Travelling Salesman Problem (TSP)
A well-known theoretical and practical problem:
▶ a salesman has to visit a number of cities
▶ what is the shortest route to visit all cities and return home?

Properties of the problem:
▶ hard to solve for large number of cities
▶ instance of a NP-complete problem

Complexity of problems
We have already encountered problems with different complexity:
▶ search through unsorted array: linear (ie, $O(n)$)
▶ binary search through sorted array: log (ie, $O(lg(n))$
▶ BubbleSort: $O(n^2)$
▶ MergeSort: $O(n lg(n))$

NP-complete?
▶ For some problems, no polynomial time solution is known — $O(n^c)$
  for some constant $c$. One class of these problems is called NP-complete
  (NP = non-polynomial).
▶ There may be polynomial solutions, but nobody found them so far.
▶ If efficient solution of a problem is not possible, we resort to heuristics
  that give us approximate solutions.
Other NP-hard problems

- Knapsack problem: given a set of whole numbers \(a_1, \ldots, a_n\), and an upper bound \(K\) find a subset of the numbers whose sum is of maximum value, subject to being no more than \(K\).
  
eg, for \(2, 4, 9, 11, 14\) and \(K = 25\), the subset is \(\{2, 9, 14\}\)

- Minesweeper: is a given configuration "possible"?

Example TSP: Romania

Simplified: Euclidean TSP

Greedy heuristic

- start at some point
- go to closest not visited city
Greedy heuristic: result

Improving the solution
- Swap neighboring cities, if it shortens path

Locally Optimal solution
Other improvements?

What other improvements can be made?

Practical 3

- Part A: capture positions of cities (from mouse clicks), and store them all in an array. Write a function to compute the length of a given tour.
- Part B: implement swap heuristic.
- Part C: implement 2-opt heuristic (more powerful).
- Part D: implement greedy heuristic.
- Part E: do better, with almost no extra work?