0.1 Traveling salesman problem

- *Algorithm:* Christofides Algorithm (algo. 1)
- *Input:* a universe of cities $V = \{1, 2, ..., n\}$, a metric distance function $d : V \times V \to R^+$.
- *Complexity:* $\mathcal{O}(n^2 \log n)$, for approximate algorithm, actually we care more about the accuracy.
- *Data structure compatibility:* graph and matrix
- *Common applications:* supply chain management and chip production

Problem. Traveling salesman problem

We assume a universe of cities $V = \{1, 2, ..., n\}$, the metric distance function is $d : V \times V \to R^+$. We want to find a path(a traverse of the cities) $k(1), k(2), ..., k(n)$, such that the total cost $\sum_{i=1}^{n-1} d_{k(i)k(i+1)} + d_{k(n)k(1)}$ is minimized.

Description

For a simple introduction to approximate algorithm, please refer to problem 71, or read the book above, or take course VG441, which is a course offered by JI. Unlike humanity course like VR477 introduction to humanity project, VG441 is a real CS course.

TSP, shorthand for traveling salesman problem originates from a salesman that wants to travel through every city exactly once using shortest distance. It is a classic problem in the field of operations research and computer science. It is related to problems like traveling purchaser problem. We would like to research on TSP because it has tons of applications in real life. For example supply chain management has to take TSP into account so that the goods are delivered in a fast and cheap way. TSP can also be modified to cope with problems in chip production and DNA measurement. If you are interested in TSP and wants to study further, you may refer to a book(collection of papers TAT) called *the Design of Approximate Algorithms*[3]. You will find variation like asymmetric TSP and euclidean TSP.

Detailed definition of this problem is shown below. Assume we have a universe of cities $V = \{1, 2, ..., n\}$, and the symmetric distance between every city pair is defined as $C = (c_{ij})$. We assume a distance function to be $d : V \times V \to R^+$. The distance can have meanings like physical distance, travel expenses, travel time cost and so on. Notice that the cost should be nonnegative. Since there does not exist an approximation algorithm if we do not restrict the cost to be metric[2]. So we should specify the cost to be metric. here metric means the cost be symmetry and it should satisfy the triangular inequality. Given the cities and the cost function between them, we would like to find a path with minimum cost that visits every city in the universe exactly once and then return to the starting point.

Time complexity analysis. For the MST, we can use the prim's algorithm taught in VE477 and the time complexity should be $\mathcal{O}(E + logV) = \mathcal{O}(n^2)$ in our case[1]. Also $\mathcal{O}(n^2)$ is enough for find the set of odd degree vertices. After that we do the minimum cost matching which is of complexity $\mathcal{O}(nm \log n)$, in our case, m is $\mathcal{O}(n)$. Add edges is linearly cheap, find a walk is also linearly cheap and we can ignore it. Shortcutting in the last step in the algorithm needs only linear time as well. In short, the time complexity of this Christofides algorithm depends on the time complexity of finding minimum cost matching.

Algorithm 1: Christofides

Input : a universe of cities $V = \{1, 2, ..., n\}$, a metric distance function $d : V \times V \to R^+$.
Output: a path that visits every city in the universe exactly once and then return to the starting point

Compute the minimum spanning tree(MST) M of (V,d)
Compute the minimum cost matching K on the vertices of M with odd degree
Add edges of K to M and we get an Eulerian graph D'
find a walk W' that travels through each edge of D' eactly once
shortcut W' by skipping revisited vertices to get a TSP path T'
return T'

References.

[1] Manuel. *VE477 – Introduction to Algorithms (lecture slides)*. 2020 (cit. on p. 1).

[2] Cong Shi. *VG441 – Supply Chain (lecture slides)*. 2020 (cit. on p. 1).

[3] David P Williamson and David B Shmoys. *The design of approximation algorithms*. Cambridge university press, 2011 (cit. on p. 1).