Link Colaboratory untuk Code nya :
https://colab.research.google.com/drive/1Cxij4XlGsFXwc3UnnHtjTXNm_RnoXf9I?usp=sharing

```
def find(item, arr, N):
    con = True
    co = 0
    while con and co < N:
        if(arr[co] == item):
            con = False
        co += 1
    return con

def alfa(num):
    if num == 0:
        return 'A'
    elif num == 1:
        return 'B'
    elif num == 2:
        return 'C'
    elif num == 3:
        return 'D'
    elif num == 4:
        return 'E'
```

Fungsi **Find** : Untuk mencari sebuah variabel di dalam sebuah array, nantinya akan dibutuhkan di dalam fungsi **neighbour**.
Fungsi **alfa** : hanya untuk mengubah tipe data kota yang awalnya dari integer jadi char (A-E)

```
def neighbour(current, path, passed):
    allneigh = [] # untuk cari neighbor, misal dari A bisa ke B atau ke C
    for i in range(len(path)):
        if i != current and find(i, passed, len(passed)):
            allneigh.append(i)

    best = allneigh[0]
    cost = path[current][allneigh[0]]
    for i in range(1,len(allneigh)):
        if path[current][allneigh[i]] < cost and path[current][allneigh[i]] > 0:
            best = allneigh[i]
            cost = path[current][allneigh[i]]
    print(alfa(current),'->',alfa(best))

    return best, cost
```

Fungsi **neighbour** : untuk menentukan neighbor apa saja yang terdapat dalam suatu daerah yang dimana nantinya neighbor yang ada dipilih yang terbaik (**best neighbor**) untuk dipilih

selanjutnya dalam menentukan rute terkecil. Best neighbour disini berarti memiliki jarak terpendek.
Variabel **current** digunakan untuk menentukan posisi sekarang berada di mana salesman itu.
Variabel **path** digunakan untuk menyimpan jarak setiap kota ke kota lain.
Variabel **passed** digunakan untuk memberitahu kita jalur apa saja yang telah dilalui oleh salesman.
Variabel **allneigh** digunakan untuk menyimpan semua neighbor yang bisa dilewati, misalnya start di kota A, maka salesman bisa lanjut ke kota B atau ke kota C atau ke kota E. Dengan kata lain, fungsi ini digunakan untuk memberitahu kita semua kota yang bisa dilalui atau dilewati atau semua neighbor yang ada.
Kemudian, terdapat pengulangan atau for terhadap variabel **i** untuk setiap **range(len(path))** bahwa jika variabel **i** tidak sama dengan value dari **current** and **find(i, passed, len(passed))**, maka **allneigh** akan ditambahkan sesuai dengan value dari variabel **i**.
Variabel **i** digunakan sebagai deklarasi counter untuk looping.
find(i, passed, len(passed)) digunakan untuk menemukan atau mengecek apakah kota yang sedang dicek sudah pernah dilewati atau belum, jika sudah maka akan dilakukan looping ulang, jika belum maka **allneigh** akan ditambahkan.

best = buat menyimpan best kota/ best neighbour
cost = cost dari best neighbour **allneigh [0]**-> semua kota yang bisa dilewati atau semua neighbour yang ada
Perulangan ini buat membandingkan semua neighbour dan dicari yang nilainya paling minimum kenapa difor itu dimulai dari 1 karena index yang ke 0 sudah masuk kebest dan cost nya.
Di if untuk membandingkan neighbour apabila neighbour lebih kecil maka best diganti dengan isi **allneigh[i]** dan **cost**nya diganti dengan cost neighbour yang lebih kecil setelah itu direturn ke best dan cost yang diatas.

```
def Hillclimbing(path, start, end):
    passed = [start]
    current = start
    cost = 0

    while current != end:
        tmpcurr, tmpcost = neighbour(current, path, passed)
        #print(tmpcurr)
        passed.append(tmpcurr)
        cost += tmpcost
        current = tmpcurr

    return passed, cost
```

Fungsi **Hillcliming**
passed = buat nyimpan kota yg sudah di lewati.
karena **current** = start berarti nilai awal **passed** adalah kota yang di input user.

fungsi while **current != end** kalau misalnya kota saat ini bukan goal / kota tujuannya, maka dia akan jalankan perintah dibawahnya yang berfungsi untuk mendapatkan posisi saat ini dan dia bisa jalan kemana saja dengan sebelumnya mengecek **cost** yang paling rendah.
Kalau misalnya sudah di bandingkan antara kota A dan kota B dan ternyata kota A lebih kecil costnya maka dia akan menambahkan **tmpcost** ke variabel **cost** dan menambahkan kota yang terpilih itu ke **passed**, lalu dia akan mereturn kan nama kotanya dan total cost nya

```
def main():
    path = [[0,52,0,75,30],
    [52,0,30,0,0],
    [75,30,0,52,75],
    [75,75,52,0,40],
    [30,75,75,40,0]]

    print("Input Start Poin: ")
    start = input().upper()
    print("Input Destination: ")
    end = input().upper()
    start = ord(start)-65
    end = ord(end)-65

    hasil, cost = Hillclimbing(path, start, end)
    for i in range(len(hasil)):
        hasil[i] = alfa(hasil[i])
    print(hasil)
    print('Cost = ',cost)

if __name__ == "__main__":
    main()
```

Lalu di **main()**, saat program berjalan user akan diminta menginput kota start dan kota tujuan, lalu akan langsung dijalankan fungsi **Hillclimbing**, yang nantinya hasil akan disimpan di variabel **hasil** dan **cost**.