Prezentare soluție concurs programare

Soluția începe prin importarea modulelor necesare și definirea funcțiilor utilizate pentru realizarea unei soluții.

Funcțiile folosite în rezolvarea problemei sunt:

1. Funcția ‚**getLines**', care primește ca intrare numele unui fișier. Aceasta deschide fișierul, citește liniile folosind metoda ‚**readLines()**' și le stochează în lista ‚**file_lines**'. Funcția returnează lista de linii.
2. Funcția ‚**getCoordinates**', care primeste ca intrare o listă de linii, furnizată de funcția anterioară. Aceasta caută linia „NODE_COORD_SECTION\n" în listă pentru a determina unde încep indicii și coordonatele punctelor. Găsește, de asemenea, linia „GTSP_SET_SECTION:\n" pentru a ști unde se termină enumerarea coordonatelor. Utilizând indicii de început și sfârșit obținuți, funcția parcurge liniile în intervalul specificat și extrage coordonatele. Fiecare linie este curățată de caracterul newline, împărțită în valori individuale și convertită în numere întregi. Coordonatele extrase sunt apoi adăugate ca tupluri în lista ‚**coordinates_list**'. În final, funcția returnează lista de coordonate.
3. Funcția ‚**distanceCalculator**', care primeste ca intrare două tupluri, fiecare reprezentând indicele și perechea de coordonate a unui punct dat. Aceasta despachetează tuplurile pentru a obține indexul și coordonatele menționate anterior. Distanța euclidiană între cele două puncte este calculată folosind formula distanței, cu modificarea că rezultatul este un număr întreg, rotunjit în sus la cel mai apropiat întreg de la valoarea ei flotantă.
4. Funcția ‚**nearestNeighbour**' primește ca intrare o listă de puncte și numărul de puncte de vizitat. Inițializează variabila ‚**shortest_path**' cu valoarea ‚None' și variabila ‚**shortest_distance**' cu un număr foarte mare. Aceasta parcurge fiecare punct din listă ca punct de pornire. Creează o copie a listei originale de puncte și înlătură punctul de pornire curent. Punctul curent este adăugat în lista ‚**path**', iar distanța totală este inițializată cu **0**. Funcția intră apoi într-un ciclu în care selectează punctul cel mai apropiat dintre punctele rămase și îl adaugă la cale. Distanța dintre punctul curent și punctul cel mai apropiat este adăugată la distanța totală. Punctul cel mai apropiat este eliminat din mulțimea punctelor rămase. Acest proces continuă până când numărul dorit de puncte de vizitat este atins. Odată ce ciclul se încheie, funcția adaugă indexul punctului de pornire la sfârșitul căii pentru a completa ciclul. Distanța de la ultimul punct vizitat înapoi la punctul de pornire este calculată folosind funcția ‚**distanceCalculator**' și adăugată la distanța totală. Dacă distanța totală a căii curente este mai mică decât ‚**shortest_distance**' găsită până în acel moment, calea curentă și distanța devin noua cea mai scurtă cale și distanță. După ce a parcurs toate punctele posibile de pornire, funcția returnează cea mai scurtă cale și distanța.
5. Funcția ‚**populateOutputFile**', care primește numele fișierului de ieșire, calea cea mai scurtă și distanța totală ca intrare. Deschide fișierul de ieșire în modul de scriere și scrie în el lungimea căii, ordinea punctelor vizitate și distanța totală. Funcția parcurge apoi fiecare fișier din folderul specificat folosind ‚**os.listdir()**'. Pentru fiecare fișier, extrage numele fișierului de intrare și de ieșire, citește liniile fișierului de intrare, calculează valorile necesare, găsește cea mai scurtă cale și scrie rezultatele în fișierul de ieșire folosind funcția ‚**populateOutputFile**'.

Codul oferă o modalitate flexibilă de a procesa fie un singur fișier de intrare, fie mai multe fișiere de intrare dintr-un folder dat. Utilizează algoritmul vecinului cel mai apropiat pentru a găsi cea mai scurtă cale pe baza procentului specificat de puncte de vizitat. Rezultatele sunt scrise în fișiere de ieșire.