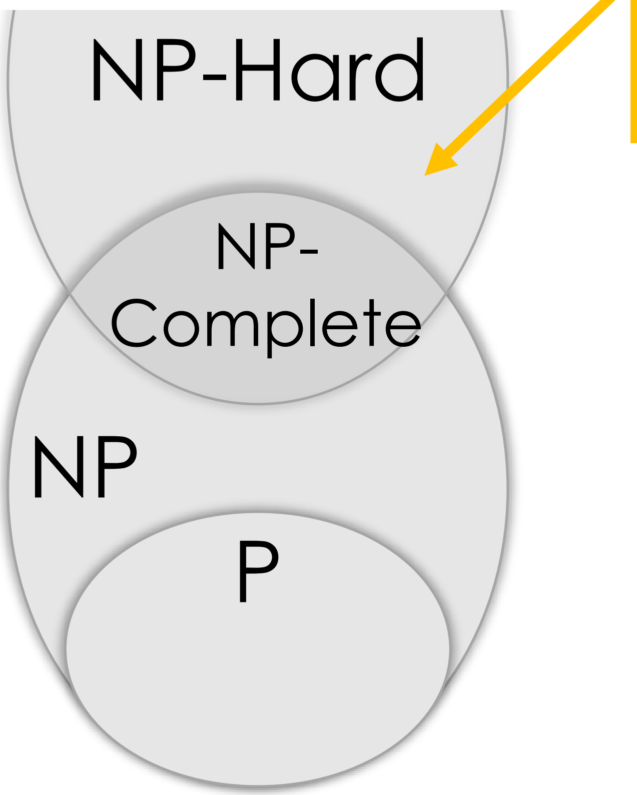
Approximations and Heuristics

Travelling Salesperson (TSP)

By the end of this video you will be able to…

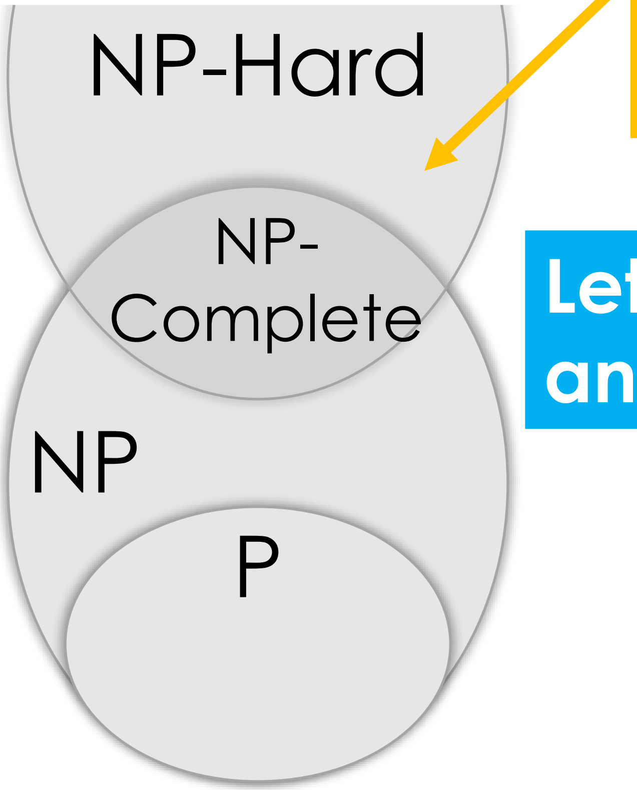
- Use heuristics to find reasonable solutions to hard problems
- Apply the 2-Opt Heuristic to the TSP

Complexity Theory



TSP "optimization": given n cities with one Hometown and all pairwise distances, plan a tour starting and ending at Hometown that visits every city exactly once and has minimum distance.

Complexity Theory



TSP "optimization": given n cities with one Hometown and all pairwise distances, plan a tour starting and ending at Hometown that visits every city exactly once and has minimum distance.

Let's relax the "minimum distance" constraint, and find a reasonable solution

Heuristics and Approximation Algorithms

Heuristics and Approximation Algorithms

In TSP, given n cities with one Hometown and all pairwise distances, plan a tour starting and ending at Hometown that visits every city exactly once and has minimum distance.

Greedy algorithm: pick best next choice

Heuristics and Approximations for TSP

Constructions:

Heuristics and Approximations for TSP

Constructions:

Build a solution

Nearest Neighbor (Greedy)

Christofides Algorithm

Heuristics and Approximations for TSP

Constructions:

Nearest Neighbor (Greedy)

Christofides Algorithm

Iterative:

Heuristics and Approximations for TSP

Constructions:

Nearest Neighbor (Greedy)

Christofides Algorithm

Iterative:

Improve a solution

k-opt and Lin-Kernighan

genetic algorithms

MORE!

Heuristics and Approximations for TSP

Constructions:

Nearest Neighbor (Greedy)

Christofides Algorithm

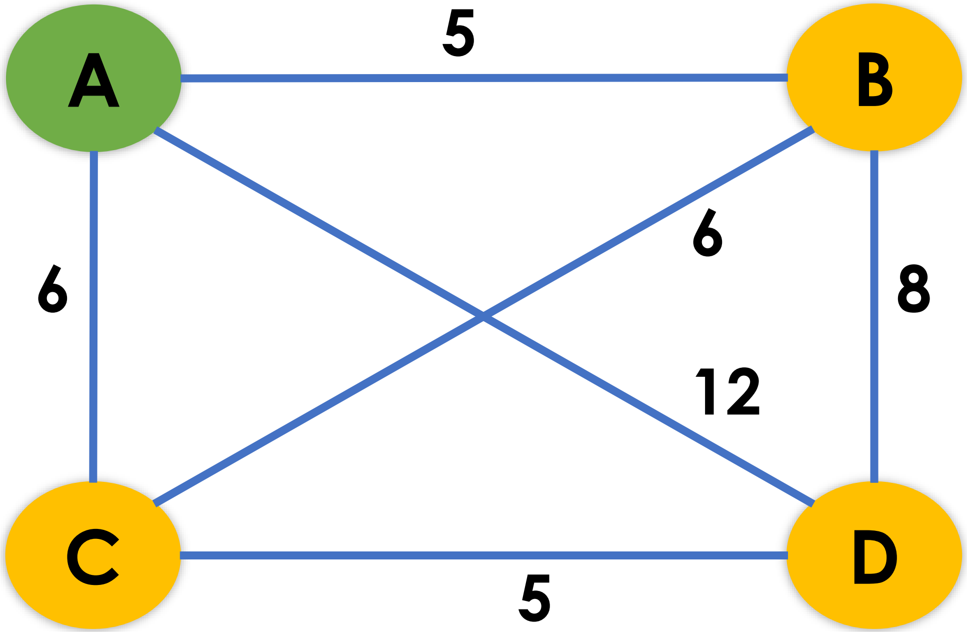
Let's examine a combination: Greedy + 2-opt

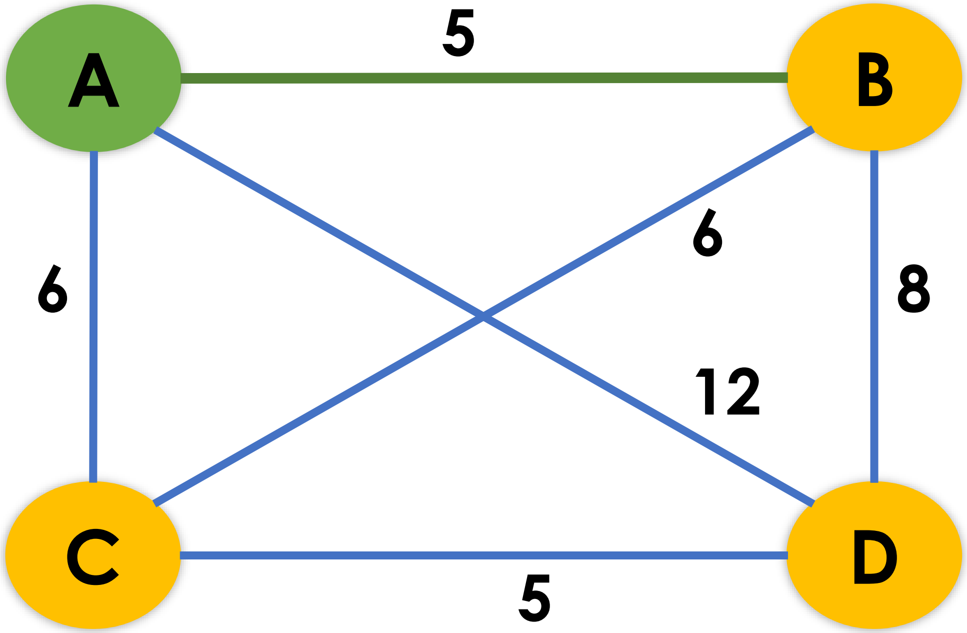
Iterative:

k-opt and Lin-Kernighan heuristics

genetic algorithms

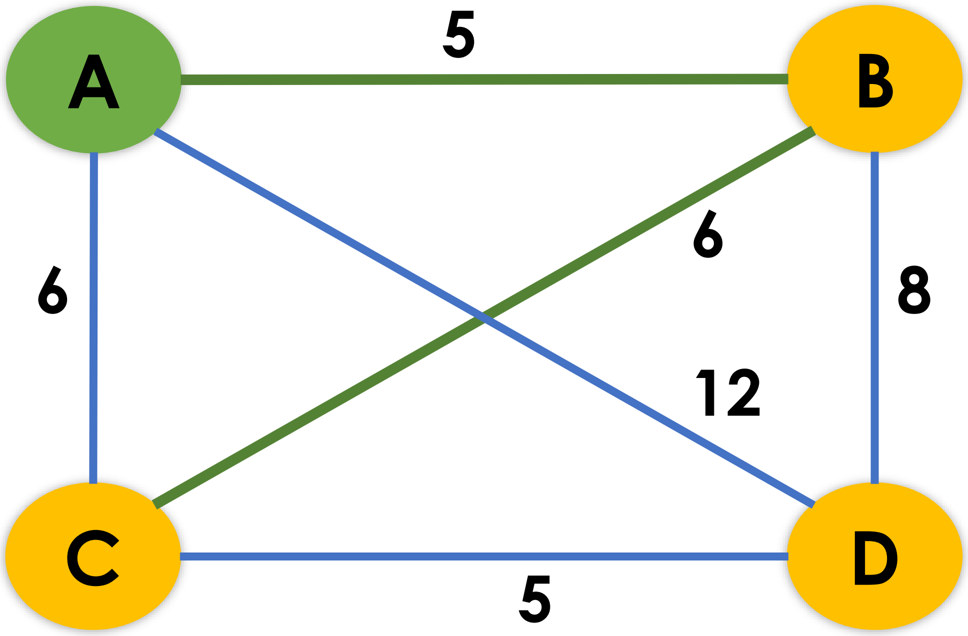
MORE!


A
5
B
6
6
8
12
C
5
D



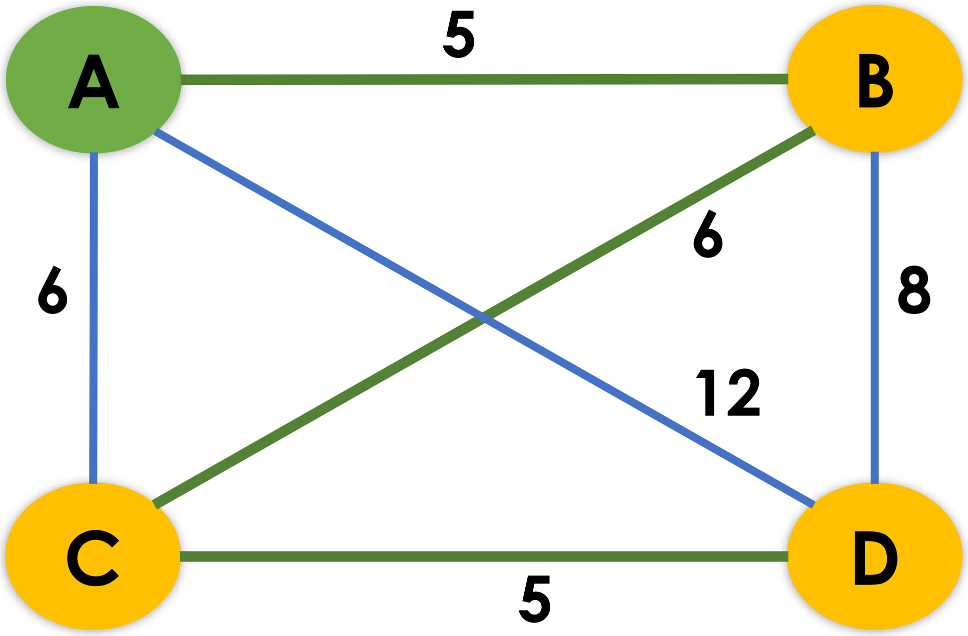
Greedy

A → B



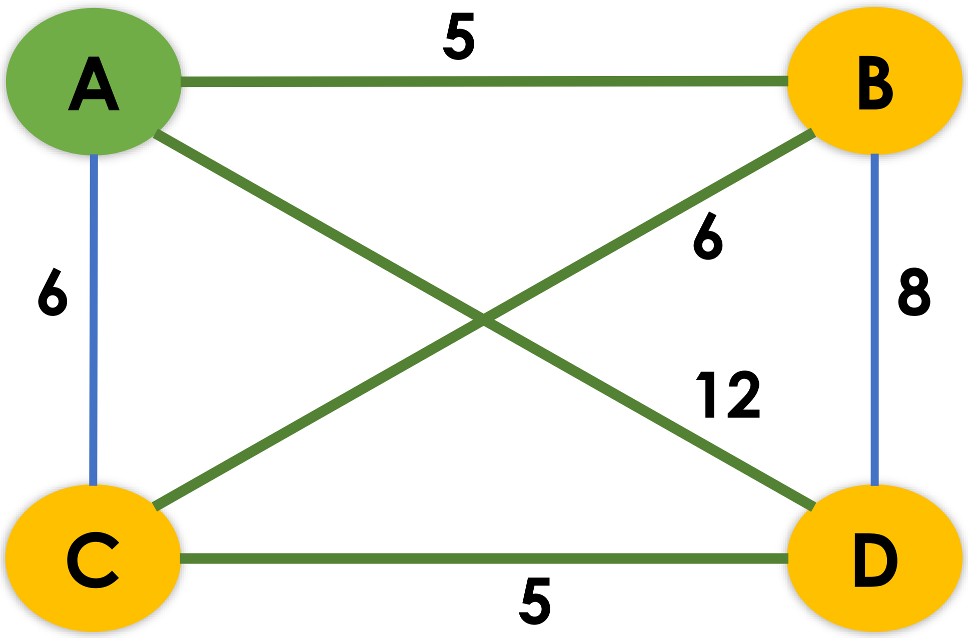
Greedy

A → B → C



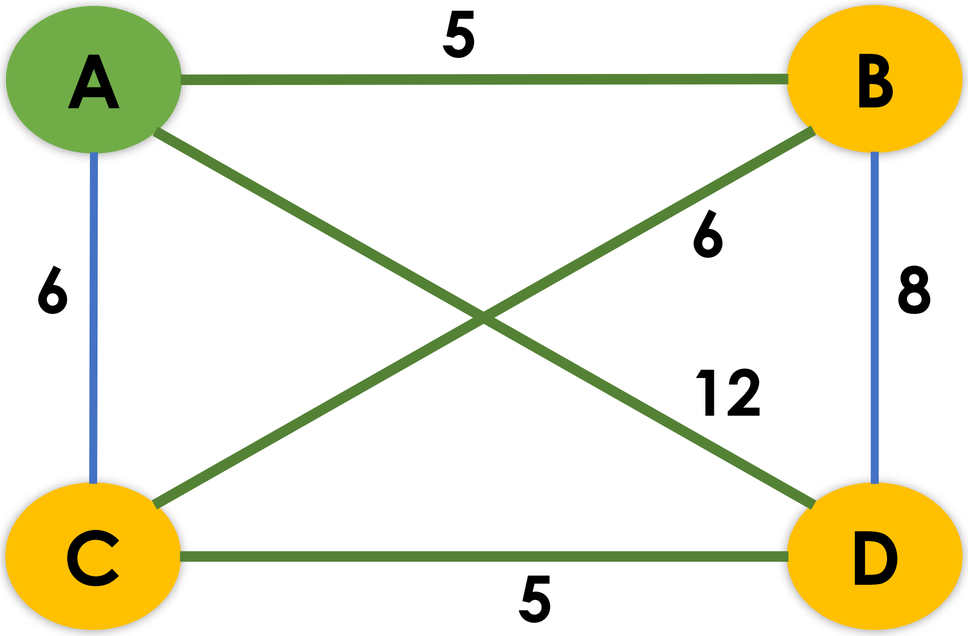
Greedy

A → B → C → D



Greedy

A → B → C → D → A L: 28

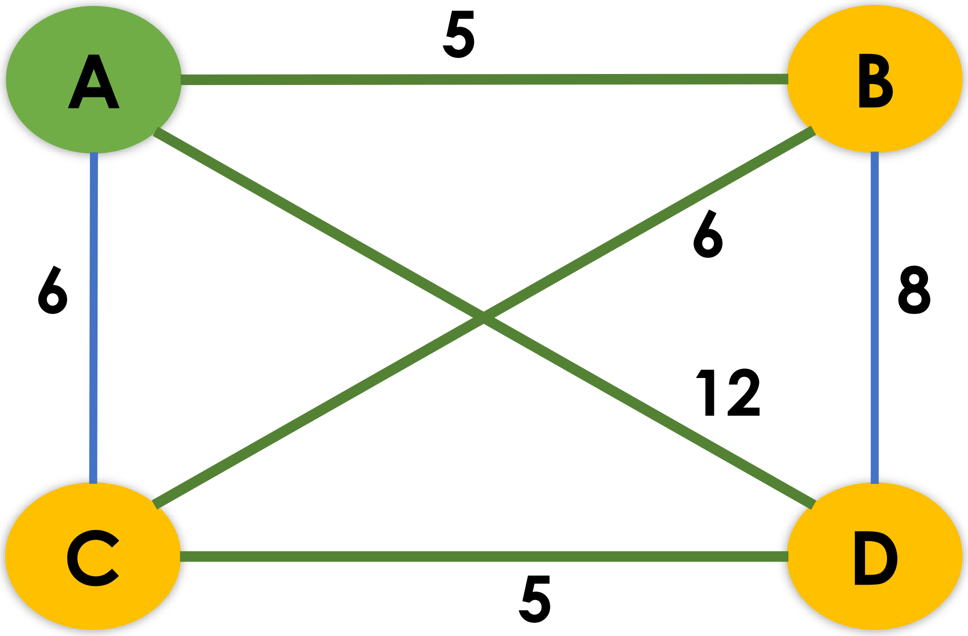


Greedy

A → B → C → D → A L: 28

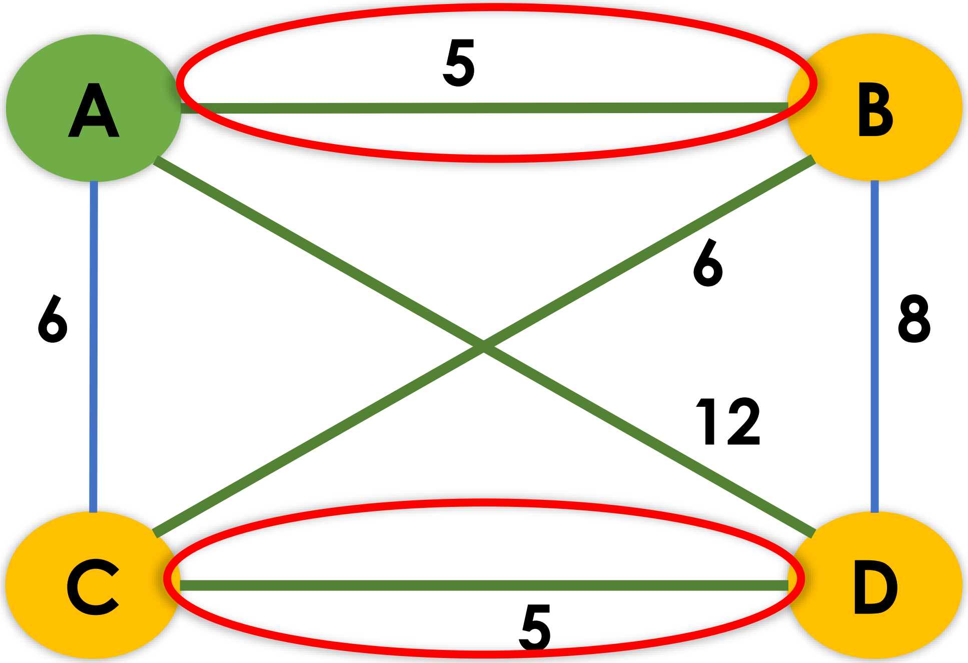
Let's improve on this solution

2-Opt Heuristic



A → B → C → D → A L: 28

Intuition: Examine pairs of edges, remove them, repair the solution, and see if it's better.

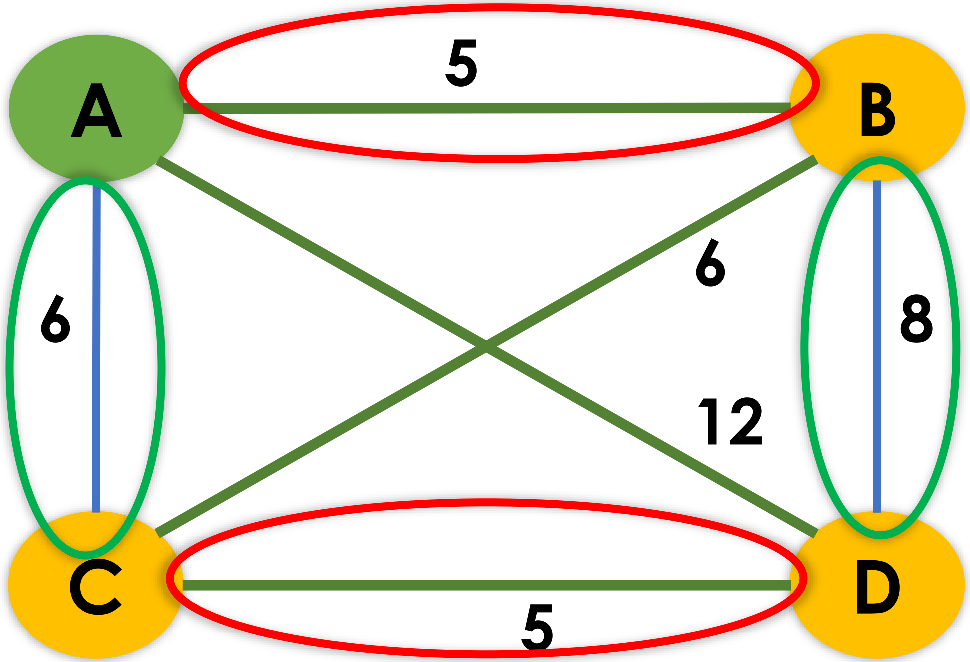


2-Opt Heuristic

A → B → C → D → A L: 28

For example, let's remove AB and CD

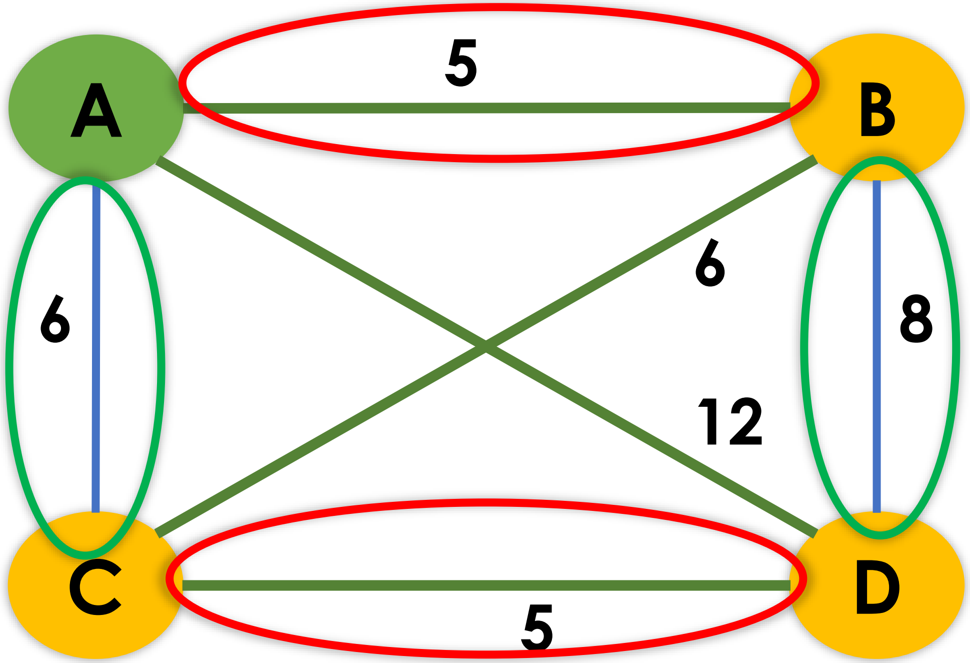
2-Opt Heuristic



A → B → C → D → A L: 28

For example, let's remove AB and CD, then repair the solution by adding AC and BD

2-Opt Heuristic

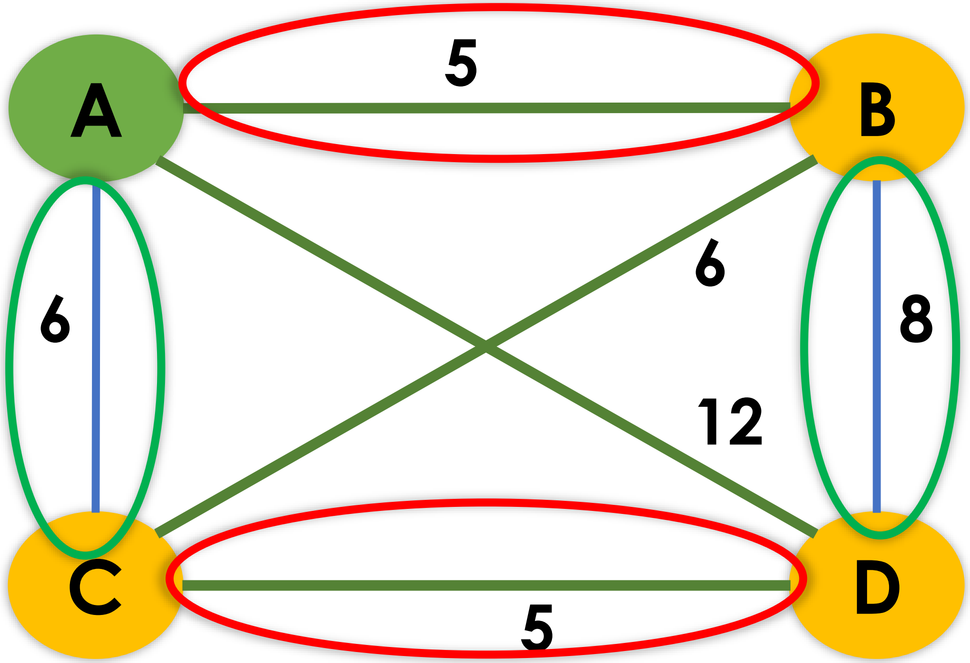


A → B → C → D → A L: 28

A → C → B → D → A L: 31

For example, let's remove AB and CD, then repair the solution by adding AC and BD

2-Opt Heuristic

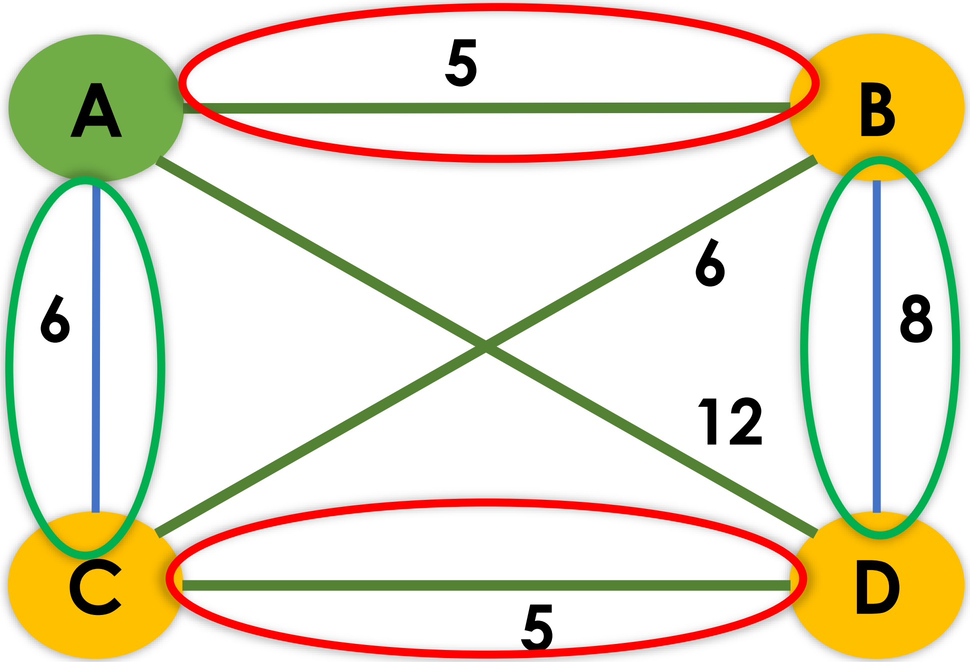


A → B → C → D → A L: 28

A → C → B → D → A L: 31

This is worse!

For example, let's remove AB and CD, then repair the solution by adding AC and BD

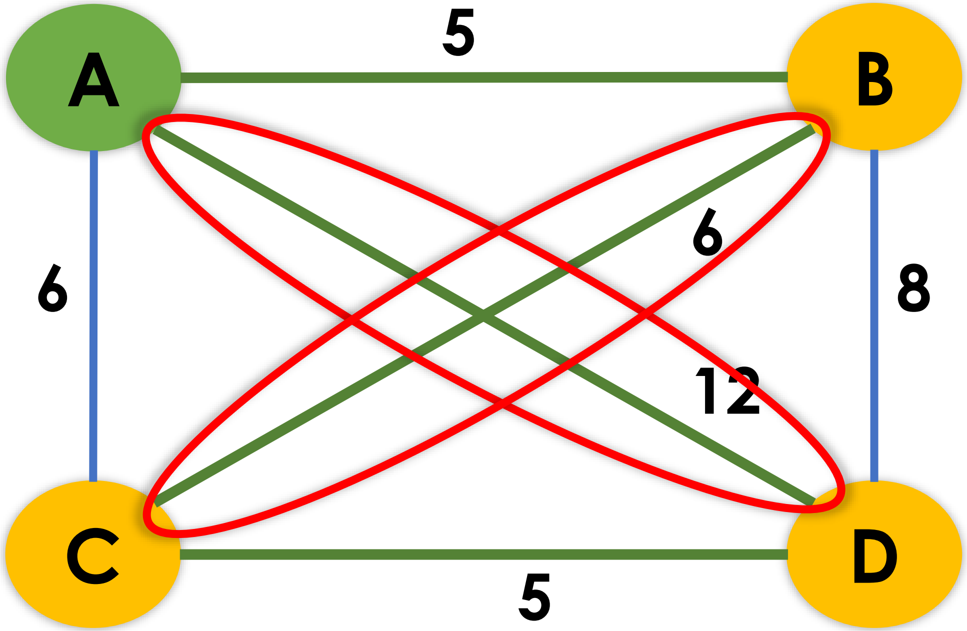


2-Opt Heuristic

A → B → C → D → A L: 28

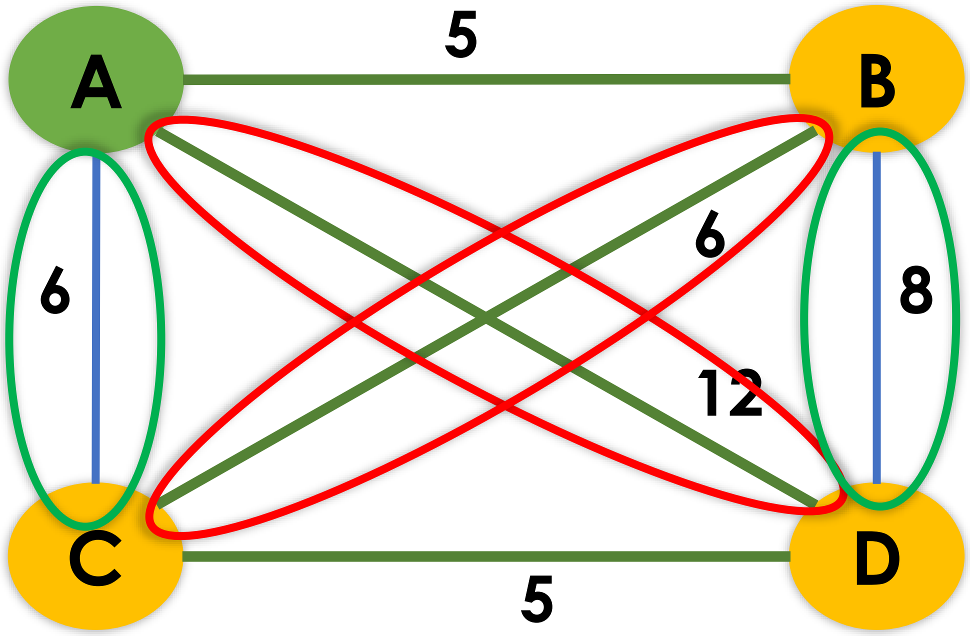
~~A → C → B → D → A L: 31~~

2-Opt Heuristic



A → B → C → D → A L: 28

Now, let's remove AB and CD

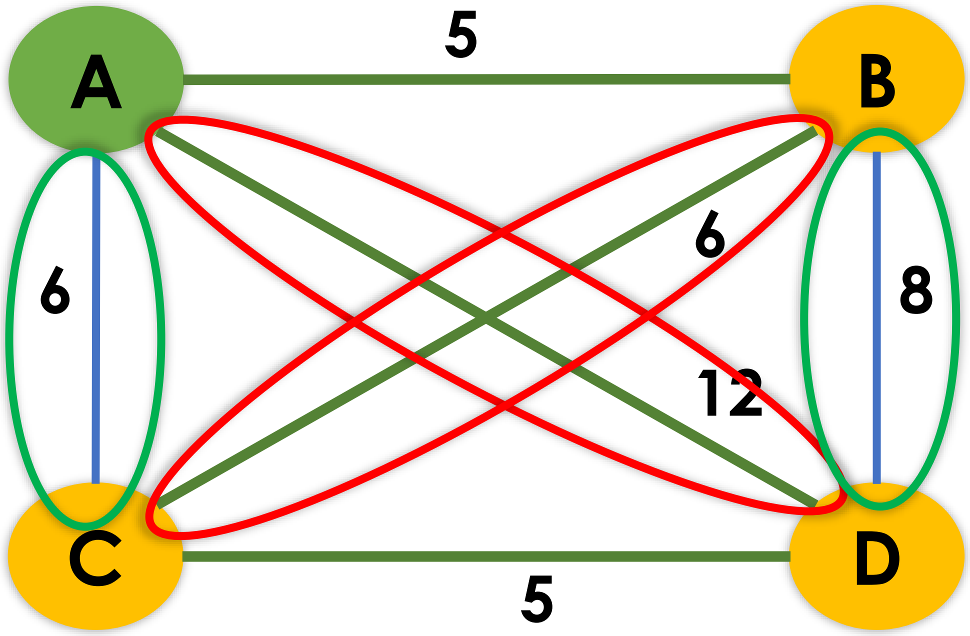


2-Opt Heuristic

A → B → C → D → A L: 28

Now, let's remove AB and CD, then repair the solution by adding AC and BD

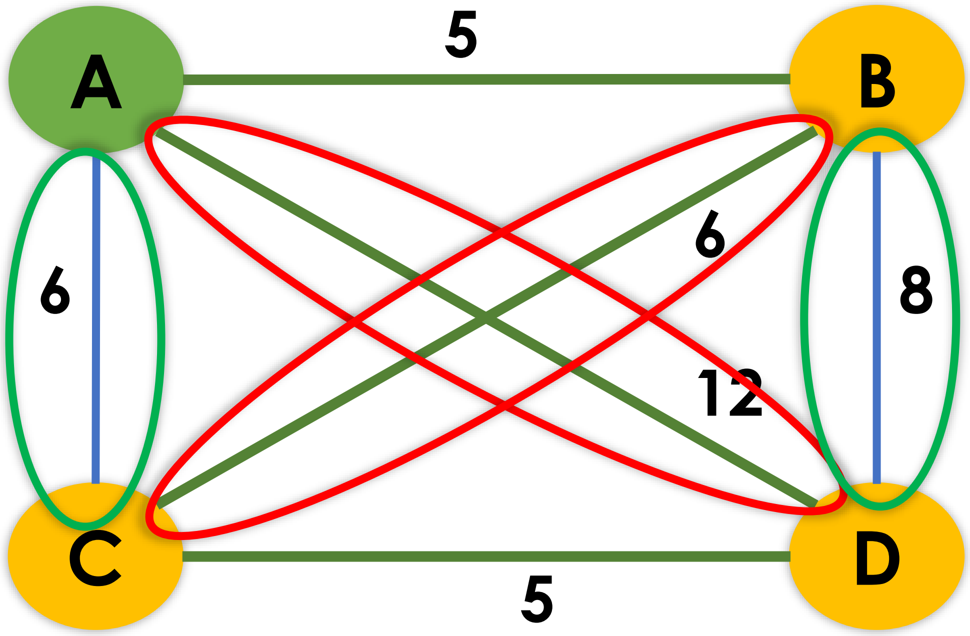
2-Opt Heuristic



A → B → C → D → A L: 28

A → B → D → C → A L: 24

Now, let's remove AB and CD, then repair the solution by adding AC and BD



2-Opt Heuristic

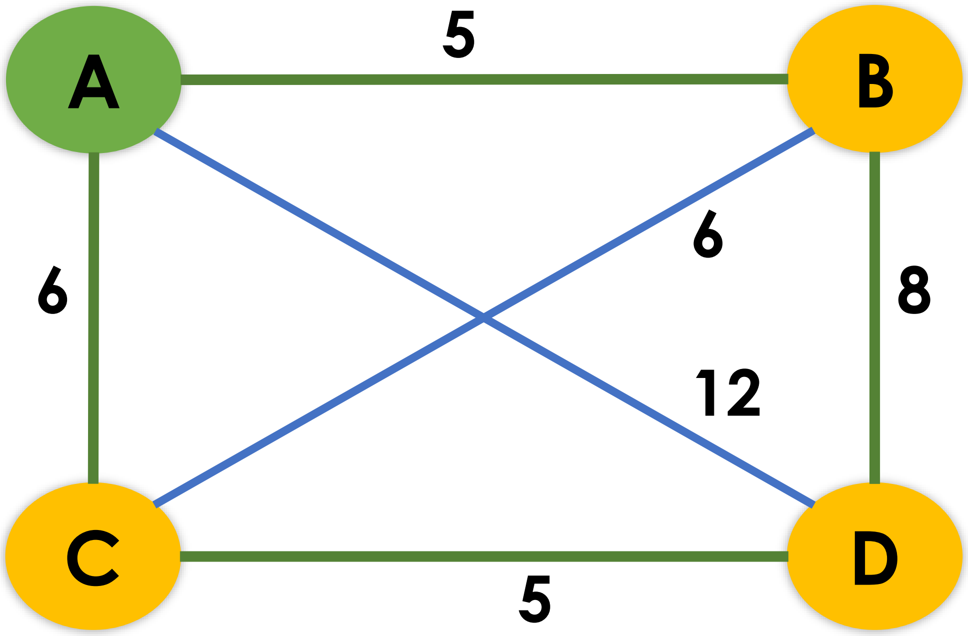
A → B → C → D → A L: 28

A → B → D → C → A L: 24

This is better!

Now, let's remove AB and CD, then repair the solution by adding AC and BD

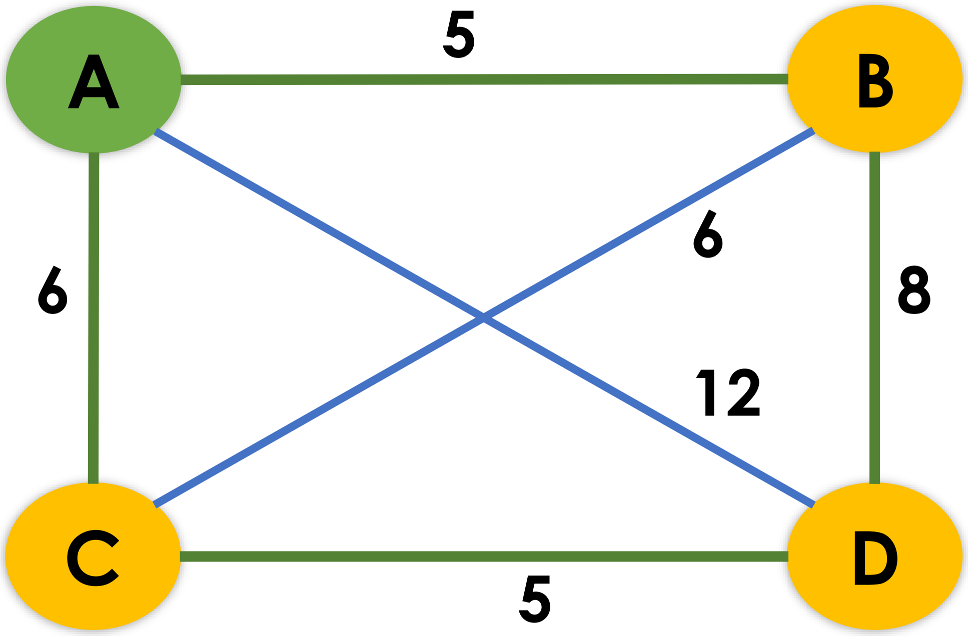
2-Opt Heuristic



~~A → B → C → D → A L: 28~~

A → B → D → C → A L: 24

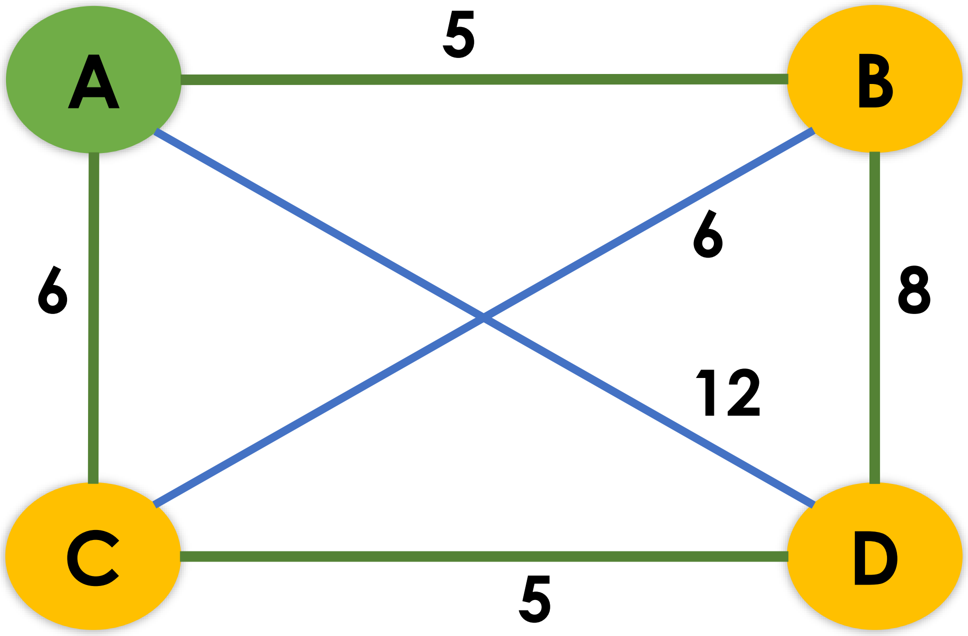
Replace our current solution with the better solution.



2-Opt Heuristic

A → B → D → C → A L: 24

Replace our current solution with the better solution.

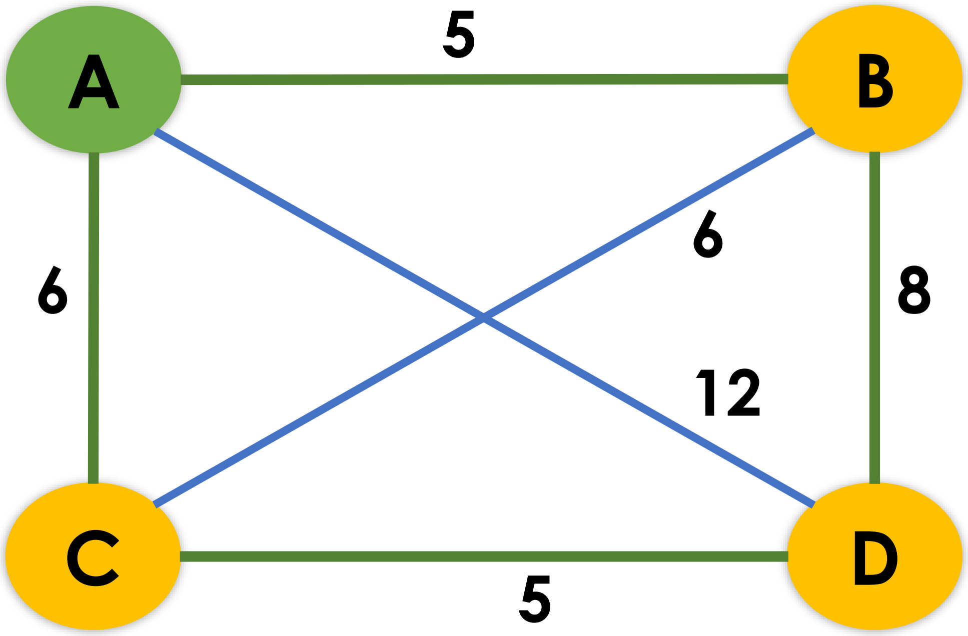


2-Opt Heuristic

A → B → D → C → A L: 24

Keep repeating for pairs of edges until no more improvements are found

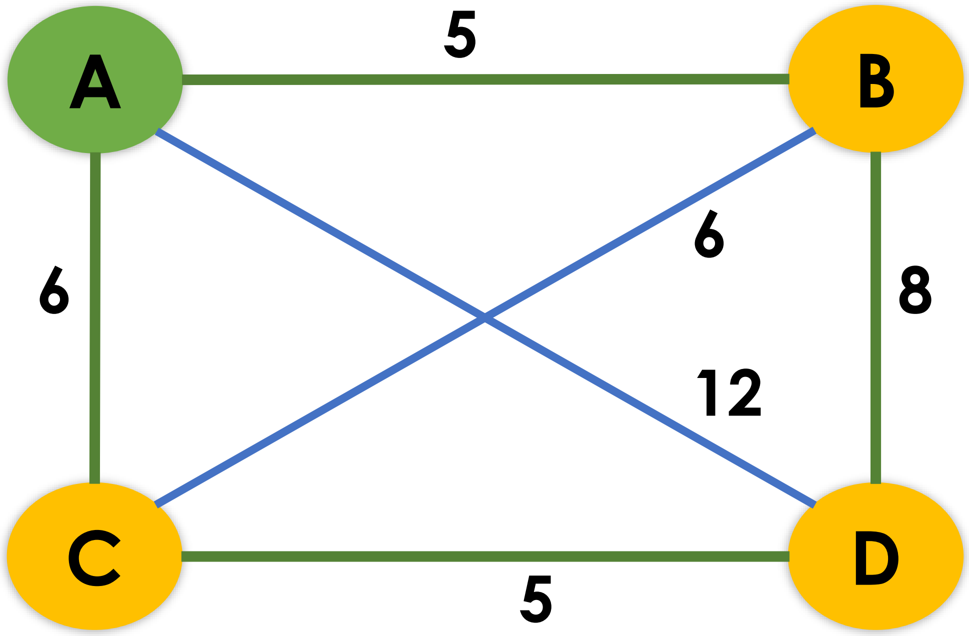
Greedy + 2-Opt Heuristic



A → B → D → C → A L: 24

Found optimal solution for this simple example

Greedy + 2-Opt Heuristic



A → B → D → C → A L: 24

Won't necessarily find the optimal, but it's fast - O(n^2) - and can be fairly effective

Heuristics and Approximation Algorithms

Constructions:

Nearest Neighbor (Greedy)

Christofides Algorithm

Iterative:

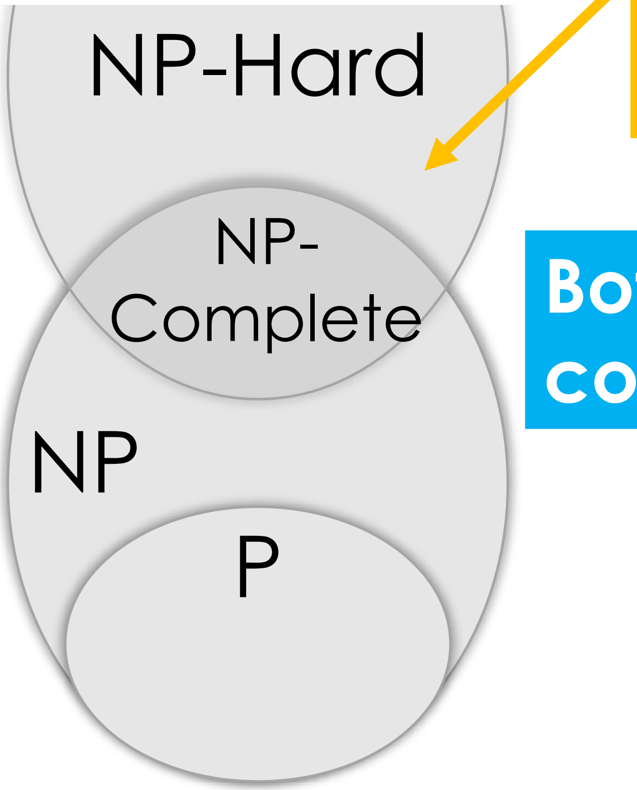
k-opt and Lin-Kernighan

genetic algorithms

MORE!

We encourage you to explore some of these on your own.

Complexity Theory



TSP "optimization": given n cities with one Hometown and all pairwise distances, plan a tour starting and ending at Hometown that visits every city exactly once and has minimum distance.

Bottom line: if the problem is provably "hard", consider revising the problem constraints