# Descriere soluție – TSP

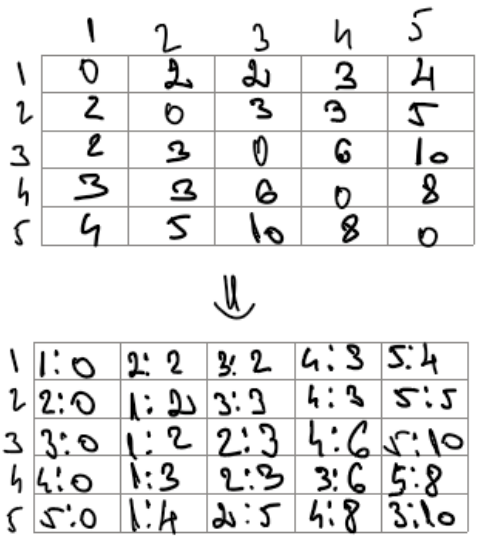
Problema dată este una clasică specifică Travelling Salesman Problem, fiind una dintre cele mai studiate probleme. Mai pe scurt, ideea este de a genera cel mai scurt traseu care să pornească dintr-un punct, să treacă prin n puncte diferite si să revină în punctul inițial. Cerința problemei a fost de a citi coordonatele și dimensiunea (`x`) dintr-un fișier GTSP, respectiv de a genera cel mai scurt traseu care să treacă prin n puncte (`n = p% * x`) și dimensiunea acestuia aproximat la cel mai apropiat număr întreg. Limbajul folosit pentru acest proiect a fost C++, iar mediul de lucru a fost CLion de la JetBrains.

1. Citirea datelor din fișier

   Fiecare linie citită din fișier a fost de tip string (`#include <string>`), iar în cazul în care linia conținea substringul „DIMENSION”, atunci întregului text i s-a aplicat un split pe caracterul „:”. Astfel, am avut un șir de 2 elemente, având pe prima poziție cuvântul „DIMENSION”, iar pe a doua poziție numărul de coordonate. În cazul în care stringul era egal cu „NODE_COORD_SECTION”, atunci următoarele x citiri erau cele ale coordonatelor.

2. Generarea matricii de distanțe

   Pentru a putea determina cel mai scurt drum, m-am gândit să generez o matrice de distanțe. Astfel că pentru o matrice a de dimensiune x*x (x fiind numărul total de coordonate) am atribuit fiecărui element a[i][j] o pereche de numere ce avea drept componente punctul j, respectiv distanta dintre i și j. La final, după ce am calculat distanțele pe linia i, am sortat această linie pentru a avea distanțele cele mai mici pe primele poziții, cu scopul de a optimiza puțin programul și de a nu căuta pe parcursul executării minimul pentru fiecare poziție. Tipul de sortare este QuickSort, folosit din librăria `<algorithm>.`
   Un exemplu de matrice de distanță este:

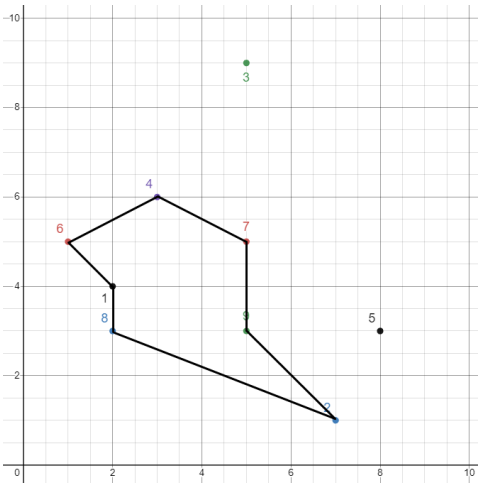


3. Generarea rutelor

   Fiecare drum a fost generat, cu ajutorul matricii de distanțe. Am creat o funcție `generateRoute()` ce are ca și argumente un număr întreg, reprezentând punctul de pornire al drumului, și un alt număr întreg ce reprezintă numărul de puncte diferite ce trebuie vizitate. Din punctul start am încercat să parcurg fiecare posibilitate de punct cu care s-ar putea lega acel punct, adică am parcurs tot vectorul a[start], generând diferite posibilități de rute. După aceea, următoarele n-1 puncte vor fi minimele din matrice. După ce s-a terminat generarea unei posibile rute, aceasta va fi comparată cu o rută minimă, iar în cazul în care drumul este cel mai scurt, va fi memorat.
   Cu ajutorul acestei funcții, voi genera drumuri pornind de la toate punctele citite, astfel încât să nu existe drumuri minime care să nu fi fost parcurse.

Un exemplu de fișier, respectiv date de intrare ar putea fi:

| fisier.gtsp | console |
|---|---|
| ...<br>DIMENSION: 9<br>...<br>NODE_COORD_SECTION<br>1 2 4<br>2 7 1<br>3 5 9<br>4 3 6<br>5 8 3<br>6 1 5<br>7 5 5<br>8 2 3<br>9 5 3<br>EOF | 75<br>fisier.gtsp |
| fisier.out | --- Puncte vizitate<br>7<br>--- Ordinea de vizitare<br>8, 1, 6, 4, 7, 9, 2, 8<br>--- Distanța totală calculată<br>14 |




Explicația exemplului este faptul că cel mai scurt drum (trecând prin 7 puncte – deoarece p = 75, iar 75% din 9 este ~7) pornește din punctul 8, trece prin punctele 1, 6, 4, 7, 9 și 2, iar la final revine în punctul 8, având distanța 14. Reprezentarea grafică a drumului este dată alături.