PROJE 3:BAĞLI LİSTE UYGULAMASI

Erdem Nayın 180202050 – Melih Yeşilyurt 180202060

Bilgisayar Mühendisliği Bölümü

Kocaeli Üniversitesi

erdem_nayin@hotmail.com – Melih58yesilyurt@outlook.com

1-)Özet

Kocaeli'den başlayarak kullanıcının seçtiği şehirleri dolaşarak en kısa mesafeden Kocaeli'ne geri dönen program geliştirildi. Uygulama açıldığında ilk önce kullanıcıya Kocaeli haricinde kaç şehire teslimat yapmak istediği soruluyor. Kullanıcı maksimum 10 adet şehir girebilir. Daha fazla şehir gezmek isterse program hata mesajı belirterek kriterlere uygun sayıda şehir sayısı almak için kullanıcıdan tekrar şehir sayısı girmesini istiyor. Kullanıcı, teslimat yapılacak şehir sayısını girdikten sonra gelen ekranda Türkiye Siyasi Haritası üzerinden uğramak istediği şehirleri seçiyor. Eğer kullanıcı belirttiği şehir sayısından fazla şehir girerse seçtiği şehirlerin plaka numarasına göre küçükten büyüğe şekilde belirttiği şehir sayısı kadar şehir seçiliyor. Daha sonra program algoritmaları kullanarak tüm şehirlerin dolaşılıp Kocaeli'ne geri dönüldüğü en kısa yolu buluyor. Daha sonra ekrandaki Türkiye Siyasi Haritası üzerinde bulunan yolu çiziliyor ve kullanıcıya gösteriliyor.

2-)Giriş

Bir kullanıcının Kocaeli'den başlayıp maksimum 10 şehrin dolaşıldığı ve tekrar Kocaeline en kısa yoldan dönüldüğü bir uygulama yapılacaktır.

İlk önce kullanıcı gezmek istediği şehir sayısını giriyor. Program kaç şehrin gezileceğinin sayısını cityCount değişkeninde tutuyor. Daha sonra kullanıcı ekrandaki haritadan gezmek istediği şehirleri seçiyor, program bu şehirlerin ve Kocaeli'nin plakalarını `int[] travellingCities` dizisinde tutuyor. Daha sonra program bu şehirleri başta Kocaeli olacak şekilde City classı türünden temsil edildiği bir `ArrayList<City> cities` ArrayListi oluşturuyor. City classı bir şehri temsil ediyor ve içinde plaka numarasını tutan `private int plateNo`, diğer şehirlere olan en kısa mesafelerini tutan `ArrayList<Integer> distancesToOtherCities`, bu en kısa mesafelerin pathleri plaka numaralarına göre tutan iki boyutlu `ArrayList<ArrayList<Integer>> paths`, plateNo için getter ve setter methodlarıyla birlikte City classının constructor methodu bulunuyor.

Daha sonra elimizde bulunan cities ArrayListi, travellingCities dizisi ve programın içinde gömülü bulunan Türkiye'deki tüm şehirler ve komşu olan şehirlerin mesafelerinin bulunduğu graph yapısı dijkstra algoritmasına gönderiliyor. Böylece Kocaeli ve gezilecek diğer şehirlerin, Türkiye'deki tüm şehirlere olan en kısa mesafeli yolları ve pathleri hesaplanıyor ve City tipinde tutulan şehirlerimizin içinde bulunan ve yukarıda açıkladığımız distancesToOtherCities ve paths ArrayListleri, bu bilgilerle dolduruluyor.

Daha sonra Kocaeli ile kullanıcının girdiği şehirler ve bu şehirlerin birbirlerine olan uzaklıklarının bulunduğu `int[][] arrayForTSP` graph yapısı oluşturuluyor. Oluşturulan bu graph, kullanıcının girdiği tüm şehirlerin Kocaeli'den başlanarak gezildiği ve Kocaeli'ne tekrar ulaşımının çözümünü verecek olan minimum spanning tree çözümü olan TSP fonksiyonuna gönderiliyor. Bu fonksiyonda, kullanıcının verdiği şehirlerin ve Kocaeli'nin komşulara uğramadan gezildiği en kısa mesafeli yol ve yolun mesafesi bulunuyor. Daha sonra bu yolların arasına komşu şehirler de eklenerek `ArrayList<Integer> pathForMapDrawing` arrayListi oluşturuluyor ve sonunda Kocaeli'den başlayıp kullanıcının seçtiği şehirleri gezen ve Kocaeli'ne bu şehirlerin komşuları üzerinden bağlantı kurarak ulaşan en kısa mesafeli path bulunuyor ve ekranda gösteriliyor.

3-)Yöntem

Proje Java programlama dili, IntelliJ ve Scenebuilder geliştirme ortamları kullanılarak yapıldı.

Java dilinde bulunan hazır kütüphanelerden ve fonksiyonlardan yararlanıldı. Bulunmayan fonksiyonlar ekibimiz tarafından yapıldı.

Proje yapılırken birçok problemle karşılaşıldı. Problemlerin çözümünde ekip halinde çözüm aranıldı, aynı problem veya benzeri bir problemle karşılaşmış arkadaşlarımızla görüş alışverişi yapıldı, internette daha önce bu problemlerle karşılaşan insanların bu problemlerin üstesinden nasıl geldiği araştırıldı, stackoverflow'da bu sorunlar hakkında developerların görüşü alındı ve farklı bakış açıları kazanıldı. Bulunan birçok farklı çözümden problemlerimizin çözümlerine uygun olanları seçildi.

Proje geliştirilirken ilk olarak ,tüm şehirleri teker teker dolaşıp kullanıcının istediği şehirler gezildikçe onları ve şu ana kadar gezilen yolu tutacak ve tüm şehirler gezilince oluşan pathimiz daha önce aynı yolla bulunan pathlerden kısaysa onu tutacak şeklinde bir greedy yaklaşım sergilemeyi düşündük. Fakat böyle yaparsak çözüm çok uzun sürecek, belki loopa girecek ve eğer kontrolünü iyi yapamazsak dakikalarca arasa bile doğru çözüme ulaşılamayacaktı. O yüzden yeni bir yaklaşım aramaya koyulduk. Stackoverflow, geeksforgeeks gibi siteler ve forumlardan bu sorunun çözümüne yararlı olabilecek çözüm yolları bulmaya çalıştık ve bulamadık. Eğer şehirler arasındaki kuş uçuşu mesefe,

dakika cinsinden ortalama uzaklıklar gibi bilgiler olsaydı yapay zeka kodlamalarında kullanılan heuristic algoritmalardan yararlanarak olası optimum çözümlere ulaşabilirdik fakat elimizde sadece şehirler arası en kısa mesafeler bulunduğundan brute force algoritmalardan başka çözüm yolumuz yoktu. Bunun sonucunda brute force kullanarak kendi yaklaşımımızı oluşturmamız gerektiği konusunda hemfikir olduk ve Kesikli Matematik dersinde işlediğimiz graf ve ağaç yapıları üzerinde gezinme konularının üzeinden tekrar geçtik ve sonunda probleme bir yaklaşımda bulunduk.

İlk önce dijkstra algoritması kullanarak Kocaeli ve kullanıcının seçtiği şehirlerin, Türkiye'de bulunan tüm şehirlere olan en kısa mesafelerini ve pathlerini bir grafın matris temsili üzerinden hesapladık. Kullandığımız matris 81x81 olup tüm şehirlerin, kendi komşularına olan uzaklıklarını içeriyordu. Örneğin 0.satır Ankara olarak temsil edilsin. 0. satır ve 1.sütun Ankara'nın Adıyaman'a olan uzaklığını, eğer Adıyaman Ankara'ya komşuysa ilgili uzaklık değeri, değilse sıfır olarak temsil edilmekte. Kocaeli ve kullanıcının seçtiği şehirler için bulduğumuz en kısa mesafeleri ve pathleri City referans tipinde bir ArrayListte depoladık. Böylece elimizde gezilecek şehirlerin kendi aralarındaki ve uğrayacakları muhtemel diğer şehirler için en kısa mesafeleri ve pathlerini bulmuş olduk.

Daha sonra tüm nodeları birbirine komşu olan bir grafın, verilen herhangi bir nodedan başlayarak diğer tüm nodeları en kısa mesafeyle dolaşmasını sağlayan minimum spanning tree algoritması kullandık. Bu algoritmayı kullanmak için Kocaeli ve kullanıcının seçtiği şehirlerin birbirlerine göre uzaklıklarını içeren grafın matris temsilini daha önceden hesapladığımız ve tuttuğumuz şehirlerin birbirine olan en kısa mesaflerini çekerek matrisi oluşturduk. Sonunda elimizde Kocaeli ve kullanıcının seçtiği şehirleri gezen en kısa mesafeyi ve pathini bulmuş olduk fakat şu an elimizde sadece ana şehirler arasında bulunmuş olan en kısa mesafe pathi olmuş oldu. Yani henüz gezilecek iki ana şehir arasındaki path yok. Bu pathleri de dijkstra algoritmasında daha önceden hesapladığımız ve City tipindeki şehirlerimizin içinde tuttuğumuz için genel olarak en kısa yol pathini aralardaki pathleri de doldurarak bulmuş olduk.

Kullandığımız dijkstra algoritması ve TSP algoritmalarını geeksforgeeks'ten aldık ve kaynakta ilgili kodları belirttik çünkü bu tür algoritmaları yazabilecek kabiliyete henüz sahip değildik.

Kod Konsol üzerinden çalışabilir duruma geldikten sonra geriye kalan tek şey arayüz oldu. Scenebuilder kullanılarak arayüz tasarlandı. 81 tane checkbox, 2 button, 2 label, 1 textfield, 1 tane polyline kullanıldı. 2 labelda kullanıcıya verilmek üzere mesaj yazıldı. Textfield sayesinde seçilmek istenen şehir sayısı kullanıcıdan alındı. Gönder adlı buton sayesinde algoritmaya bu sayı gönderildi. Kullanıcı seçmek istediği şehirleri checkboxlara tıklayarak seçebildi. Kocaelini kullanıcının seçmesine izin

verilmedi çünkü yola ilk oradan çıkılıp en son oraya dönüleceği için oradaki checkbox görünmez hale getirildi. Daha sonra start butonu sayesinde seçilen şehirler algoritmaya gönderildi ve Yazdığımız algoritmalar sayesinde en kısa yol bulundu. Polyline kullanılarak haritada güzergah gösterildi. Güzergahta ki her şehirin konumunu bulmak için checkboxların konumları kullanıldı. Bu konumlar Polyline'ın içine aktarıldı ve ekrana yansıtıldı.

4-)Sonuç

Sonuçta Java dili kullanılarak travelling salesman, minimum spanning tree algoritmaları, dijkstra algoritması, JavaFX ile arayüz tasarımı gibi konularda program yazılarak nesneye yönelik programlama düşünce tarzının ve algoritmik düşünme yetisinin gelişmesi sağlandı.

5-)Deneysel Sonuçlar

Seçilen Şehirler Uşak, Sivas, Edirne, Bilecik, İzmir, Konya, Mersin, Hatay, Hakkari, Kars. Dijkstra seçilen şehirlerin tüm şehirlere olan uzaklıklarını ve pathlerini buldu.

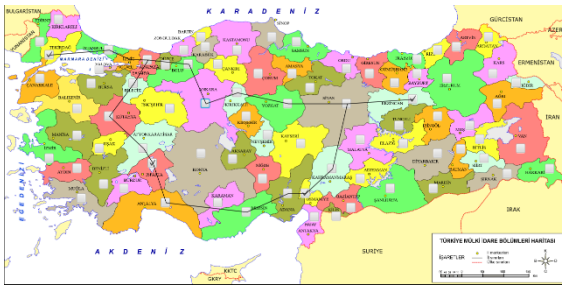
Grafik Sonuç:

Konsol Sonuç:

```
0  136  382  1746  1087  889  459  1422  557  781  387
136  0  518  1720  1029  769  421  1396  421  755  251
382  518  0  2128  1469  1271  590  1804  939  1163  750
1746  1720  2128  0  904  992  1908  639  1340  967  1678
1087  1029  1469  904  0  260  1176  1027  608  561  946
889  769  1271  992  260  0  916  1107  348  522  686
459  421  590  1908  1176  916  0  1707  568  1066  230
1422  1396  1804  639  1027  1107  1707  0  1139  641  1477
557  421  939  1340  608  348  568  1139  0  498  338
781  755  1163  967  561  522  1066  641  498  0  836
387  251  750  1678  946  686  230  1477  338  836  0
Minimum cost : 5223
Path Taken : 0 1 9 7 3 4 5 8 10 6 2 0
Path taken : 40 10 57 35 29 30 32 41 63 34 21 40
[40, 53, 10]
[10, 25, 5, 70, 65, 57]
[57, 23, 24, 35]
[35, 3, 64, 29]
[29, 72, 46, 62, 26, 30]
[30, 0, 32]
[32, 41]
[41, 2, 63]
[63, 44, 34]
[34, 9, 16, 21]
[21, 58, 33, 40]
41 54 11 26 6 71 66 58 24 25 36 4 65 30 73 47 63 27 31 1 33 42 3 64 45 35 10 17 22 59 34 41
```

Seçilen Şehirler Tekirdağ, Erzincan, Isparta, Osmaniye. Dijkstra seçilen şehirlerin tüm şehirlere olan uzaklıklarını ve pathlerini buldu.

Grafik Sonuç:



Konsol Sonuç:

```
0  1029  516  242  983
1029  0  1009  1271  680
516  1009  0  758  752
242  1271  758  0  1225
983  680  752  1225  0
Minimum cost : 3461
Path Taken : 0 1 4 2 3 0
Path taken : 40 23 79 31 58 40
[40, 53, 13, 5, 70, 65, 57, 23]
[23, 57, 45, 79]
[79, 0, 32, 6, 31]
[31, 2, 42, 10, 53, 40, 33, 58]
[58, 33, 40]
41 54 14 6 71 66 58 24 58 46 80 1 33 7 32 3 43 11 54 41 34 59 34 41
```

6-)Yalancı Kod

- Başla
- Uygulamayı göster
- Seçmek istediğin şehir Sayısını gir
- Gönder Butonuna bas
- Seçilmek istenen şehir sayısı kadar, seçmek istenilen şehirlerin

bulunduğu yerlerdeki checkboxları işaretle
- Başlat butonuna bas
- Yolu konsolda ve ekranda göster

7-)Kaynakça

https://www.youtube.com/watch?v=eB61LXLZVqs

https://www.geeksforgeeks.org/java-program-for-dijkstras-shortest-path-algorithm-greedy-algo-7/

kodda kullandığımız => https://www.geeksforgeeks.org/printing-paths-dijkstras-shortest-path-algorithm/?ref=rp

Kodda kullandığıımız=>https://www.geeksforgeeks.org/traveling-salesman-problem-using-branch-and-bound-2/

https://www.geeksforgeeks.org/traveling-salesman-problem-tsp-implementation/

https://www.geeksforgeeks.org/maximum-possible-edge-disjoint-spanning-tree-from-a-complete-graph/?ref=rp

stackoverflow'da sorduğumuz soru:https://stackoverflow.com/questions/60873945/algorithm-shortest-travelling-route-between-10-cities

https://www.geeksforgeeks.org/hamiltonian-cycle-backtracking-6/?ref=rp