
      if(!visited[i])
    if(mindistance+cost[nextnode][i]<distance[i]) {
      distance[i]=mindistance+cost[nextnode][i];
      pred[i]=nextnode;
    }
    count++;
  }
  for(i=0;i<n;i++)
  if(i!=startnode) {
    cout<<"\nDistance of node"<<i<<"="<<distance[i];
```

```
      cout<<"\nPath="<<i;
      j=i;
      do {
        j=pred[j];
        cout<<"<-"<<j;
      }while(j!=startnode);
    }
}
```

```
Distance of node1=1
Path=1<-0
Distance of node2=5
Path=2<-3<-0
Distance of node3=3
Path=3<-0
Distance of node4=6
Path=4<-2<-3<-0
```

C. Relevance of the algorithm in proposed system/application

Dijkstra's algorithm is finding the shortest path from a starting node to a target node in a weighted graph. The algorithm creates a tree of shortest paths from the starting vertex, the source, to all other points in the graph.

The purpose of using Dijkstra's Shortest Path Algorithm is to determine the fastest route to a specified destination. It results in having cheaper shipping fees as products do not need to travel to unnecessary locations to reach their intended recipient.

References

Binary Search. (2020, February 03). Retrieved January 29, 2021, from https://www.geeksforgeeks.org/binary-search/

Merge Sort. (2020, November 18). Retrieved January 29, 2021, from https://www.geeksforgeeks.org/merge-sort/

Dhanvani, P. (2013, November 19). Merge sort: Pseudo code of merge sort: Merge sort in data structure: Divide and conquer approach. Retrieved February 04, 2021, from https://freefeast.info/general-it-articles/merge-sort-pseudo-code-of-merge-sort-merge-sort-in-data-structure-divide-and-conquer-approach/

Dijsktra's algorithm. (2020, September 30). Retrieved January 29, 2021, from https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/

Dijkstra's Algorithm. (n.d.). Retrieved from https://www.programiz.com/dsa/dijkstra-algorithm

(n.d.). Retrieved from https://www.tutorialspoint.com/cplusplus-program-for-dijkstra-s-shortest-path-algorithm

Computer Science. (2016, September 30). Binary Search - Algorithm and Pseudo-code [Video]. YouTube. https://www.youtube.com/watch?app=desktop&v=2BhQxgIgXX4