PREPERATION

```
graph = {
    'ADANA': ['HATAY','OSMANİYE','KAHRAMANMARAŞ','KAYSERİ','NİĞDE','MERSİN'],
    'ADIYAMAN': ['ŞANLIURFA','DİYARBAKIR','MALATYA','KAHRAMANMARAŞ','GAZİANTEP'],
    'AFYONKARAHİSAR': ['ISPARTA','KONYA','ESKİŞEHİR','KÜTAHYA','UŞAK','DENİZLİ','
    'AĞRI': ['VAN','IĞDIR','KARS','ERZURUM','MUŞ','BİTLİS'],
    'AMASYA': ['YOZGAT','TOKAT','SAMSUN','ÇORUM'],
    'ANKARA': ['KONYA','AKSARAY','KIRŞEHİR','KIRIKKALE','ÇANKIRI','BOLU','ESKİŞEH
    'ANTALYA':['MERSİN','KARAMAN','KONYA','ISPARTA','BURDUR','MUĞLA'],
    'ARTVİN':['RİZE','ERZURUM','ARDAHAN'],
    'AYDIN':['MUĞLA','DENİZLİ','MANİSA','İZMİR'],
    'BALIKESİR':['İZMİR','MANİSA','KÜTAHYA','BURSA','ÇANAKKALE'],
    'BİLECİK':['KÜTAHYA','ESKİŞEHİR','BOLU','SAKARYA (ADAPAZARI)','BURSA'],
    'BİNGÖL':['DİYARBAKIR','MUŞ','ERZURUM','ERZİNCAN','TUNCELİ','ELAZIĞ'],
    'BİTLİS':['SİİRT','VAN','AĞRI','MUŞ','BATMAN'],
    'BOLU':['ESKİŞEHİR','ANKARA','ÇANKIRI','ZONGULDAK','DÜZCE','SAKARYA (ADAPAZAR
    'BURDUR':['MUĞLA','ANTALYA','ISPARTA','AFYONKARAHİSAR','DENİZLİ'],
    'BURSA':['BALIKESİR','KÜTAHYA','BİLECİK','SAKARYA (ADAPAZARI)','KOCAELİ (İZMİ
    'ÇANAKKALE':['BALIKESİR','TEKİRDAĞ','EDİRNE'],
    'ÇANKIRI':['ANKARA','KIRIKKALE','ÇORUM','KASTAMONU','ZONGULDAK','BOLU','KARAE
    'ÇORUM':['YOZGAT','AMASYA','SAMSUN','SİNOP','KASTAMONU','ÇANKIRI','KIRIKKALE'
    'DENİZLİ':['MUĞLA','BURDUR','AFYONKARAHİSAR','UŞAK','MANİSA','AYDIN'],
    'DİYARBAKIR':['ŞANLIURFA','MARDİN','BATMAN','MUŞ','BİNGÖL','ELAZIĞ','MALATYA'
    'EDİRNE':['ÇANAKKALE','TEKİRDAĞ','KIRIKKALE'],
    'ELAZIĞ':['DİYARBAKIR','BİNGÖL','TUNCELİ','ERZİNCAN','MALATYA'],
    'ERZİNCAN':['ELAZIĞ','TUNCELİ','BİNGÖL','ERZURUM','BAYBURT','GÜMÜŞHANE','GİRE
    'ERZURUM':['BİNGÖL','MUŞ','AĞRI','KARS','ARDAHAN','ARTVİN','RİZE','TRABZON','
    'ESKİŞEHİR':['AFYONKARAHİSAR','KONYA','ANKARA','BOLU','BİLECİK','KÜTAHYA'],
    'GAZİANTEP':['KİLİS','ŞANLIURFA','ADIYAMAN','KAHRAMANMARAŞ','OSMANİYE','HATAY
    'GİRESUN':['GÜMÜŞHANE','TRABZON','ERZİNCAN','SİVAS','ORDU'],
    'GÜMÜŞHANE':['ERZİNCAN','BAYBURT','TRABZON','GİRESUN'],
    'HAKKARİ':['VAN','ŞIRNAK'],
    'HATAY':['KİLİS','GAZİANTEP','OSMANİYE','ADANA'],
    'ISPARTA':['ANTALYA','KONYA','AFYONKARAHİSAR','BURDUR'],
    'MERSİN':['ADANA','NİĞDE','KONYA','KARAMAN','ANTALYA'],
    'İSTANBUL':['KOCAELİ (İZMİT)','TEKİRDAĞ','KIRKLARELİ'],
    'İZMİR':['AYDIN','MANİSA','BALIKESİR'],
    'KARS':['AĞRI','IĞDIR','ARDAHAN','ERZURUM'],
    'KASTAMONU':['ÇORUM','SİNOP','ÇANKIRI','BARTIN','KARABÜK'],
    'KAYSERİ':['ADANA','KAHRAMANMARAŞ','SİVAS','YOZGAT','NEVŞEHİR','NİĞDE'],
    'KIRKLARELİ':['EDİRNE','TEKİRDAĞ','İSTANBUL'],
    'KIRŞEHİR':['NEVŞEHİR','YOZGAT','KIRIKKALE','ANKARA','AKSARAY'],
    'KOCAELİ (İZMİT)':['YALOVA','İSTANBUL','BURSA','BİLECİK','SAKARYA (ADAPAZARI)
    'KONYA':['ANTALYA','KARAMAN','MERSİN','NİĞDE','AKSARAY','ANKARA','ESKİŞEHİR',
    'KÜTAHYA':['MANİSA','UŞAK','AFYONKARAHİSAR','ESKİŞEHİR','BİLECİK','BURSA','BA
    'MALATYA':['KAHRAMANMARAŞ','ADIYAMAN','DİYARBAKIR','ELAZIĞ','ERZİNCAN','SİVAS
    'MANİSA':['İZMİR','AYDIN','DENİZLİ','UŞAK','KÜTAHYA','BALIKESİR'],
    'KAHRAMANMARAŞ':['GAZİANTEP','ADIYAMAN','MALATYA','SİVAS','KAYSERİ','ADANA','
    'MARDİN':['ŞANLIURFA','DİYARBAKIR','BATMAN','SİİRT','ŞIRNAK'],
    'MUĞLA':['ANTALYA','BURDUR','DENİZLİ','AYDIN'],
    'MUŞ':['DİYARBAKIR','BATMAN','BİTLİS','AĞRI','ERZURUM','BİNGÖL'],
    'NEVŞEHİR':['NİĞDE','KAYSERİ','YOZGAT','KIRŞEHİR','AKSARAY'],
```

```
    'NİĞDE':['NEVŞEHİR','KAYSERİ','ADANA','MERSİN','KONYA','AKSARAY'],
    'ORDU':['SAMSUN','TOKAT','SİVAS','GİRESUN'],
    'RİZE':['ARTVİN','ERZURUM','BAYBURT','TRABZON'],
    'SAKARYA (ADAPAZARI)':['DÜZCE','BOLU','BİLECİK','BURSA','KOCAELİ (İZMİT)'],
    'SAMSUN':['ORDU','TOKAT','AMASYA','ÇORUM','SİNOP'],
    'SİİRT':['VAN','BİTLİS','BATMAN','MARDİN','ŞIRNAK'],
    'SİNOP':['SAMSUN','ÇORUM','KASTAMONU'],
    'SİVAS':['KAYSERİ','KAHRAMANMARAŞ','MALATYA','ERZİNCAN','GİRESUN','ORDU','TOK
    'TEKİRDAĞ':['İSTANBUL','KIRKLARELİ','EDİRNE','ÇANAKKALE'],
    'TOKAT':['SİVAS','ORDU','SAMSUN','AMASYA','YOZGAT'],
    'TRABZON':['RİZE','BAYBURT','GÜMÜŞHANE','GİRESUN'],
    'TUNCELİ':['ELAZIĞ','BİNGÖL','ERZİNCAN'],
    'ŞANLIURFA':['GAZİANTEP','ADIYAMAN','DİYARBAKIR','MARDİN'],
    'UŞAK':['MANİSA','DENİZLİ','AFYONKARAHİSAR','KÜTAHYA'],
    'VAN':['HAKKARİ','ŞIRNAK','SİİRT','BİTLİS','AĞRI'],
    'YOZGAT':['KAYSERİ','SİVAS','TOKAT','AMASYA','ÇORUM','KIRIKKALE','KIRŞEHİR','
    'ZONGULDAK':['BARTIN','ÇANKIRI','BOLU','DÜZCE','KARABÜK'],
    'AKSARAY':['NİĞDE','NEVŞEHİR','KIRŞEHİR','ANKARA','KONYA'],
    'BAYBURT':['ERZİNCAN','ERZURUM','RİZE','TRABZON','GÜMÜŞHANE'],
    'KARAMAN':['MERSİN','KONYA','ANTALYA'],
    'KIRIKKALE':['KIRŞEHİR','YOZGAT','ÇORUM','ÇANKIRI','ANKARA'],
    'BATMAN':['MARDİN','SİİRT','BİTLİS','MUŞ','DİYARBAKIR'],
    'ŞIRNAK':['MARDİN','SİİRT','VAN','HAKKARİ'],
    'BARTIN':['KASTAMONU','ZONGULDAK','KARABÜK'],
    'ARDAHAN':['KARS','ERZURUM','ARTVİN'],
    'IĞDIR':['AĞRI','KARS'],
    'YALOVA':['KOCAELİ (İZMİT)','BURSA'],
    'KARABÜK':['ZONGULDAK','BARTIN','KASTAMONU','ÇANKIRI'],
    'KİLİS':['GAZİANTEP','HATAY'],
    'OSMANİYE':['GAZİANTEP','KAHRAMANMARAŞ','ADANA','HATAY'],
    'DÜZCE':['ZONGULDAK','BOLU','SAKARYA (ADAPAZARI)']

} # Nothing but cities with adjacent with each other.
```

```
import pandas as pd
df= pd.read_csv('/content/DataSet/iller_mesafe.csv')

def neighbourHerusticService(targetCity,solutionPath):
  solutionPath.remove(targetCity)
  selectedRow=df.loc[(df['İL ADI']==targetCity)]
  data = selectedRow[selectedRow.columns.intersection(solutionPath)]
  return data #Will bring the distance to adjacent cities

def cityHerusticService(targetCity,city):
  cityList=[city]
  selectedRow=df.loc[(df['İL ADI']==targetCity)]
  data = selectedRow[selectedRow.columns.intersection(list(cityList))]
  #print(data.iloc[0][city])
  return data.iloc[0][city] #Will bring the distance to adjacent cities

def neighbourDistanceBringer(city1,city2):
  city2List=[city2]
  selectedRow=df.loc[(df['İL ADI']==city1)]
  distance = selectedRow[selectedRow.columns.intersection(list(city2List))]
```

```
  return distance.iloc[0][city2]
  ##return distance # Will bring Distance between 2 city.

def solutionDistanceKMService(solution):
  totalPathKm=0;
  for index in range(0,len(solution)-1):
    city1 = solution[index]
    city2 = solution[index+1]
    distanceBetween = neighbourDistanceBringer(city1,city2)
    totalPathKm = totalPathKm+distanceBetween
  return totalPathKm

def listNeighbourDistanceList(state,neighbourList):
  nList = {}
  for i in range(0,len(neighbourList)):
    nList[neighbourList[i]] = neighbourDistanceBringer(state,neighbourList[i])
  #print("I AM DISTANCE LIST",nList)
  return nList

def heuristicNeighbourDistanceList(target,neighbourList):
  nList = {}
  for i in range(0,len(neighbourList)):
    nList[neighbourList[i]] = cityHerusticService(target,neighbourList[i])
  #print("I AM HEURISTIC LIST",nList)
  return nList



def aStarListMaker(state,target,neighbourList):
  nList = {}
  for i in range(0,len(neighbourList)):
    nList[neighbourList[i]] = cityHerusticService(target,neighbourList[i])+neighb
  #print("I AM A STAR LIST",nList)
  return nList
```

```
visitedAStarSearch =[]
def aStarSearch(source, target):
  visitedAStarSearch.clear()
  explored = []
  nList = {}
  queue = [[source]]
  if source == target:
      return source
  while queue:
    path = queue.pop(0)
    node = path[-1]
    if node not in explored:
      ##print("CURRENT STATE",node)
      visitedAStarSearch.append(node)
      neighbours = graph[node]  # komşular geldi.# şimdi sortlamamız gerek.
      #Dictonary oluşturulmalı ({CityA,50}
      nList= sorted(aStarListMaker(node,target,neighbours).items(), key=lambda x:
      a,b = nList[0]
      candidatePath = list(path)
      candidatePath.append(a)
```

```
      candidatePath.append(a)
      queue.append(candidatePath)
      if a == target:
        return candidatePath
      explored.append(node)
  return "There is no way to find that node."
```

```
visitedBFS = []
def breadFirstSearch(source, target):
    visitedBFS.clear()
    explored = []
    queue = [[source]]
    if source == target:
        return source
    while queue:
        path = queue.pop(0)
        node = path[-1]
        if node not in explored:
           # print("Current State", node)
            visitedBFS.append(node)
            neighbours = graph[node]  # komşular geldi.
            #print("Neigbours : ", neighbours)
            for neighbour in neighbours:
                candidatePath = list(path)
                candidatePath.append(neighbour)
                queue.append(candidatePath)
               # print("candidatePath : ", candidatePath)
                if neighbour == target:
                    return candidatePath
            explored.append(node)
    return "There is no way to find that node."
```

```
visitedDfs = set()
def dfs_paths(source, target):
    visitedDfs.clear()
    stack = [(source, [source])]
    while stack:
        (city, path) = stack.pop()
        if city not in visitedDfs:
            if city == target:
                return path
            visitedDfs.add(city)
            for neighbor in graph[city]:
                stack.append((neighbor, path + [neighbor]))
```

```
print("I AM BFS : ",breadFirstSearch( 'GİRESUN', 'İSTANBUL'))
print("I AM VISITED SET- BFS! ", set(reversed(visitedBFS)))
print("Length of Nodes",len(set(reversed(visitedBFS))))
print("------------------------------------------------------------")
print("I AM DFS",dfs_paths('GİRESUN', 'İSTANBUL'))
print("Visited DFS ",visitedDfs)
```

```
print("Length of Nodes",len(visitedDfs)) ## Gives number of hops
print("------------------------------------------------------------")
print("I AM A* : ",aStarSearch('GİRESUN', 'İSTANBUL'))
print("I AM VISITED SET- A*! ", set(reversed(visitedAStarSearch)))
print("Length of Nodes",len(set(reversed(visitedAStarSearch))))
```

PART 1

AlgorithmSelector: DFS

City1: GİRESUN

City2: İSTANBUL

```
DFS SOLUTION : ['GİRESUN', 'ORDU', 'SİVAS', 'YOZGAT', 'NEVŞEHİR', 'AKSARAY', 'KONYA'
Visited Nodes DFS  {'BURDUR', 'SAMSUN', 'BİLECİK', 'KOCAELİ (İZMİT)', 'ÇANKIRI', 'KO
Length of Visited Nodes 38
```

PART 2

Number_Of_Tries: 10

```
City Pairs are :
-----------------------------------------------------------
DENİZLİ And  AMASYA
HATAY And  KIRŞEHİR
IĞDIR And  SAKARYA (ADAPAZARI)
ŞANLIURFA And  DİYARBAKIR
TOKAT And  UŞAK
DÜZCE And  GİRESUN
ANTALYA And  MANİSA
OSMANİYE And  BARTIN
ELAZIĞ And  MUŞ
MERSİN And  KAHRAMANMARAŞ
-----------------------------------------------------------
```

	BFS	DFS	A*
Average Nodes Visited	28.8	38.2	4.8
Average Solution Path Lenght	5.4	32	5.8
Average Solution Cost in KM	803.5	5577.8	692.4