What to do with $NP$-hard optimization problems:

- efficient exhaustive search (backtracking, branch & bound) - exponential time.
- heuristics
  - local search - start with some solution and try to improve it via small "local" changes. "Simulated annealing" overcomes local optima.
  - might be no guarantee on run-time or on quality of solution.

- approximation algs. - today's topic
  - poly time and a guarantee on the quality of the solution
    e.g. for min problem, might guarantee a solution $\leq 2 \cdot \text{min}$. 
Example

Vertex Cover

\[G = (V, E)\]

want set \( C \subseteq V \) s.t.

\( \forall (u, v) \in E, u \in C \text{ or } v \in C \text{ (or both)} \)

minimize \(|C|\)

Greedy Algorithm 1

\[C \leftarrow \emptyset\]

repeat

\[C \leftarrow C \cup \{\text{vertex of max degree}\}\]

remove covered edges

until no edges remain

Example

Note: this is a poly-time algorithm
Greedy algorithm 2

\[ C = \emptyset \quad F = E \]  // F is uncovered edges

\textbf{while} \quad F \neq \emptyset

\textbf{pick} \quad e = (u, v) \quad \textbf{from} \quad F

\textbf{add} \quad u \quad \textbf{and} \quad v \quad \textbf{to} \quad C

\textbf{remove} \quad \textbf{edges} \quad \textbf{incident} \quad \textbf{to} \quad u \quad \textbf{from} \quad F

\textbf{end}

Ex. Find a graph where Alg. 2 is better than Alg. 1

\textbf{[solution is on Piazza]}

Note that the alg. takes poly time.

\textbf{Let} \quad C = \text{vertex cover found by alg. 2}

\textbf{Claim} \quad |C| \leq 2 \cdot |C^\text{opt}|
Pf. The set of edges you pick forms a matching \( M \) (no 2 edges are incident)
\[
\begin{align*}
|C| &= 2|M| \\
\text{Any vertex cover must have at least one vertex from each edge in a matching,} \\
1M| &\leq |C_{opt}| \\
\text{Thus } |C| &\leq 2|C_{opt}|.
\end{align*}
\]
Alg 2 has approx. factor 2.

**FACT:** Alg 1 has approx. factor \( \Theta(\log n) \)

Recall that Vertex Cover and Independent Set are closely related. However,

**FACT:** Ind. Set has no good approximation algorithm unless P=NP

(CS 466 covers these topics)
Example: Travelling Salesman Problem (TSP) for points in the plane w/ Euclidean distances.

- Complete graph

- Triangle inequality: \( w(a, c) \leq w(a, b) + w(b, c) \)

Approx. Alg:
- Compute MST (in black)
- Take a tour by walking around it. (in blue)
  (we visit every vertex but maybe more than once)
- Take shortcuts to avoid revisiting (in red)

Note: Δ inequality \( \Rightarrow \) shortcuts shorter

We can do this in poly. time.
let \( l \) be length of resulting tour
\[ l_{\text{TSP}} = " \] of min TSP tour

\textbf{Claim} \( l \leq 2l_{\text{TSP}} \) \text{ [Note } l_{\text{TSP}} \leq l]\)

So in poly time we find a tour within \( 2 \times \) optimum.
we call this approximation factor \( 2 \)

\textbf{PF of Claim} \( l_{\text{MST}} = \) length of MST

\[ l_{\text{MST}} \leq l_{\text{TSP}} \text{ because deleting one edge of TSP gives a spanning tree} \]

\[ l \leq 2l_{\text{MST}} \text{ because we use every MST edge twice, then take short cuts. (use } \triangle \text{ineq.}) \]

Putting these together:

\[ l \leq 2l_{\text{TSP}} \]

We say this alg. has approximation factor 2.

\textbf{FACT:} the factor of 2 can be improved for this problem. For any \( \varepsilon > 0 \) there is an algorithm that finds a tour of length \( \leq (1 + \varepsilon)l_{\text{TSP}}. \) But as \( \varepsilon \to 0, \) the run time becomes exponential.