Analysis of Algorithms: Traveling Salesman

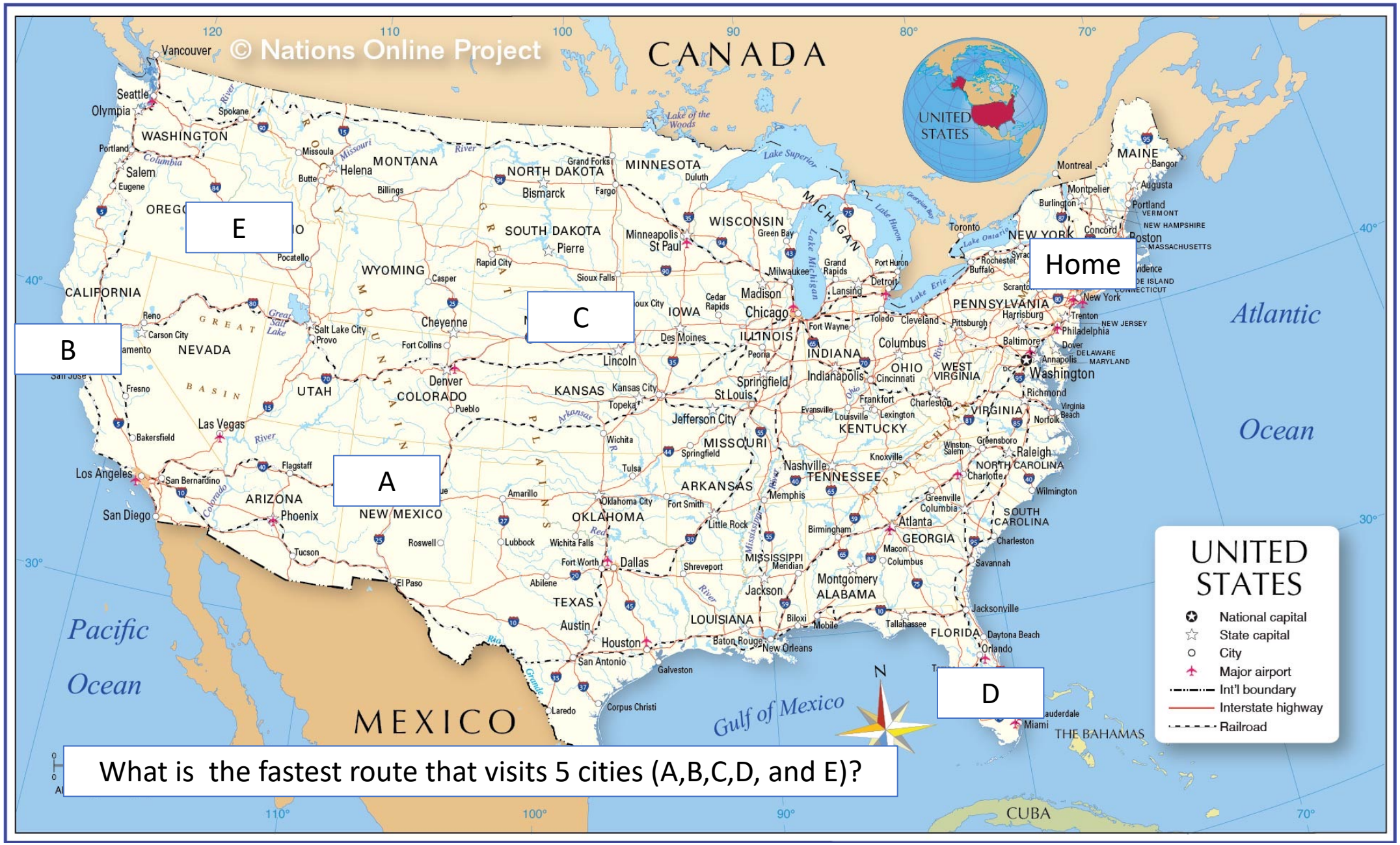
Garrett Dancik, PhD

Fall 2024

Course Notes: https://gdancik.github.io

The Traveling Salesman Problem

- Not all problems can be "solved" in a practical amount of time
- One such problem is the "Traveling Salesman" problem
  - The salesman needs to visit *n* cities and then return home.
  - We know the distances between each pair of cities
  - What is the fastest route that the salesman can take to visit each city once?



https://www.nationsonline.org/oneworld/map/usa_map.htm

Traveling Salesman Problem

- The optimal solution can be found by looking at all possible trips that the salesman can take, and picking the one with the shortest travel time. Any algorithm that involves directly looking all possibilities is a *brute force* algorithm.
- The number of possible trips can be found by using the *fundamental counting rule*.
  - For a series of decisions, if there are n_1 choices for the first decision, and n_2 choices for the next decision, and n_3 choices for the next decision, etc., then the total number of possible choices is $n_1 \times n_2 \times n_3 \ldots$

The number of ways to visit 5 cities before returning home is:

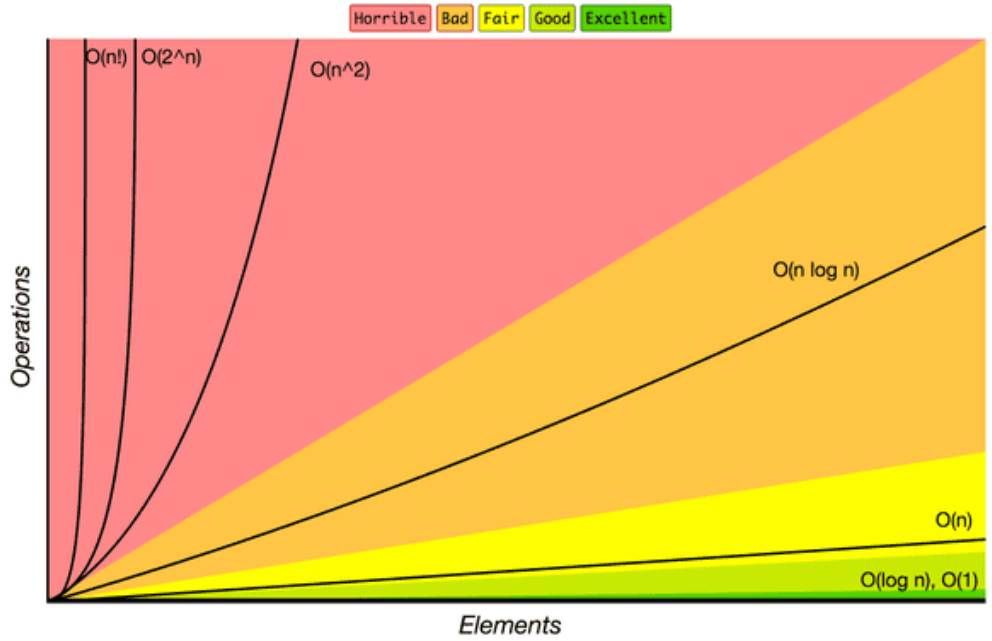
# of choices for first city		# of choices for 2nd city		# of choices for 3rd city		# of choices for 4th city		# of choices for last city
5	x	4	x	3	x	2	x	1

$$5 \times 4 \times 3 \times 2 \times 1 = 120$$

Example: How many possible outcomes are there if you flip a coin 3 times?

Traveling salesman running time

- In general, if the salesman wants to visit *n* cities, the number of possible trips is $n! = n \times (n-1) \times (n-2) \times \cdots \times 2 \times 1$
- The running time for a *brute force* traveling salesman algorithm is $\theta(n!)$. Algorithms with this order of magnitude generally cannot be run in a reasonable amount of time



Traveling Salesman Brute Force

Suppose that a computer can calculate the total travel time for 1 million routes per second.

n	$n!$	Running time
5	120	0.00012 seconds
10	3,628,800	3.6288 seconds
11	39,916,800	39.91 seconds
12	479,001,600	7 minutes 59 seconds
13	6,227,020,800	1 hour 43 minutes
15	1,307,674,000,000	over 15 days
20	2,432,902,000,000,000,000	over 77,000 years!

Traveling Salesman in practice

- For many problems, we can use a *heuristic* to return a reasonable solution
- A heuristic is a "short-cut" that finds an approximate solution to a problem; it is not guaranteed to return the optimal solution (though it may often do so in practice)
- For the Traveling Salesman problem, a heuristic algorithm is the following.
  - Start from Home
  - While (not all cities have been visited)
    - Visit the city that is the closest to your current location.
  - Return Home

Traveling Salesman in practice

- Start from Home
- While (not all cities have been visited)
  - Visit the city that is the closest to your current location.
- Return Home

Suppose we have 5 cities and they are A,B,C,D, and E

We need to look at the distance between each of the 5 cities and home: (Home → A, Home → B, Home → C, Home → D, Home → E)

One of the cities is visited (suppose it is B). Now we need to look at the distance between each of the 4 remaining cities and the one we have selected: (B → A, B → C, B → D , B → E)

Now look at distance between each of the 3 remaining cities, etc

Number of cities remaining	Number of distances to consider
5	5
4	4
3	3
2	2
1	-

The number of distances to consider is the sum of the integers between 2 and 5 (or in general between 2 and *n*). The running time is $\theta(n^2)$

Traveling Salesman Brute Force vs. Heuristic

Suppose that a computer can calculate the total travel time for 1 million routes per second.

Brute force

n	$n!$	Running time
5	120	0.00012 seconds
10	3,628,800	3.6288 seconds
11	39,916,800	39.91 seconds
12	479,001,600	7 minutes 59 seconds
13	6,227,020,800	1 hour 43 minutes
15	1,307,674,000,000	over 15 days
20	2,432,902,000,000,000,000	over 77,000 years!

Nearest city heuristic

n	n^2	Running time
5	25	0.025 ms
10	100	0.100 ms
11	121	0.121 ms
12	144	0.144 ms
13	169	0.169 ms
15	225	0.225 ms
20	400	0.400 ms