Задачата за търговския пътник
(Валери Валериев Станчев)

1. Формулировка:

Търговски пътник трябва да посети списък от градове. Ако е известно разстоянието между всяка двойка градове, кой е най-късият път, който посещава всеки град точно по веднъж и се връща в града, от който е започнало пътуването?

2. Преобразуване към графова задача:

Даден е неориентиран тегловен пълен граф без примки, в който върховете са градовете, а ребрата – връзките между тях с тегло равно на разстоянието между градовете. Да се намери Хамилтоновият цикъл с минимална сумарна тежест, получена от всички ребра в намерения път.

3. Първоначален анализ:

Има точни алгоритми, които решават задачата бързо само за малки подадени данни. От друга страна, има евристични алгоритми, които завършват в добро време за големи данни, но правят апроксимация на търсения път, т.е. не гарантират оптималност.

4. Brute Force algorithm (точен алгоритъм):

Най-наивното, но коректно и просто решение, е да се генерират всички пермутации на върховете и да се вземе тази, която има най-малко сумарно тегло.

Памет: $O(n^2)$
Време: $O(n!)$

5. Dynamic programming, Held-Karp algorithm (точен алгоритъм):

Друго решение, значително по-бързо, но все пак бавно като цяло, използва факта, че всеки оптимален път се състои от оптимални подпътища, т.е. задачата може да се сведе до решаването на по-малки подзадачи.

Нека означим с $dp[i][j]$ минималното разстояние, започващо от връх 0, като сме посетили всички върхове в множеството i, като сме завършили в град j, който принадлежи на множеството. Тогава $dp[i][j]=min_{v \in i \setminus \{j\}}(dp[i \setminus \{j\}][v]+dist[v][j])$, където $dist[u][v]$ е теглото на реброто между върховете u и v. Тогава стойността на търсения минимален път ще бъде $min_{v \in \{1,2,\dots,n-1\}}(dp[\{1,2,\dots,n-1\}][v]+dist[v][0])$.

Памет: $O(n * 2^n)$
Време: $O(n^2 * 2^n)$

6. Nearest Neighbour algorithm (евристичен алгоритъм):

Основната идея на това решение е на всяка стъпка да се избере най-близкия недобавен все още връх. По-конкретно:

1. Избира се произволен връх.
2. Намира се най-близкият непосетен съсед на текущия връх, прави се посетен и се отива на него.
3. Ако има непосетени върхове, повтори стъпка 2.
4. Върни се в първоначално избрания връх.

Памет: $O(n^2)$
Време: $O(n^2)$

7. Fathest Insertion algorithm (евристичен алгоритъм):
   1. Нека текущият ни път е съставен от 2 върха, крайща на ребро с голямо тегло.
   2. Ако всички върхове са посетени, премини на стъпка 4. Иначе, за всеки непосетен връх изчисли минималното разстояние между него и който и да е връх от текущия път. Намери върхът, чието разстояние от изчислените е максимално.
   3. Намери последователната двойка върхове от пътя, в която е най-добре да добавим намерения връх във 2ра стъпка, така че сумарното тегло от всички ребра на новополучения път да е минимално.
   4. Добави първоначално избрания връх в пътя.

   Памет: $O(n^2)$
   Време: $O(n^2)$

Използвани източници:

1. https://en.wikipedia.org/wiki/Travelling_salesman_problem
2. https://en.wikipedia.org/wiki/Held%E2%80%93Karp_algorithm
3. Heuristics for the Traveling Salesman Problem, Christian Nilsson
4. The Traveling Salesman Problem - MIT Math