- Most of the problems we face in practice are NP-hard.
- MST vs. Steiner tree

\[
\text{MST} \quad \text{Steiner tree}
\]

- Shortest path vs. Travelling Salesman (TSP)
  
  (Visit all vertices in the graph
  find the shortest cycle)

- NP-hardness
  - It is hard to solve ALL problems of this type.
  - It may not be hard to solve any specific problem of this type.

- Clever search algorithms
  - Search algorithm
    - 3-SAT \ n \ variables
    - Enumerate \ 2^n \ possible assignments for these \ n \ variables
    - Not feasible when \ n \geq 50.
    - \( X_1 \lor X_2 \lor X_3 \)
      - \( X_1 = 0, \ X_2 = 0, \ X_3 = 0 \)

- Popular heuristics for 3-SAT: DPLL

  1. If a clause has only 1 literal, then it must be true
     \( X_1, \overline{X_2} \)
     Modify clauses when fixing value of variable
     \( X_1 \lor X_2 \lor X_3 \)
     - \( X_2 \lor X_3 \)
     - \( X_1 = 0 \)
     - \( X_1 = 1 \)
\[ X_1 \lor X_2 \lor X_3 \begin{cases} \neg x_i = 0 & x_i = 1 \\ \neg x_i = 1 & \end{cases} \]

1. Delete the clause

2. If a variable only appears in 1 form (\( X_i \) or \( \neg X_i \)), set it to the correct value.

- Rely on approximation algorithms.
- Find approximate solutions instead of optimal solutions.

- \text{e.g. TSP}

\[ \begin{array}{ccc}
6 & \text{12} & \text{13} \\
\text{11} & & \text{8} \\
\text{10} & \end{array} \]

- Say an algorithm is a \( p \)-approximation, if it can always output a solution whose cost is no more than \( p \cdot \text{OPT} \)
  (\text{OPT: cost of optimal solution})

- Simple approximation algorithm for TSP (with triangle inequality)
  \[ W_{u,v} \leq W_{u,w} + W_{w,v} \]

- 1. Find a MST
- 2. Visit vertices in DFS order (preorder) of the MST.
- Claim: This is a 2-approximation algorithm.
- Proof: \[ \text{cost of MST} \leq \text{cost of TSP} \]
  \[ \begin{align*}
  \text{cost of alg} & \leq 2 \times \text{cost of MST} \\
  & \leq \text{cost of blue cycle} + \text{cost of red cycle}
  \end{align*} \]
- FPTAS (fully polynomial-time approximation scheme)
  there is an algorithm that can find a \((1 + \varepsilon)\) approximate solution in \(\text{poly}(n, \frac{1}{\varepsilon})\) time.
- Knapsack
- average case analysis
  - input is generated by nature, not by an adversary.
  - input is generated by some distribution, the algorithm should work with high probability under this distribution.
- average case \neq randomized algorithm.
- example: community problem (simple ver)

people in same community connected w.p. \(P\)

\[ p \quad \text{different} \quad q \]

\(P > q\)

BALANCED-SEPARATOR: find a cut that divide people into two equal-size groups with fewer edges.
equal-size groups with fewer edges.

NP-hard.

- For this particular distribution, communities can be found by applying "spectral clustering" (PCA on the adjacency matrix).

- Use hardness for crypto.