NP-completeness

- Many many problems are NP-complete
- If you solve one of them efficiently, you solve all of them efficiently
- We don’t know how to solve any of them efficiently
Approximation Algorithms

• So it’s unlikely we’ll solve one of these soon :(  
• Instead of finding the best solution, we’ll find a solution that is close :)}
Traveling Salesman

**Given:** n cities with distances

**Goal:** Compute shortest tour to visit them
Traveling Salesman

**Given**: n cities with distances
**Goal**: Compute shortest tour to visit them

**Metric TSP**: distances satisfy triangle inequality:
\[
\text{distance}(a,c) \leq \text{distance}(a,b) + \text{distance}(b,c)
\]

**Idea**: use MST!
**Prove**: tour within factor 2 of best possible
MST tour: Show that it is within factor 2!
MST tour: Take the Euler tour of tree.
Claim: Every tour costs at least as much as MST.
Pf: Every tour contains a spanning tree

Claim: Euler tour costs at most 2 MST.
Pf: Can carry out Euler tour using each edge at most 2 times.
Vertex Cover

Find smallest set of vertices touching every edge
Vertex Cover

Find smallest set of vertices touching every edge
Vertex Cover

Find smallest set of vertices touching every edge
Vertex Cover

Find smallest set of vertices touching every edge
Vertex Cover

Find smallest set of vertices touching every edge
Vertex Cover

Find smallest set of vertices touching every edge

Vertex Cover size 5
Greedy algorithms?

- Include vertex that covers most new edges?
Algorithm: Pick vertex that covers most new edges

Each vertex on top row has one edge into each of the groups below.
Algorithm: Pick vertex that covers most new edges

Each vertex on top row has one edge into each of the groups below.
Algorithm: Pick vertex that covers most new edges

Each vertex on top row has one edge into each of the groups below.
Algorithm: Pick vertex that covers most new edges

Each vertex on top row has one edge into each of the groups below.
Algorithm: Pick vertex that covers most new edges

Each vertex on top row has one edge into each of the groups below.
Algorithm: Pick vertex that covers most new edges

Each vertex on top row has one edge into each of the groups below.
Algorithm: Pick vertex that covers most new edges

Each vertex on top row has one edge into each of the groups below.
Algorithm: Pick vertex that covers most new edges

Each vertex on top row has one edge into each of the groups below.
Algorithm: Pick vertex that covers most new edges

Each vertex on top row has one edge into each of the groups below.
Algorithm: Pick vertex that covers most new edges

Each vertex on top row has one edge into each of the groups below.
Algorithm: Pick vertex that covers most new edges

Each vertex on top row has one edge into each of the groups below.
Algorithm: Pick vertex that covers most new edges

Each vertex on top row has one edge into each of the groups below.
Algorithm: Pick vertex that covers most new edges

Each vertex on top row has one edge into each of the groups below.
Algorithm: Pick vertex that covers most new edges

Each vertex on top row has one edge into each of the groups below.

Vertex Cover size 20
Algorithm: Pick vertex that covers most new edges

Each vertex on top row has one edge into each of the groups below.

Optimal Vertex Cover size 8
Greedy Rule: Pick vertex that covers the most edges
Could pick $B_1, \ldots, B_n$: $n \log(n)$ vertices

$n$ vertices each
vertex has at
most one edge
into $B_i$

$B_n$ $B_{n-1}$
degree $n$

$n/i$ vertices of degree $i$

$B_1$
Greedy Rule:
Pick uncovered edge, add its end points

Find smallest set of vertices touching every edge
Greedy Rule:
Pick uncovered edge, add its end points

Find smallest set of vertices touching every edge
Greedy Rule:
Pick uncovered edge, add its end points

Find smallest set of vertices touching every edge

Vertex Cover size 6
Each vertex on top row has one edge into each of the groups below.

Greedy Rule:
Pick uncovered edge, add its end points
Each vertex on top row has one edge into each of the groups below.

Greedy Rule:
Pick uncovered edge, add its end points
Greedy Rule:
Pick uncovered edge, add its end points

Each vertex on top row has one edge into each of the groups below.
Each vertex on top row has one edge into each of the groups below.

Greedy Rule:
Pick uncovered edge, add its end points
Each vertex on top row has one edge into each of the groups below.

Greedy Rule:
Pick uncovered edge, add its end points
Each vertex on top row has one edge into each of the groups below.

Greedy Rule:
Pick uncovered edge, add its end points

Vertex Cover size 16
Theorem: Size of greedy vertex cover is at most twice as big as size of optimal cover

Proof: Consider k edges picked.
Theorem: Size of greedy vertex cover is at most twice as big as size of optimal cover

Proof: Consider $k$ edges picked.

Edges do not touch: every cover must pick one vertex per edge! Thus every vertex cover has $k$ vertices.
Set Cover

Find smallest collection of sets containing every point
Set Cover

Find smallest collection of sets containing every point

Set Cover size 4
Find smallest collection of sets containing every point

Greedy Set Cover: Pick the set that maximizes # new elements covered
Greedy Set Cover: Pick the set that maximizes # new elements covered

Find smallest collection of sets containing every point
Greedy Set Cover: Pick the set that maximizes # new elements covered

Find smallest collection of sets containing every point
Greedy Set Cover: Pick the set that maximizes \# new elements covered

Find smallest collection of sets containing every point
Greedy Set Cover: Pick the set that maximizes # new elements covered

Find smallest collection of sets containing every point
Theorem: Greedy finds best cover upto a factor of $\ln(n)$. 

Greedy Set Cover: Pick the set that maximizes the number of new elements covered.
Greedy Set Cover: Pick the set that maximizes # new elements covered
Greedy Set Cover: Pick the set that maximizes # new elements covered
Greedy Set Cover: Pick the set that maximizes # new elements covered
Greedy Set Cover: Pick the set that maximizes # new elements covered
Greedy Set Cover: Pick the set that maximizes \# new elements covered
Greedy Set Cover: Pick the set that maximizes # new elements covered

Greedy solution: 5 sets
Greedy Set Cover: Pick the set that maximizes # new elements covered

optimal solution: 2 sets

greedy solution: 5 sets
Greedy Set Cover: Pick the set that maximizes # new elements covered

optimal solution: 2 sets

greedy solution: \( \log(n) \) sets
Greedy Set Cover: Pick the set that maximizes # new elements covered

**Theorem:** If the best solution has k sets, greedy finds at most $k \ln(n)$ sets.

**Pf:**
Suppose there is a set cover of size $k$. 
Greedy Set Cover: Pick the set that maximizes # new elements covered

**Theorem:** If the best solution has k sets, greedy finds at most k \( k \ln(n) \) sets.

**Pf:** Suppose there is a set cover of size k.

There is set that covers 1/k fraction of remaining elements, since there are k sets that cover all remaining elements. So in each step, algorithm will cover 1/k fraction of remaining elements.
Greedy Set Cover: Pick the set that maximizes # new elements covered

**Theorem:** If the best solution has k sets, greedy finds at most k ln(n) sets.

**Pf:** Suppose there is a set cover of size k.

There is set that covers 1/k fraction of remaining elements, since there are k sets that cover all remaining elements. So in each step, algorithm will cover 1/k fraction of remaining elements.

#elements uncovered after t steps ≤ n(1-1/k)^t < ne^{-t/k}. So after t = k ln (n) steps, number of uncovered elements < 1.