前言：

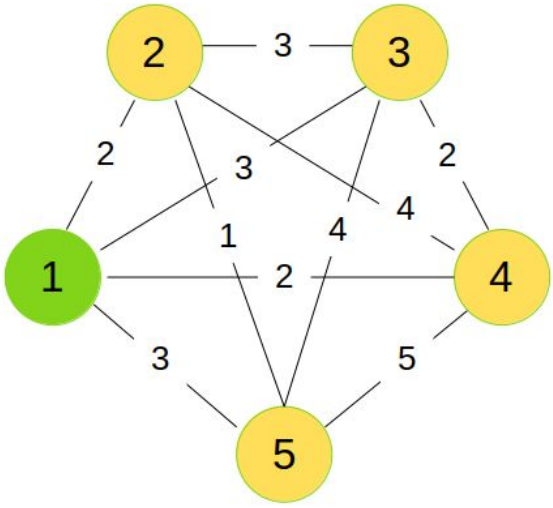
第三周的課程延續上週，講解了許多 NP 問題的近似解法，如包裝問題、Vertex Cover 等，通常這些演算法無法提供精確的解，但可以在多項式時間內，求得至少 50% 或更為接近的答案。主要的策略有：

1. Greedy Heuristic → 利用貪婪演算法(可能搭配動態規劃)求解。
2. Local Search → 找出當前鄰近節點的最佳解，作為下次迭代的當前解，直到達到一個局部的最佳解或設定的迭代次數才停止。(這也是很多最佳化算法的基本精神)

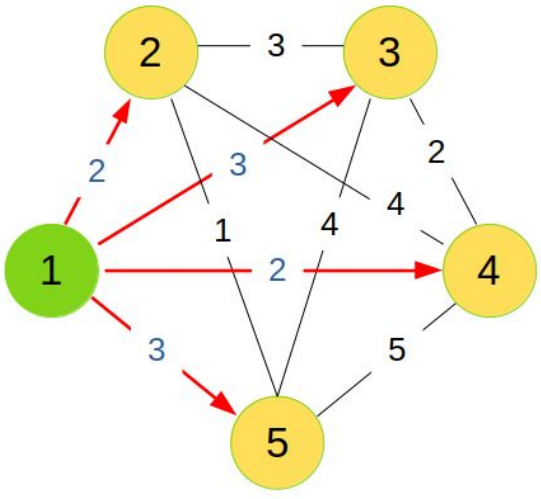
而作業比較簡單，為利用 Nearest Neighbor 來求解 TSP 問題。

Nearest Neighbor for TSP：

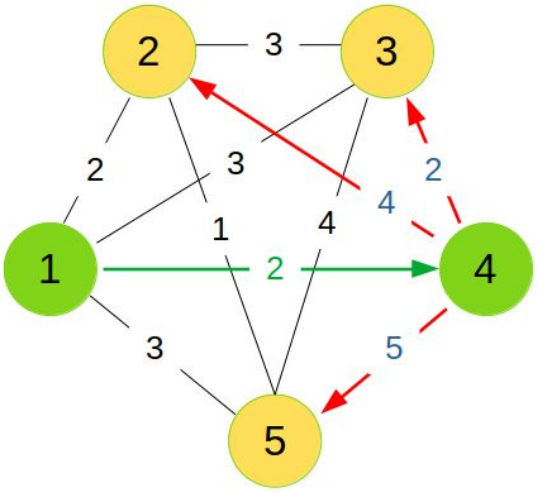
最鄰近節點算法非常直觀，算是貪婪演算法的一種，從某個點出發，並設為當前所在位置，每次都找離當前位置最近且未訪問過的點，成為下一個目的地，逐步迭代最後回到原點完成計算，如以下示範：



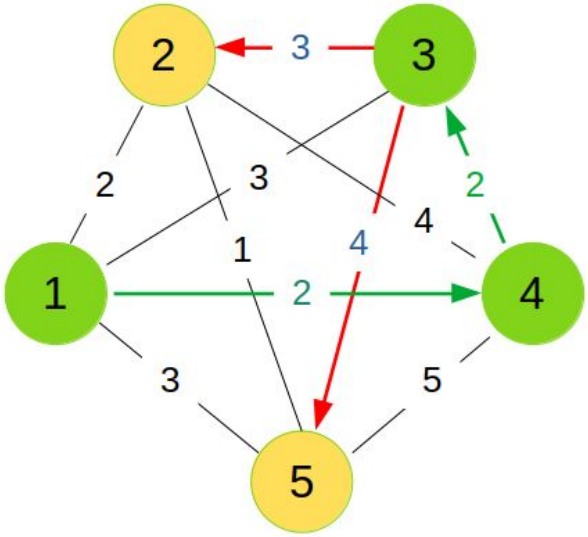
Step 1. 假設從節點 1 開始，(1, 2) (1, 4)的距離最短，從中任意選擇節點 4 並標記為已探索。



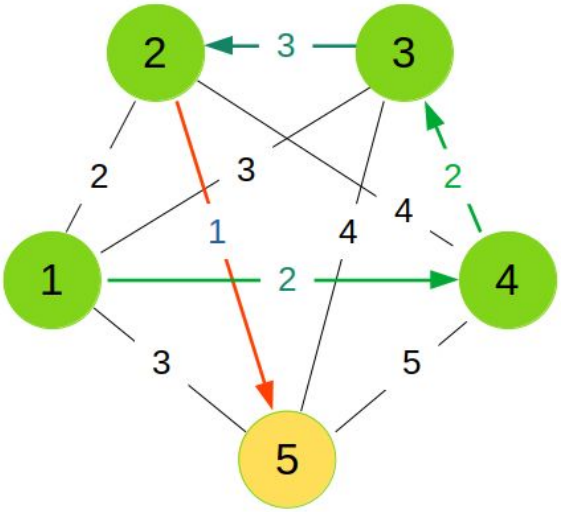
Step 2. 選擇節點 3。

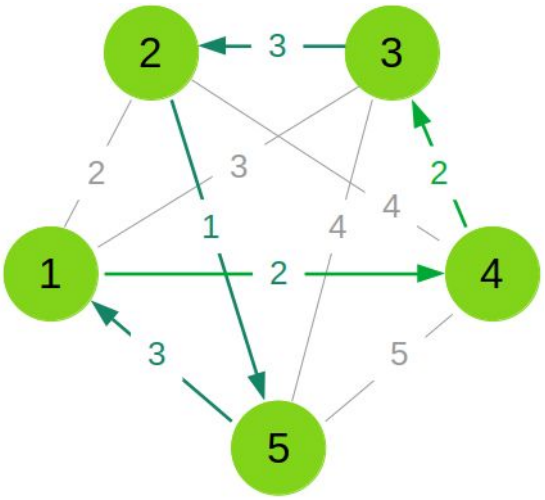


Step 3. 選擇節點 2。



Step 4. 選擇節點 5，並回到原點完成計算。

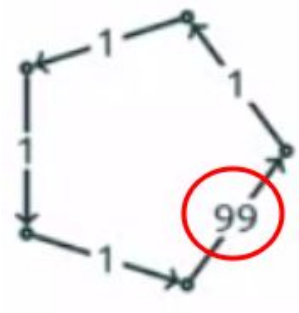




Bad Case：

Nearest Neighbor 雖然直觀、快速，但缺點也很明顯，有些情況會與最佳解有不小的落差，以下利用兩個例子來說明，Nearest Neighbor 可能會遇到的情形：

1. 起點與最後一個拜訪的點之間的權重非常大，造成遠比最佳解來的差。



2. 歐幾里得距離局部最佳解的組合，並非全局最佳解。如下方例子，最佳解約 26.42，但 Nearest Neighbor 算出來約 28.33。

作業——以 Nearest Neighbor 方法求解旅行推銷員問題 (TSP)：

問題描述：

txt 檔中每一行代表一個城市的位置，第一欄為 x 座標，第二欄為 y 座標，兩城市間的距離以歐幾里得距離定義，從第 1 點開始，以 Nearest Neighbor 計算，試求通過這 33708 個城市最後回到出發點的距離為何？

解題方法：

比起上週，這次的作業相對簡單很多，照 Nearest Neighbor 的算法求解即可。

參考資料：

http://logistics.iem.yzu.edu.tw/Courses/1041/TSP%20Problem.pdf

https://www.coursera.org/lecture/delivery-problem/nearest-neighbor-0jr4r