**PRAKTIKUM KECERDASAN BUATAN**

MENCARI JALUR TERCEPAT MENGGUNAKAN ALGORITMA SEARCHING

“Disusun Untuk Memenuhi Tugas Mata Kuliah Praktikum Kecerdasan Buatan”

Dosen pengampu : Fitri Nuraeni, S.Kom., M.Kom.



Disusun Oleh :

|  |  |
| --- | --- |
| Yoga Agustiansyah | 2206050 |
| Ahmad Nur Sahid | 2206042 |
| Mochammad Dimas Nugraha | 2206043 |
| Ramada Gumelar | 2206044 |

Teknik Informatika B

**PROGRAM STUDI TEKNIK INFORMATIKA**

**INSTITUT TEKNOLOGI GARUT**

**2024**

**Studi Kasus** Mencari Jalur Tercepat Menggunakan Algoritma Searching

**Kasus**

Pastikan anda dan anggota kelompok anda mengikuti intruksi dibawah ini:

1. Buatlah *graph* atau *tree* yang menghubungkan titik lokasi rumah/ kostan masing-masing anggota kelompok dan titik lokasi kampus ITG sebagai simpul. Gunakan jarak berdasarkan *google maps* sebagai nilai bobot tiap sisi.
2. Tambahkan informasi jumlah tempat kuliner disetiap sisi sesuai informasi *google maps*. Diasumsikan bahwa setiap satu tempat kuliner akan menambah waktu tempuh selama 5 menit, karena macet berburu takjil & menu buka puasa.
3. Jika kelompok anda akan melaksanakan Buka Puasa Bersama di Kampus ITG. Perjalanan dimulai dari titik lokasi Ketua Kelompok yang menjemput anggota ke-1, lalu menjemput anggota ke-2, dan seterusnya, sampai berangkat bersama seluruh anggota kelompok dan berakhir di titik lokasi kampus ITG.
4. Gunakan algoritma pencarian dengan kinerja yang sesuai dengan kondisi permasalahan masing-masing kelompok, untuk mendapatkan waktu tempuh paling cepat!

**Algoritma Yang Digunakan**

Simulated Annealing

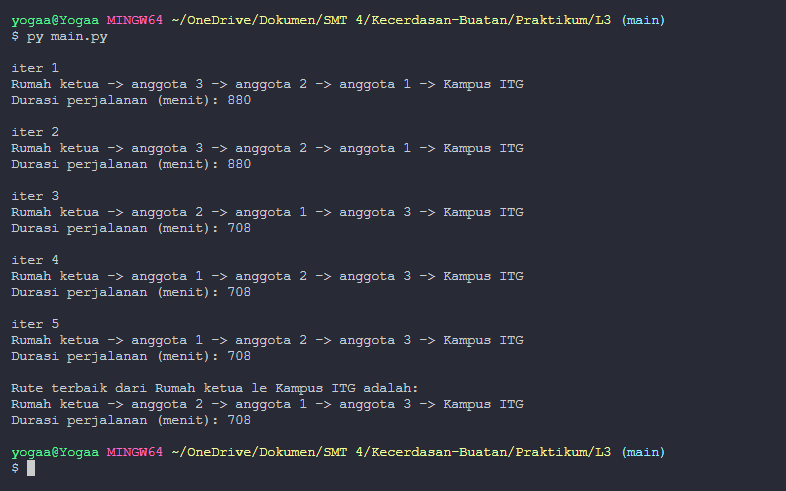
Keterangan

Selesai di lab

**Kode Program**

|  |  |
| --- | --- |
| Source Code   |  | | --- | | import random  import math  def rute\_acak(*awal*, *tujuan*):      titik = awal      rute = [titik]  *# mendapatkan semua jalan yang harus di kunjungi*      kunjungan = list(graph.keys())    *# mengeluarkan titik awal dan tujuan*      kunjungan.pop(kunjungan.index(titik))      kunjungan.pop(kunjungan.index(tujuan))      while kunjungan:          rute\_baru = list(graph[titik].keys())          titik\_baru = random.choice(rute\_baru)    *# titik yang dipilih sudah pernah dikunjungi*          if titik\_baru not in kunjungan:              continue  *# menambahkan titik yang baru saja dikunjungi*          rute.append(titik\_baru)          kunjungan.pop(kunjungan.index(titik\_baru))          titik = titik\_baru      rute.append(tujuan)      return rute  def hitung\_nilai\_rute(*rute*):      cost = 0      for n in range(len(rute)-1):          titik = rute[n]         *# titik ke n*          next\_titik = rute[n+1]  *# titik selanjutnya dari n*  *# dapatkan jumlah rumah makan*          jml\_makan = jumlah\_rumah\_makan[titik][next\_titik]          jarak = graph[titik][next\_titik]  *# hitung total waktu yang ditempuh*          cost += jarak + (jml\_makan \* 5)      return cost  def simulated\_analing(*awal*, *tujuan*, *max\_iterasi*, *suhu*, *alpha*):  *"""fungsi untuk mencari solusi rute terbaik menggunakan simulated analing*  *"""*  *# tetapkan solusi terbaik untuk saat ini*      solusi\_terbaik = rute\_acak(awal, tujuan)      rute\_sekarang = solusi\_terbaik  *# lakukan itersi sebanyak `max\_iterasi`*      for i in range(max\_iterasi):  *# dapatkan rute baru untuk setiap iterasi*          rute\_baru = rute\_acak(awal, tujuan)    *# delta panjang panjang rute*          delta\_panjang\_jalur = hitung\_nilai\_rute(rute\_baru) - hitung\_nilai\_rute(rute\_sekarang)  *# cek jika delta pakna*          if delta\_panjang\_jalur < 0:              rute\_sekarang = rute\_baru          else:  *# rumus simulated\_analing*              p = random.random()              if p < math.exp(-delta\_panjang\_jalur / suhu):                  rute\_sekarang = rute\_baru  *# cek jika rute sekarang lebih baik dari solusi terbaik*          if hitung\_nilai\_rute(rute\_sekarang) < hitung\_nilai\_rute(solusi\_terbaik):              solusi\_terbaik = rute\_sekarang  *# debug*          print("")          print(f"iter {i+1}")          print(" -> ".join(rute\_sekarang))          print(f"Durasi perjalanan (menit): {hitung\_nilai\_rute(rute\_sekarang)}")            suhu \*= alpha      return solusi\_terbaik  *# jarak tempuh / menit*  *# ketua : ahmad nur*  *# anggota 1 : yoga*  *# anggota 2 : dimas*  *# anggota 3 : ramadhan*  graph = {      'Kampus ITG': {"Rumah ketua": 29, 'anggota 1': 13, 'anggota 2': 21, 'anggota 3': 51},      'Rumah ketua': {"Kampus ITG": 29, 'anggota 1': 38, 'anggota 2': 45, 'anggota 3': 78},      "anggota 1": {'Kampus ITG': 13, 'Rumah ketua': 38, 'anggota 2': 10, 'anggota 3': 57},      'anggota 2': {'Kampus ITG': 21, 'anggota 1': 10, 'rumah ketua': 45, 'anggota 3': 64},      'anggota 3': {'Kampus ITG': 51, 'anggota 1': 57, 'anggota 2': 64, 'rumah ketua': 78}  }  *# dari ketua ke setiap anggota*  jumlah\_rumah\_makan = {      'Kampus ITG': {"Rumah ketua": 50, 'anggota 1': 10, 'anggota 2': 13, 'anggota 3': 35},      'Rumah ketua': {"Kampus ITG": 50, 'anggota 1': 24, 'anggota 2': 27, 'anggota 3': 83},      "anggota 1": {'Kampus ITG': 10, 'Rumah ketua': 24, 'anggota 2': 3, 'anggota 3': 44},      'anggota 2': {'Kampus ITG': 13, 'anggota 1': 3, 'rumah ketua': 27, 'anggota 3': 47},      'anggota 3': {'Kampus ITG': 35, 'anggota 1': 44, 'anggota 2': 47, 'rumah ketua': 83}  }  awal = 'Rumah ketua'  tujuan = 'Kampus ITG'  rute = simulated\_analing(awal, tujuan, 5, 100, 0.9)  print()  print(f"Rute terbaik dari {awal} le {tujuan} adalah: ")  print(" -> ".join(rute))  print(f"Durasi perjalanan (menit): {hitung\_nilai\_rute(rute)}") | |

Output



Graph

