# ECE374 SP23 HW8

## Contributors

Zhirong Chen (zhirong4)

Ziyuan Chen (ziyuanc3)

## Problem 1

The traveling salesman problem can be defined in two ways:

- The Traveling Salesman Problem $\text{TSP}(G)$
  - **Input:** A weighted graph $G$
  - **Output:** Which tour $(v_1, v_2, \ldots, v_n)$ minimizes $\Sigma_{i=1}^{n-1} (d[v_i, v_i + 1]) + d[v_n, v_1]$
- The Traveling Salesman *Decision* Problem $\text{TSDP}(G, k)$
  - **Input:** A weighted graph $G$ and an integer $k$
  - **Output:** Does there exist an TSP tour with cost $\leq k$

***Suppose*** we are given an algorithm that can solve the traveling salesman decision problem in linear time. Give an efficient algorithm to find the actual TSP tour by making a polynomial number of calls to this subroutine.

## Solution

*Intuition.* We use binary search to find the optimal TSP cost $k$. Then we repeatedly run $\text{TSDP}(G, k)$ on the graph $G$ **with a possible "next step" edge removed** to find the actual TSP tour - if the cost raises without a specific edge, then the edge must be the right step to take.

---

*Pseudocode.*

$$
\begin{array}{l}
\text{BinarySearch}(\text{lower}, \text{upper}) \\
\quad \textbf{if } \text{upper} - \text{lower} < \text{BF\_THRESHOLD} \\
\quad\quad \text{find } k \text{ by brute force} \\
\quad \textbf{else} \\
\quad\quad \text{mid} \leftarrow (\text{lower} + \text{upper})/2 \\
\quad\quad \textbf{if } \text{TSDP}(G, \text{mid}) \\
\quad\quad\quad k \leftarrow \text{BinarySearch}(\text{lower}, \text{mid}) \\
\quad\quad \textbf{else} \\
\quad\quad\quad k \leftarrow \text{BinarySearch}(\text{mid}, \text{upper}) \\
\quad \textbf{return } k
\end{array}
$$

```
TSP(G(V, E))
    lower ← |V| × min(E)
    upper ← |V| × max(E)
    k ← BinarySearch(lower, upper)             // k holds the optimal TSP cost
    v ← any vertex in G                        // start the tour at any vertex
    tour ← [ ]                                 // ordered list!
    while tour ≠ V
        for u ∈ adj(v) − tour                  // try all possible "next steps"
            G.removeEdge(v, u)
            if not TSDP(G, k) and v ∉ tour     // if the cost raises without this edge...
                G.addEdge(v, u)
                tour ← tour ∪ {v}              // then this must be the right step!
                v ← u                          // v is "current", u is "next"
        if |tour| = |V| − 1
            tour ← tour ∪ {v}
    return tour                                // one vertex left, we have no choice
```

---

*Runtime analysis.*

- $\mathrm{TSDP}$ runs in linear time.
- $\mathrm{BinarySearch}$ runs in $\log_2(\mathrm{upper} - \mathrm{lower}) = O(\log VE)$ time.
- $\mathrm{TSP}$ runs in $O(VE)$ time (nested loop examines all neighbors of all vertices in the worst case).

Total runtime is $O(VE + \log VE) = O(VE)$.