Lesson 8: Combinatorial Models and the Traveling Salesperson Problem

1 Combinatorial Models

Many optimization problems are naturally modeled by a combinatorial structure, such as a **graph**. For the next few weeks, we will be talking about several combinatorial optimization models. Combinatorial optimization problems are usually more general than the network problems we've seen before as they operate on a **graph** instead of a **network**. Recall that a network is a special type of graph.

2 Graph Terminology

Suppose $G = (V, E)$ is an *undirected* graph. (So far we have worked with directed graphs.) Recall that:

- V is the set of vertices or nodes
- E is the set of edges

$1 \to 2 \to 3 \to 4$

$1 - 2 - 3 - 4$

An undirected graph is different in that the edges can now be traversed in both directions.

$1 \xrightarrow{} 2$

x_{12}

Flow from 1 to 2

Directed

$1 - 2$

x_{12} flow from 1 to 2 or 2 to 1

Undirected

As a result, it's good to be consistent in naming the edges. For example, using our old style of naming, an edge connecting nodes 1 and 2 could be both $x_{1,2}$ and $x_{2,1}$. As a result, we will always name edges from lower index to higher index.

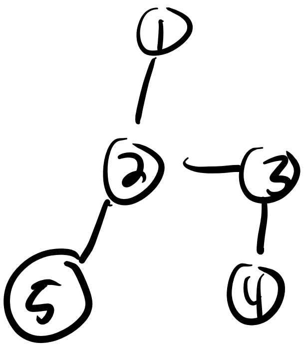
$1 - 2$

x_{12} Not x_{21}

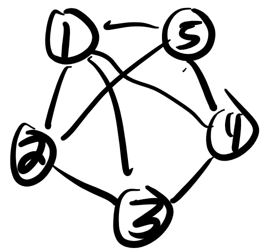
Connected : Path between every 2 nodes

> Graph G is **connected** if for *every* pair of vertices $a, \ b \in V$, there exists a **path** of edges in G connecting a and b.

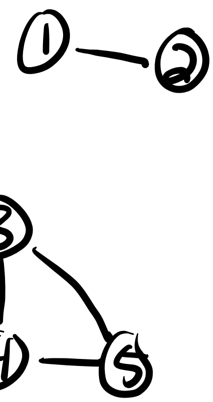
Example:



Connected



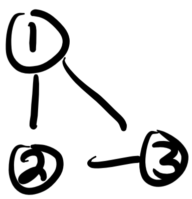
Connected



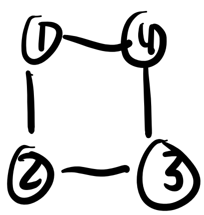
Not connected

> A **cycle** is a closed path of nodes meaning that the first and last node in the path is the same vertex.

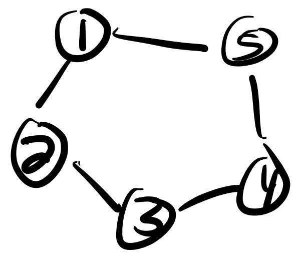
Example: Path same start and end point



Cycle 3



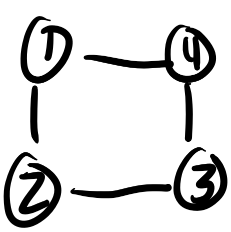
Cycle 4



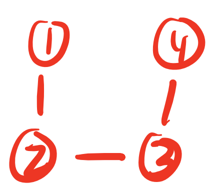
Cycle 5

> If G is a connected graph that contains a **cycle**, then the removal of a single edge from the cycle does not destroy the connectivity of G.

1. Illustrate a cycle and prove the fact that if G has a cycle, removing any edge from the cycle does not make the graph disconnected.



Remove (1,4)

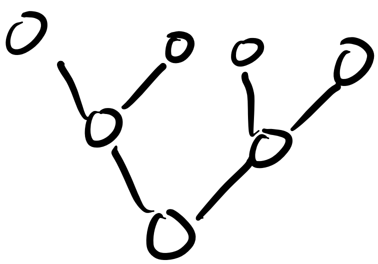


still connected

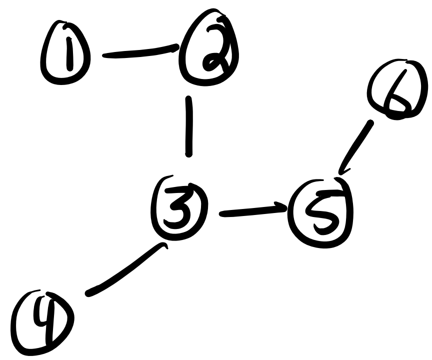
Same idea for other edges

A connected graph that contains no cycles is called a **tree**.

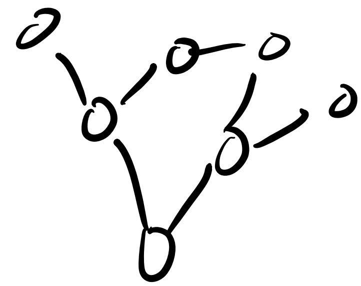
Example:



Tree



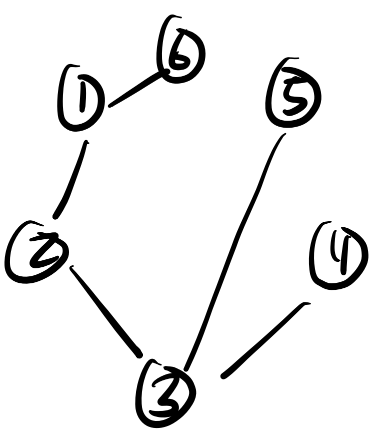
Tree



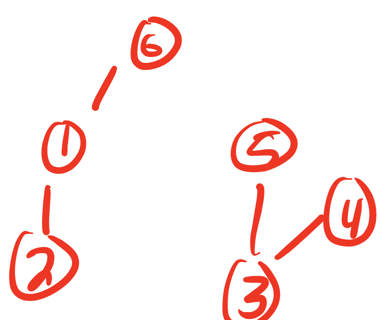
Not a tree

Trees are **minimally connected**: if we remove any edge from a tree, the resulting graph is disconnected.

2. Convince yourself of the previous fact by drawing a tree on 6 vertices. Are there any edges you can remove from the graph without losing connectivity?



Remove (2,3)



Forest (not a tree)

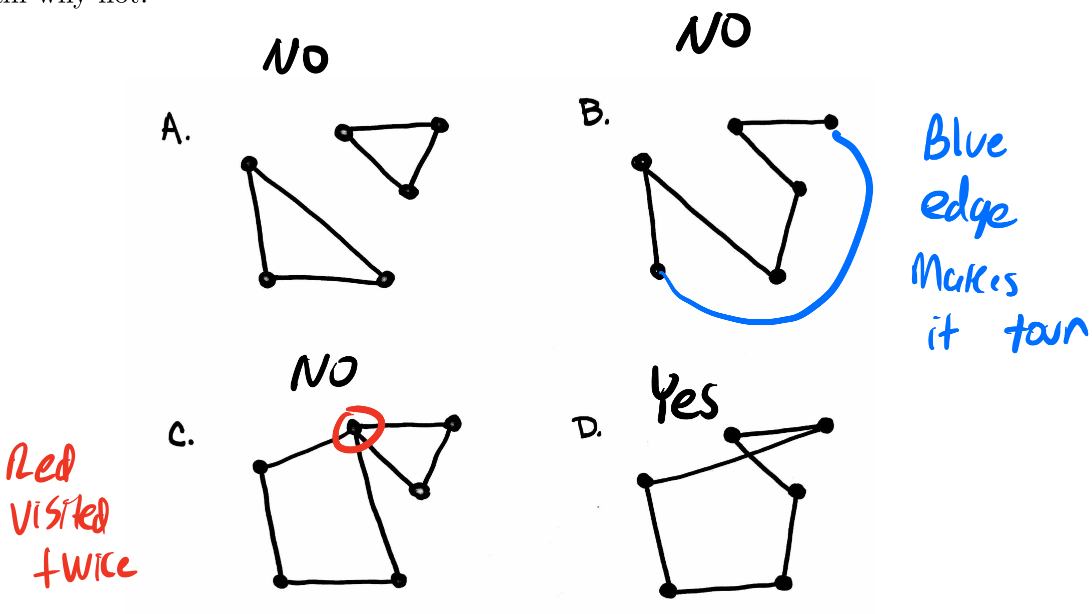
Trees are really important in tons of real world problems. For example, every time you make a Google search or use GoogleMaps an algorithm is called where one of the key parts is analyzing a huge tree. These trees are usually found via an algorithm (not math programming); but they are a structure that come up over and over again.

3 Tours and TSP

A **tour** is a closed route that visits every location exactly once.

In graph terminology, a **tour** is a single cycle (collection of edges) that visits each node exactly once. This is also known as a Hamiltonian Cycle or Hamiltonian Circuit.

Problem 1. For each graph below, does the set of edges represent a tour through the 6 nodes? If not, explain why not.



Given a graph $G = (V, E)$ with edge weights (representing costs or distances), the **Traveling Salesperson Problem (TSP)** seeks a *minimum cost tour of* G.

4 History of the TSP

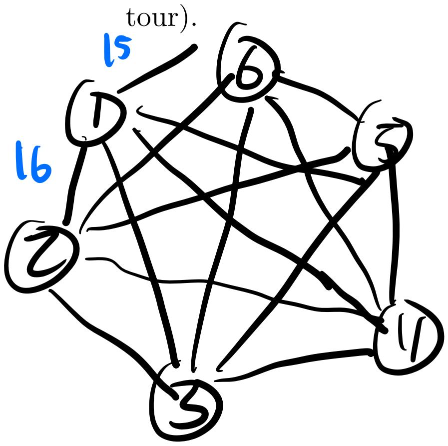
- TSP was first studied in the 1930s! The first IP formulation of TSP (which we're studying mostly in these notes) was in 1954.
- Nowadays TSP has been solved on graphs with as many as 50,000 nodes such as a historic tour of the US. http://www.math.uwaterloo.ca/tsp/

5 Visiting Graduate Schools: IP Formulation of TSP

Problem 2. A college student is interested in visiting as many graduate schools as possible. She reasons that a single visit to each school is appropriate, and she wants to return to her own campus only after visiting all the schools. It is conceivable that she visits the schools in any order, but she would like to minimize the amount of driving she has to do. If the distance between schools i and j is $d_{i,j}$ $(i < j)$, where the matrix D of distances is given below, in which order should she visit the schools? Note that school 1 is her current school.

$$D = \begin{bmatrix} & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 16 & 23 & 14 & 8 & 15 \\ 2 & - & 12 & 19 & 9 & 13 \\ 3 & - & - & 7 & 25 & 16 \\ 4 & - & - & - & 18 & 15 \\ 5 & - & - & - & - & 20 \end{bmatrix}$$

3. Draw the graph $G = (V, E)$ of the network below. Include node labels and edge costs. Highlight a collection of edges that form a tour of the graph (doesn't have to be the minimum distance tour).



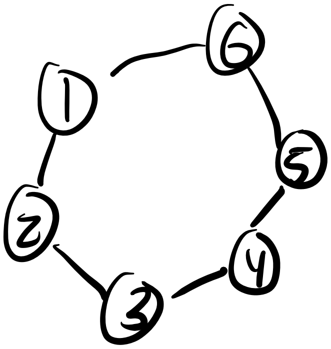
How many possible tours are there?

5! if you start at 1

6! to start anywhere

4. What type of solution do we want for this problem? What decision (variables) would we make in order to find such a solution?

$x_{ij} = 1$ if edge (i,j) is included and 0 otherwise



$x_{12} = x_{23} = \ldots = x_{56} = 1$

All other $x_{ij} = 0$

5. Define the Sets, Variables, and Parameters that are needed for the IP model.

Sets: let N be the set of nodes
let E be the set of edges

Params: let d_{ij} be the distance of edge (i,j) $\forall (i,j) \in E$

Variables: let $x_{ij} = 1$ if edge (i,j) is included and 0 otherwise $\forall (i,j) \in E$

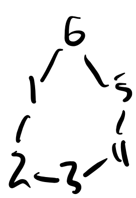
6. Write the objective function for this model.

$$\min \sum_{(i,j) \in E} d_{ij} x_{ij}$$

Not the Same as $\sum_{i \in N} \sum_{j \in N} d_{ij} x_{ij}$

7. There are two types of constraints that should be included in this model:

(a) A cycle on n nodes has exactly $\boxed{6}$ edges. Write this constraint in concrete and parameterized form.



$x_{12} + x_{13} + \dots + x_{56} = 6$] Pick only 6 of all

$$\sum_{(i,j) \in E} x_{ij} = |N|$$ ← Remember $|N|$ is the Cardinality of N = # of elements

(b) Every node is visited and departed from exactly one time. Thus, each node must be connected to exactly $\boxed{2}$ edges. Write this constraint in both concrete and parameterized form.

Node 1: $x_{12} + x_{13} + x_{14} + x_{15} + x_{16} = 2$

⋮

Node 3: $x_{13} + x_{23} + x_{34} + x_{35} + x_{36} = 2$

⋮

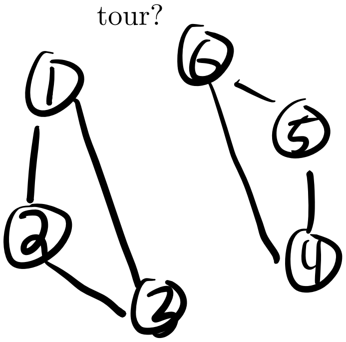
Node 6: $x_{16} + x_{26} + x_{36} + x_{46} + x_{56} = 2$

$$\sum_{(n,j) \in E} x_{nj} + \sum_{(i,n) \in E} x_{in} = 2 \quad \text{for all } n \in N$$

So, at this point, we have a model which given a set of nodes N and edges E:

- Minimizes the total cost of edges selected
- Enforces that exactly $|N|$ edges are part of the tour] 6 nodes → 6 edges
- Enforces that each node is connected to exactly 2 edges] every node has 2 edges

Using the graph of 6 nodes, can you think of a solution that satisfies these constraints but is not a tour?



6 edges ✓

every node 2 edges ✓

Tour ✗

8. Write a concrete constraint that prevents the graph above from being selected by the solver.

1-2-3 Triangle: $x_{12} + x_{13} + x_{23} \leq 3-1$

↗ forces at most 2 of these 3 edges to be picked

4-5-6 Triangle: $x_{45} + x_{46} + x_{56} \leq 3-1$

9. How many constraints of this type would be required to remove all cycles of size 3?

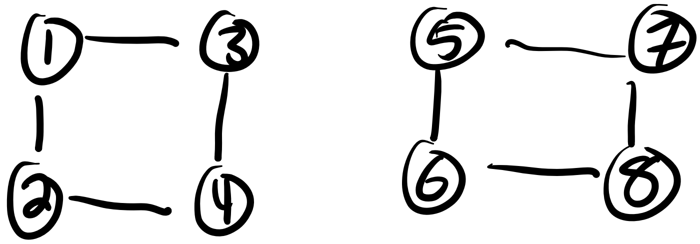
6 nodes
every 3 can be triangle] → $\binom{6}{3}$ constraints to remove all triangles

In general $\binom{n}{3}$ constraints to eliminate all triangles

> These type of constraints are called **cycle-elimination** or often **subtour-elimination** constraints. As we see above, there are way too many of these to all be included in the model. In practice, these constraints may be added iteratively to eliminate cycles in a solution returned by the solver. We will do something like this in the context of vehicle routing.

6 Subtour Elimination Constraints For More than Three Nodes

Now, we have seen an example of adding a subtour elimination constraint to eliminate a cycle of size three. Now, suppose our graph has 8 nodes and we obtain the following solution:



- 8 edges ✓
- every node 2 edges ✓
- tour ×

We want to eliminate one of the subtours on four nodes.

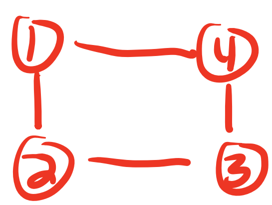
10. Write a constraint that eliminates this specific cycle on four nodes from the graph.

1-2-4-3 box. $x_{12} + x_{13} + x_{24} + x_{34} \leq 4 - 1$

↑ eliminate 1-2-4-3

11. What is the drawback of this constraint?

Use (1,4) and (2,3)



→ plug into 10

$1 + 0 + 0 + 1 \leq 3$

12. How can you modify this constraint to address this drawback?

Include all edges

$x_{12} + x_{13} + x_{24} + x_{34} + x_{14} + x_{23} \leq 3$

↑ stop any cycle of size 4 on nodes 1,2,3,4

13. Write this new constraint in parameterized form.

Idea: For all subsets of nodes, no cycle can exist on that set.

$$\sum_{\substack{(i,j) \in E: \\ i \in S \\ j \in S}} x_{ij} \leq |S| - 1 \quad \text{for all } S \subset N, \ |S| \geq 3$$

7 Putting it All Together: Parameterized TSP

Write the parameterized TSP model.

Sets

let N be the set of nodes
let E be the set of edges

Variables

let $x_{ij} = 1$ if edge (i,j) is included in tour $\forall (i,j) \in E$

Parameters

let d_{ij} be the distance of edge (i,j) $\forall (i,j) \in E$

Objective

$$\min \sum_{(i,j) \in E} d_{ij} x_{ij}$$

Constraints

$$\sum_{(i,j) \in E} x_{ij} = |N|$$

[A cycle has |N| nodes] 1 constraint

$$\sum_{(i,n) \in E} x_{in} + \sum_{(n,j) \in E} x_{nj} = 2 \quad \forall n \in N$$

(every node gets 2 edges) |N| Constraints

$$\sum_{\substack{(i,j) \in E: \\ i \in S, j \in S}} x_{ij} \leq |S| - 1 \quad \forall S \subset N \quad |S| \geq 3$$

(no cycles that aren't on all nodes)

$\approx 2^{|N|}$ constraints

$$x_{ij} \in \{0,1\} \quad \forall (i,j) \in E$$

$\underbrace{\text{Illustrating subtour elim}}$

$$\sum_{\substack{(i,j)\in E:\\ i\in S,\, j\in S}} x_{ij} \le |S| - 1 \quad \forall\, S \subset N \quad |S| \ge 3$$

$N = \{1,2,3,4,5,6,7,8\}$

$S_1 = \{1,2,4\} \rightarrow$ eliminate any cycle on $1,2,4$

$$\sum_{\substack{(i,j)\in E:\\ i\in S_1,\, j\in S_1}} x_{ij} \le |S_1| - 1$$

$$x_{12} + x_{14} + x_{24} \le 3 - 1$$

$S_2 = \{1,4,5,6\} \rightarrow$ eliminate any cycle on these

$$\sum_{\substack{(i,j)\in E:\\ i\in S_2,\, j\in S_2}} x_{ij} \le |S_2| - 1$$

$$x_{14} + x_{15} + x_{16} + x_{45} + x_{46} + x_{56} \le 4 - 1$$

Solve TSP

Formulate with no subtour constraints

Solve

Tour on all nodes

Done

No tour.

Add constraints to eliminate subtours