14. אלגוריתמים מרכזיים

14.1 האלגוריתם המרכזי

האלגוריתם עוסק בבעיית מציאת הדרך הקצרה עבור מסלול אותו יצטרך לעבור המוביל על מנת לאסוף את המוצרים שקנה המשתמש.

על מנת לאסוף את המוצרים המשתמש יכול לבחור את היעדים באופן אקראי ולאוספם ללא סדר מסוים. אך בצורה זו יכול להיווצר מצב של עבודה רבה לחינם. משום שהמשתמש יכול לקצר את דרכו בהרבה ולסיים את המסע בפחות זמן, מאמץ וכסף. ובמיוחד אם מוביל מבחוץ יעשה את העבודה, כסף רב עלול להיזרק לפח.

לכן באפליקציה ניתנת למשתמש הצעה של מסלול לאיסוף המוצרים בפחות זמן, מאמץ וכסף.

חישוב המסלול נעשה באמצעות אלגוריתם דייקסטרה חמדני.

האלגוריתם עובד בצורה כזו:

ראשית מוצא את נקודת המוצא. נקודה זו היא הנקודה הרחוקה ביותר מהקונה. בצורה זו הסבירות גבוהה יותר שהתוצאות יצאו טובות יותר.

בשביל למצוא את נקודת המוצא, האלגוריתם מוצא את מיקומו של המשתמש ולאחר מכן מוצא עבור כל נקודה במסלול (-מוצר) את המרחק שלה ממנה למיקום המשתמש.

לאחר שמצא את כל המרחקים של כל המוצרים, שולף את המוצר בעל המרחק הגדול ביותר שהתקבל. מוצר זה יהיה נקודת המוצא.

לאחר שמצא את נקודת המוצא, מחפש את הנקודה הבאה במסלול. הנקודה הבאה היא הנקודה הקרובה ביותר לנקודת המוצא.

הפונקציה מחשבת את מיקומה של נקודת המוצא ומחשבת את המרחקים של כל המוצרים (למעט המוצר שהוא נקודת המוצא).

המוצר בעל המרחק הקצר ביותר לנקודת המוצא, הוא הנקודה הבאה במסלול, והוא נשמר ברשימת המסלולים כיעד הבא.

לאחר מכן ממשיך לחפש את הנקודות הבאות כך שהנקודה הבאה במסלול היא הקרובה ביותר לנקודה הקודמת שנמצאה, בצורה של חישוב המרחקים של כל הנקודות שנשארו מהנקודה הקודמת, עד לסיום הנקודות במסלול.

בכל שלב בו נמצאה הנקודה הבאה, נשמר המרחק בקילומטרים של הנקודה מהנקודה הקודמת, כך שבסוף התהליך מתקבל מספר הקילומטרים של המסלול מנקודת המוצא עד לבית הלקוח.

וכן נשמרים מספר הנקודות במסלול.

(מספר הקילומטרים ומספר הנקודות מחושב על מנת לאפשר חישוב מדויק יותר של מחיר הובלת המוצרים לפי המסלול שמתקבל)

מציאת המרחקים נעשה בעזרת הפונקציה GetDistance שמתממשקת ל google Maps.

הפונקציה ממירה את הכתובות לקורדינציות ומעבירה אותם כפרמטר ל-Api של Google Maps הדורש key שרכשתי Google Cloud Platform (https://console.cloud.google.com/)

לפעולה זו של ההתממשקות, נשלחות 2 כתובות כל פעם: כתובת המקור וכתובת היעד.

הכתובת מורכבת מהעיר והכתובת המדויקת אותם הכניס המשתמש בעת שנרשם לאפליקציה.

בסיום הריצה הפונקציה מחזירה רשימה של המוצרים מסודרים לפי המרחקים שלהם כפי שתואר לעייל לצד לקוח שם מפרקים אותו ושותלים אותו בתוך התצוגה הגרפית.

15. קוד התוכנית

הגדרה של הנתונים ואיתחולם:

```
class Dijkstra
{
    int numOfKiloMeters;
    int numOfProduct;

    User u = new User();
    City c;
    MapsService alg;
    SortedList<string, Products> dic;
    Product p;
    ShoppingCastBL sc;
    List<Products> path;

    3 references
    public Dijkstra()
    {
        numOfKiloMeters = 0;
        numOfProduct = 1;
    }
```

יצירת מסלול- הפונקציה MainDijkstraPath():

```
public List<Products> MainDijkstraPath(int codeUser)
{
    var destination = u.GetUserById(codeUser);
    var pro = sc.GetProductInShoppingCast(codeUser);
    numOfProduct = pro.Count();
    var FirstNode = GetFirstNode(destination, pro);
    path.Add(FirstNode);
    pro.Remove(FirstNode);
    while (pro != null)
    {
        var (next, num) = NextDestination(int.Parse(path.Last().CodeSallerProduct.ToString()), pro);
        path.Add(next);
        pro.Remove(next);
        numOfKiloMeters += num;
    }
    numOfKiloMeters += GetLastNode(codeUser, path.Last().CodeProduct);
    return path;
}
```

מציאת נקודת המוצא- הפונקציה GetFirstNode():

```
public Products GetFirstNode(Users destination, List<Products> product)
{
    string FullAdressUser = c.getNameCityById(int.Parse(destination.CityUser.ToString()))
        + ' ' + destination.AddresstUser;
    foreach (var p in product)
    {
        var saller = u.GetUserById(int.Parse(p.CodeSallerProduct.ToString()));
        string FullAdressProduct = c.getNameCityById(int.Parse(saller.CityUser.ToString()))
            + ' ' + saller.AddresstUser;
        dic[alg.GetDistance(FullAdressUser, FullAdressProduct).ToString()] = p;
    }

    return dic.Last().Value;
}
```

מציאת הנקודה הבאה- הפונקציה NextDestination() :

```
public (Products, int) NextDestination(int codeUser, List<Products> product)
{
    Users user = u.GetUserById(codeUser);
    SortedList<string, Products> dic = new SortedList<string, Products>();
    string FullAdressOrigin = c.getNameCityById(int.Parse(user.CityUser.ToString()))
        + ' ' + user.AddresstUser;
    foreach (var p in product)
    {
        var saller = u.GetUserById(int.Parse(p.CodeSallerProduct.ToString()));
        string FullAdressProduct = c.getNameCityById(int.Parse(saller.CityUser.ToString()))
            + ' ' + saller.AddresstUser;
        dic[alg.GetDistance(FullAdressOrigin, FullAdressProduct).ToString()] = p;
    }

    return (dic.First().Value, int.Parse(dic.First().Key));
}
```

מציאת המרחק של הנקודה האחרונה- הפונקציה GetLastNode() :

```
public int GetLastNode(int codeUser, int codeProduct)
{

    Users user = u.GetUserById(codeUser);
    string FullAdressUser = c.getNameCityById(int.Parse(user.CityUser.ToString()))
        + ' ' + user.AddresstUser;
    var saller = u.GetUserById(int.Parse(p.GetProductById(codeProduct).CodeSallerProduct.ToString()));
    string FullAdressProduct = c.getNameCityById(int.Parse(saller.CityUser.ToString()))
        + ' ' + saller.AddresstUser;

    return int.Parse(alg.GetDistance(FullAdressUser, FullAdressProduct).ToString());
}
```

מציאת מרחק בין 2 נקודות- הפונקציה GetDistance() :

```
public async Task<string> GetDistance(string origin, string destination)
{
    string[] locationUrls = { BuildUrlForLocationId(origin), BuildUrlForLocationId(destination) },
        idLocations = new string[2];
    HttpClient http = new HttpClient();

    for (int i = 0; i < idLocations.Length; i++)
    {
        var responseId = await http.GetAsync(locationUrls[i]);

        if (responseId.IsSuccessStatusCode)
        {
            var result = await responseId.Content.ReadAsStringAsync();

            RootLocationBase root = JsonConvert.DeserializeObject<RootLocationBase>(result);
            idLocations[i] = root.results[0].place_id;
        }
    }


    var responseDistance = await http.GetAsync(BuildUrlForDistance(idLocations[0], idLocations[1]));

    if (responseDistance.IsSuccessStatusCode)
    {
        var result = await responseDistance.Content.ReadAsStringAsync();
        DistanceBase root = JsonConvert.DeserializeObject<DistanceBase>(result);
        // Console.WriteLine("Distance: " + root.rows[0].elements[0].distance.text + ".");
        // Console.WriteLine("Duration: " + root.rows[0].elements[0].duration.text + ".\n");
        return root.rows[0].elements[0].distance.text;
    }
    return "";
}
```

הפונקציה BuildUrlForLocationId() :

```
static string BuildUrlForLocationId(string address)
{
    string location = "";
    string locationAsArray;
    locationAsArray = address;

    for (int i = 0; i < locationAsArray.Length; i++)
    {
        if (i < locationAsArray.Length - 1)
            location += locationAsArray[i] + "+";
        else
            location += locationAsArray[i];
    }

    return "https://maps.googleapis.com/maps/api/place/textsearch/json?key=AIzaSyAGX5jTROUV3WlWI0zLS7V_kQi2z7PALho&query="
        + location;
}
```

הפונקציה BuildUrlForDistance() :

```
static string BuildUrlForDistance(string place1, string place2)
{
    string url = "https://maps.googleapis.com/maps/api/distancematrix/json?key=AIzaSyAGX5jTROUV3WlWI0zLS7V_kQi2z7PALho&units=imperial&origins=";
    return url + "place_id:" + place1 + "&destinations=place_id:" + place2;
}
```

עבור חישוב ההובלה- הפונקציה () DijkstraKilometters :

```
public (int, int) DijkstraKilometters()
{
    return (numOfKiloMeters, numOfProduct);
}
```

חישוב ההובלה- הפונקציה ()GetCostByDijkstraShoppingCast:

```
public int GetCostByDijkstraShoppingCast(int codeUser)
{
    Dijkstra.Dijkstra dijkstra = new Dijkstra.Dijkstra();
    var (numOfKiloMeters, numOfProduct) = dijkstra.DijkstraKilometters();
    return numOfKiloMeters * 20 + numOfProduct*20;
}
```

חישוב הובלת מוצר אחד- ()GetCostByDijkstraToProduct:

```
public int GetCostByDijkstraToProduct(int codeUser, int codeProduct)
{
    Dijkstra.Dijkstra dijkstra = new Dijkstra.Dijkstra();
    int numOfKiloMeters = dijkstra.GetLastNode(codeUser, codeProduct);

    return numOfKiloMeters * 20;
}
```