CSE 421

Vertex Cover / Set Cover

Yin Tat Lee

Approximation Algorithms

How to deal with NP-complete Problem

Many of the important problems are NP-complete.

SAT, Set Cover, Graph Coloring, TSP, Max IND Set, Vertex Cover, …

So, we cannot find optimum solutions in polynomial time.

What to do instead?

- Find optimum solution of special cases (e.g., random inputs)

- Find near optimum solution in the worst case

Approximation Algorithm

We call an algorithm has approximation ratio $\alpha(n)$ if

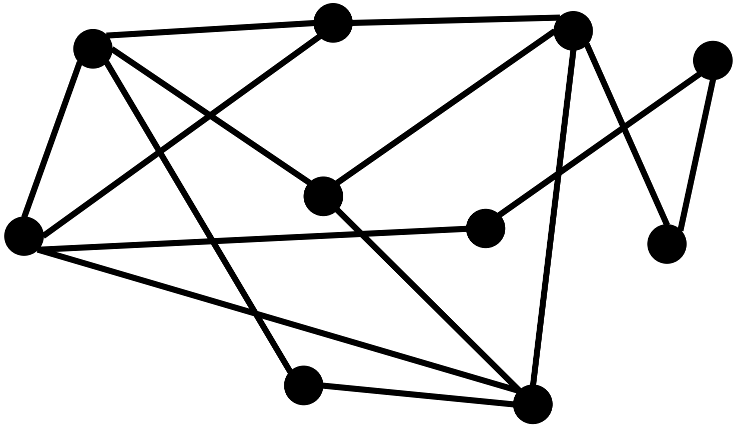
$$\frac{\text{Cost of computed solution}}{\text{Cost of the optimum}} \leq \alpha(n)$$

for any input of length n. (worst case)

Goal: For each NP-hard problem find an poly-time approximation algorithm with the best possible approximation ratio.

Vertex Cover

Given a graph $G = (V, E)$, Find smallest set of vertices touching every edge



Greedy Algorithm?

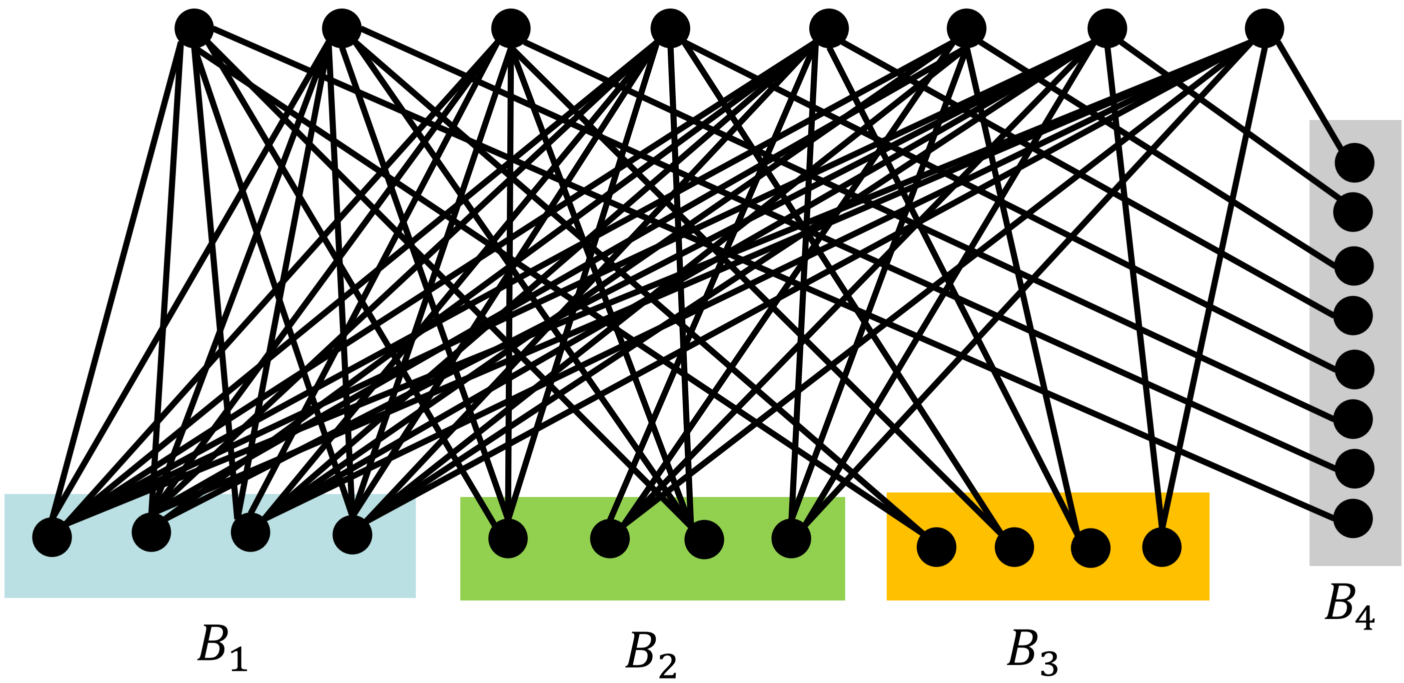
Greedy algorithms are typically used in practice to find a (good) solution to NP-hard problems

Strategy (1): Iteratively, include a vertex that covers most new edges

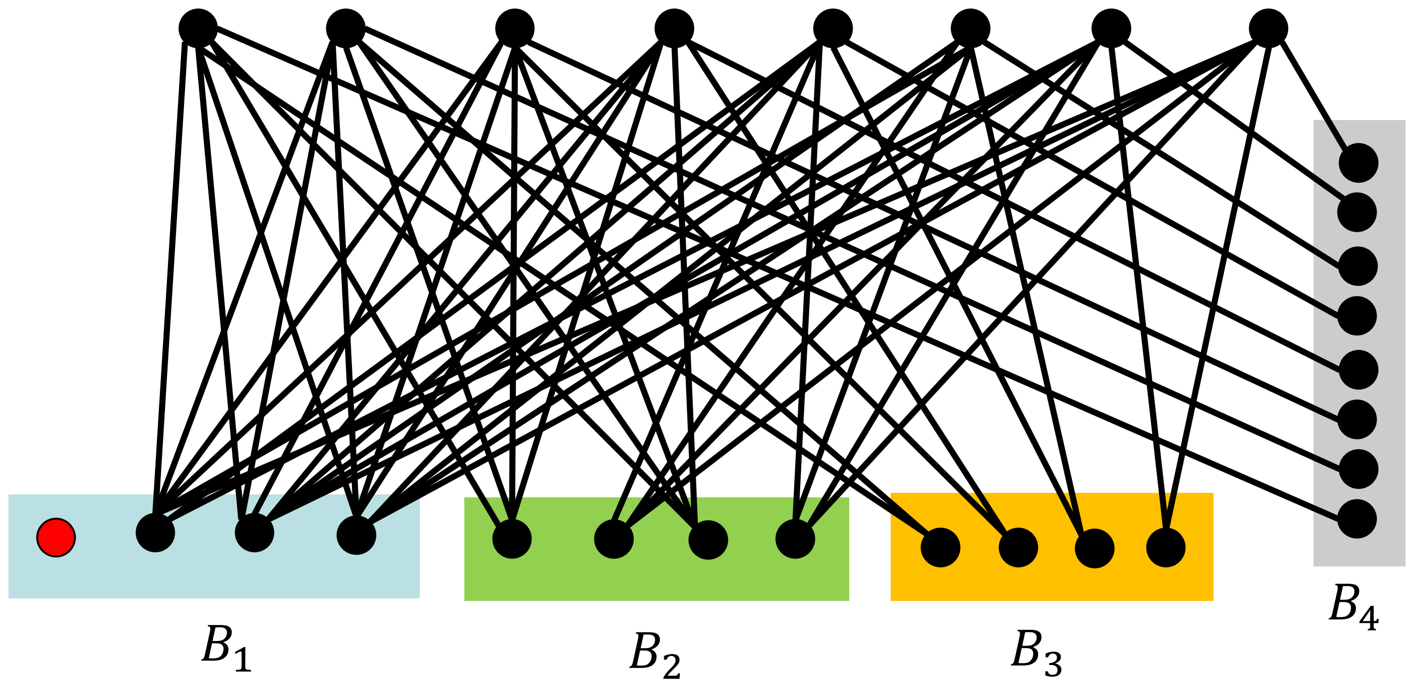
Q:Does this give an optimum solution?

A: No,

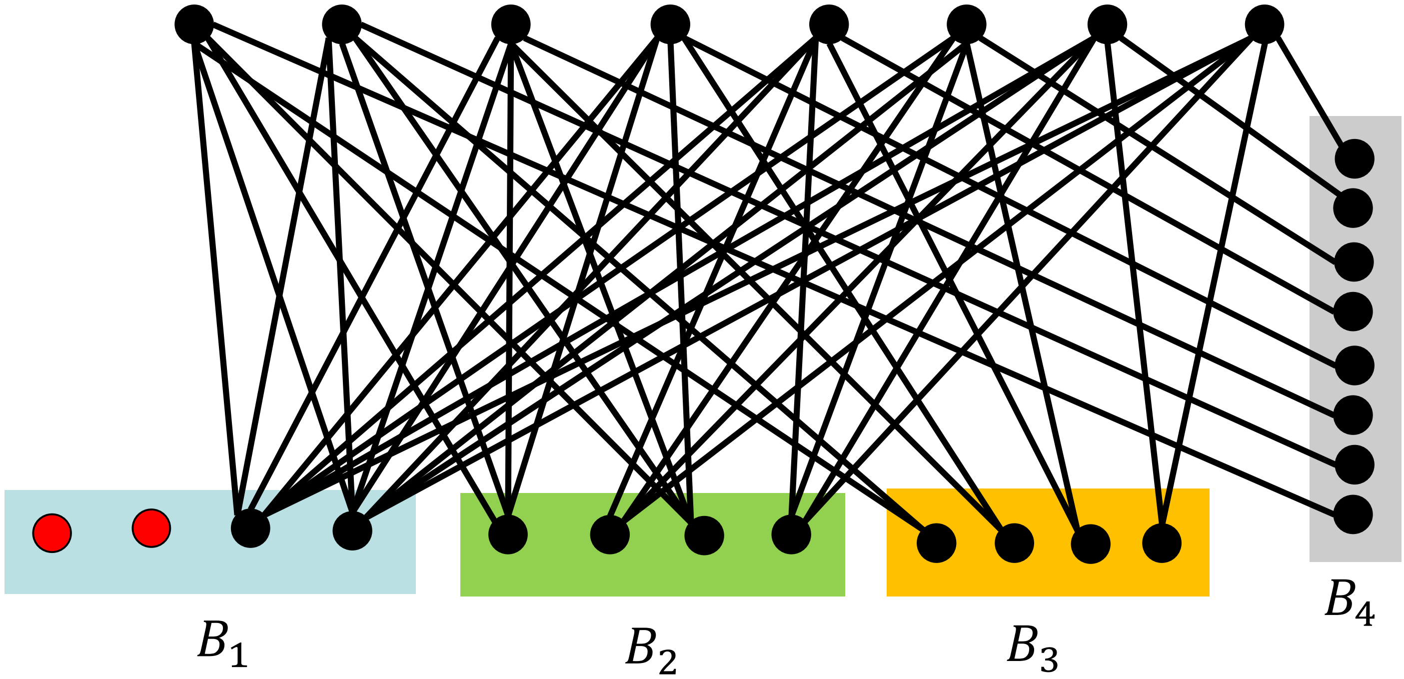
Greedy (1): Pick vertex that covers the most



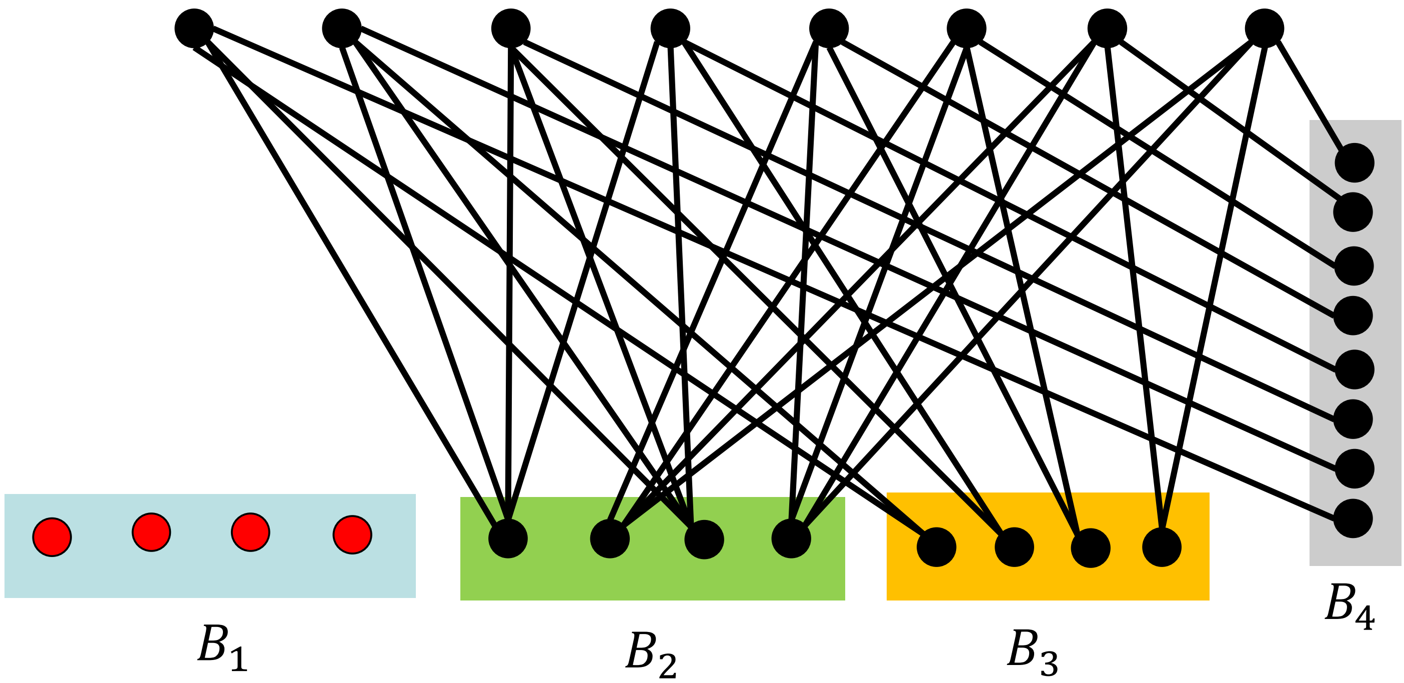
Greedy (1): Pick vertex that covers the most



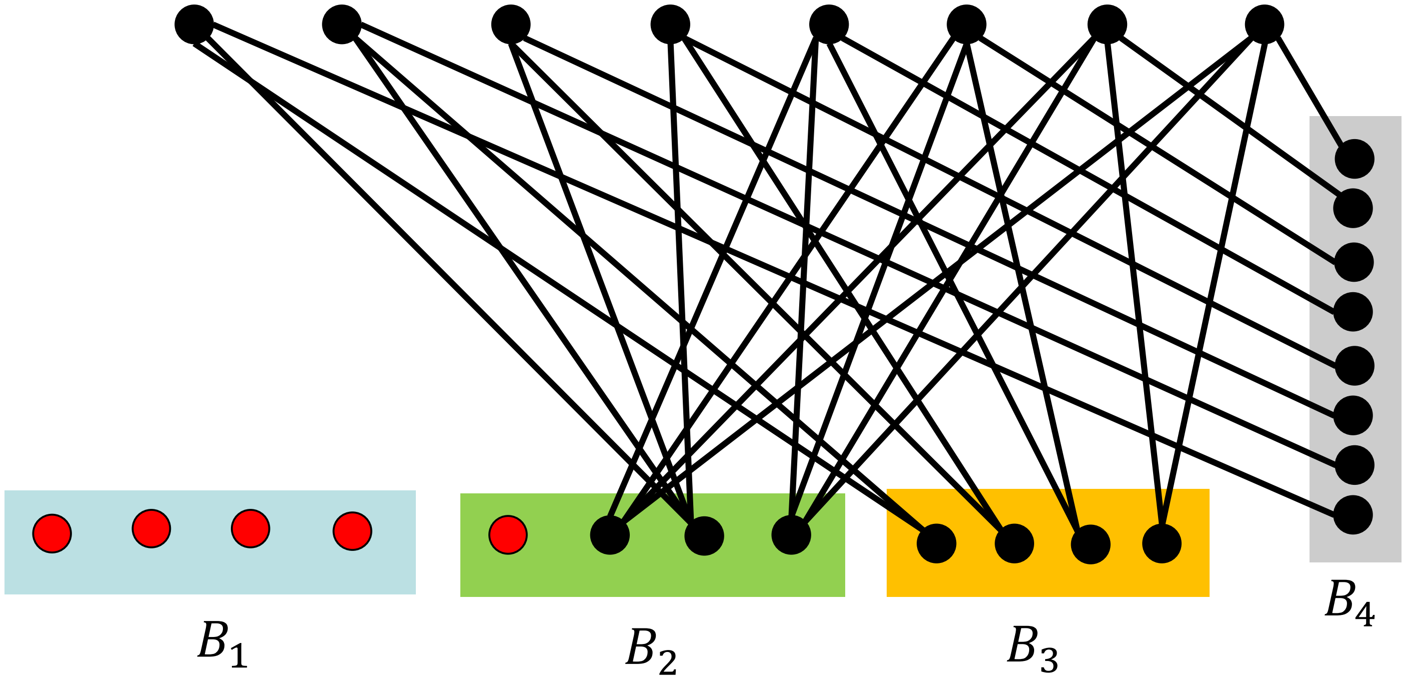
Greedy (1): Pick vertex that covers the most



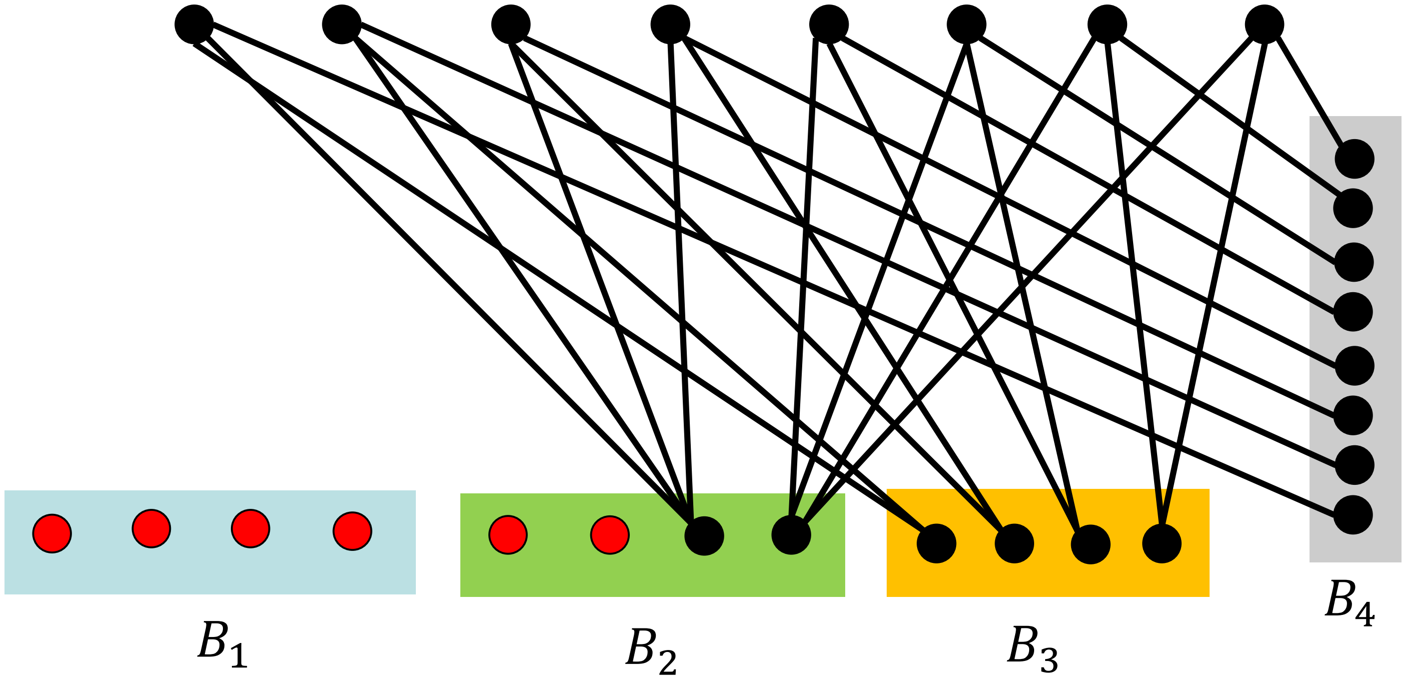
Greedy (1): Pick vertex that covers the most



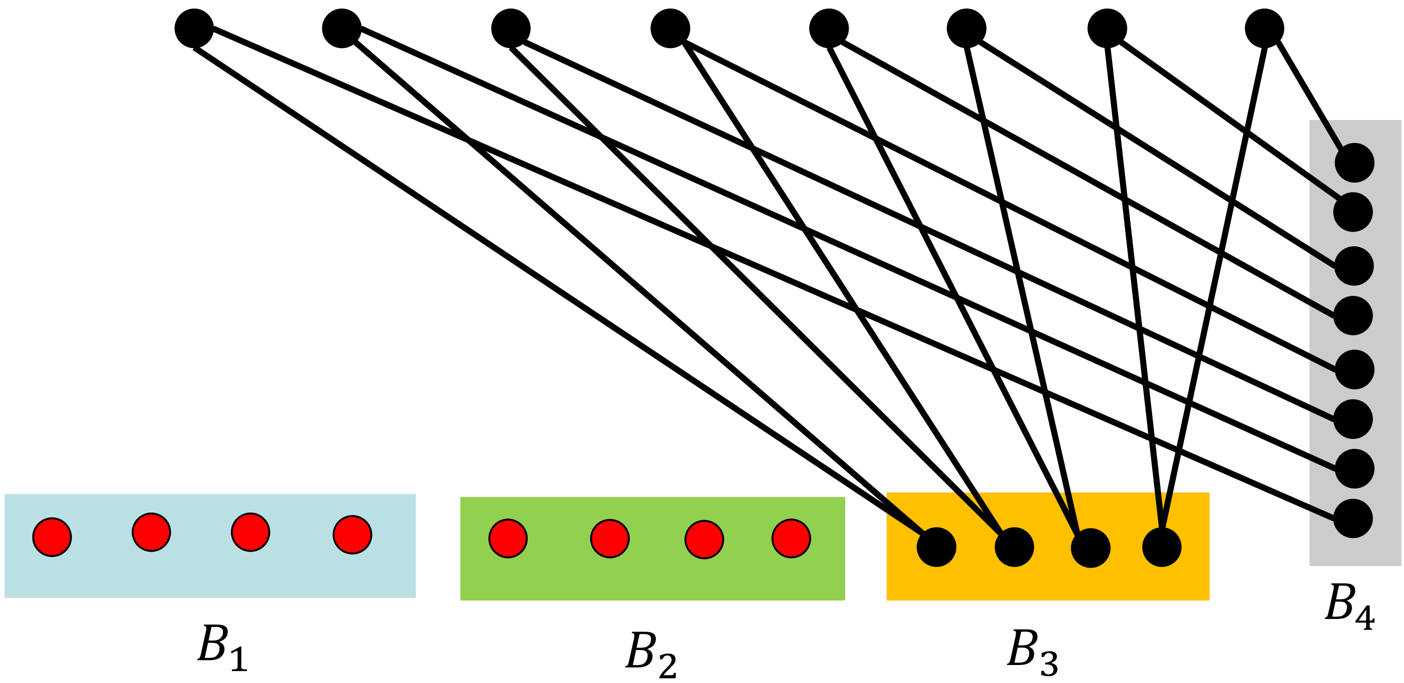
Greedy (1): Pick vertex that covers the most



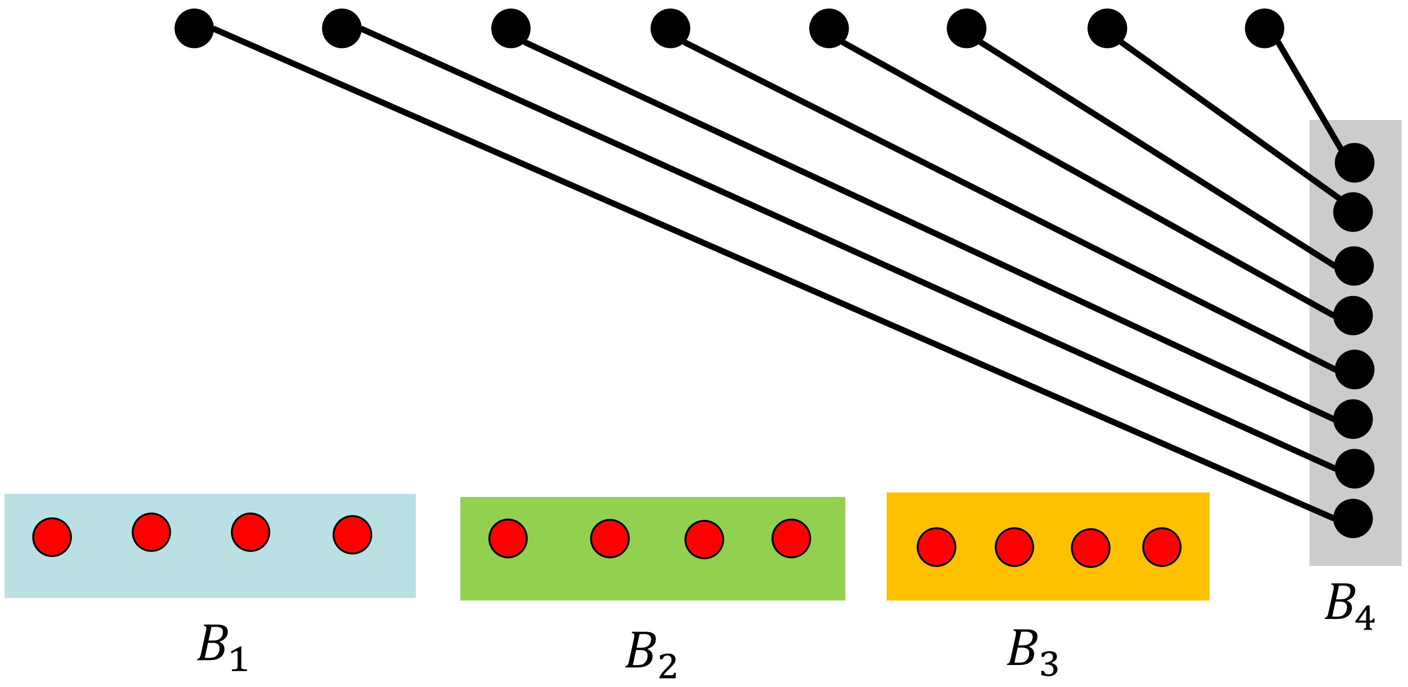
Greedy (1): Pick vertex that covers the most



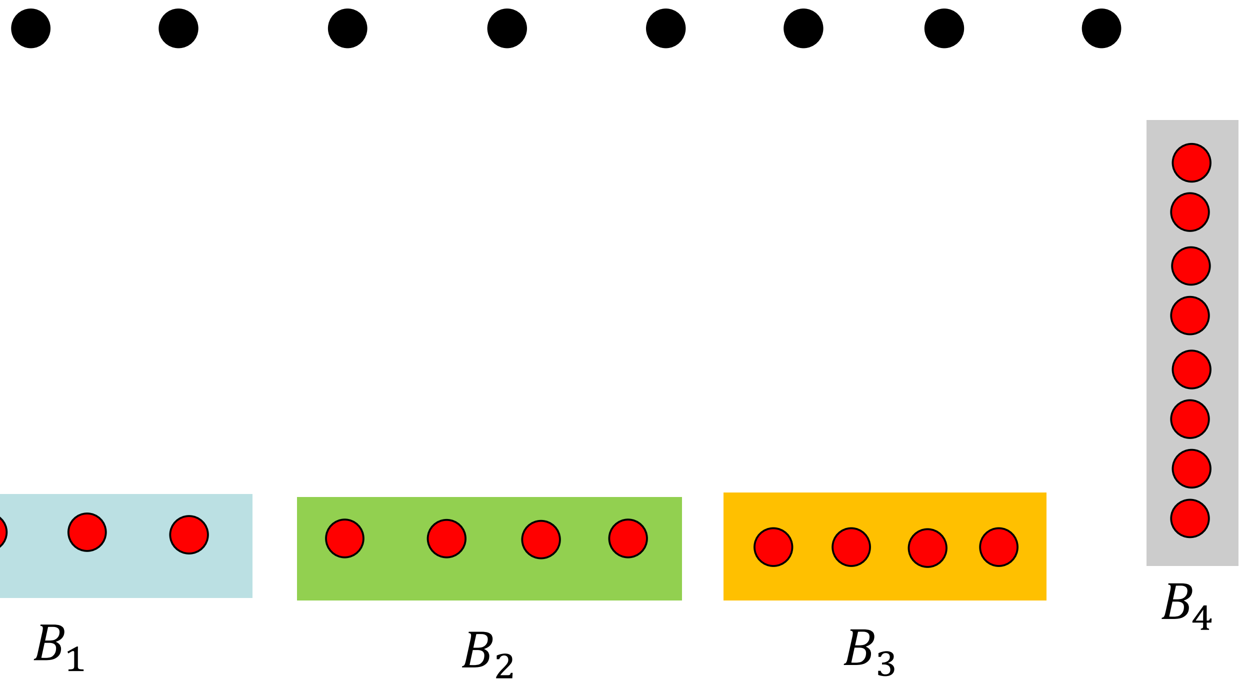
Greedy (1): Pick vertex that covers the most



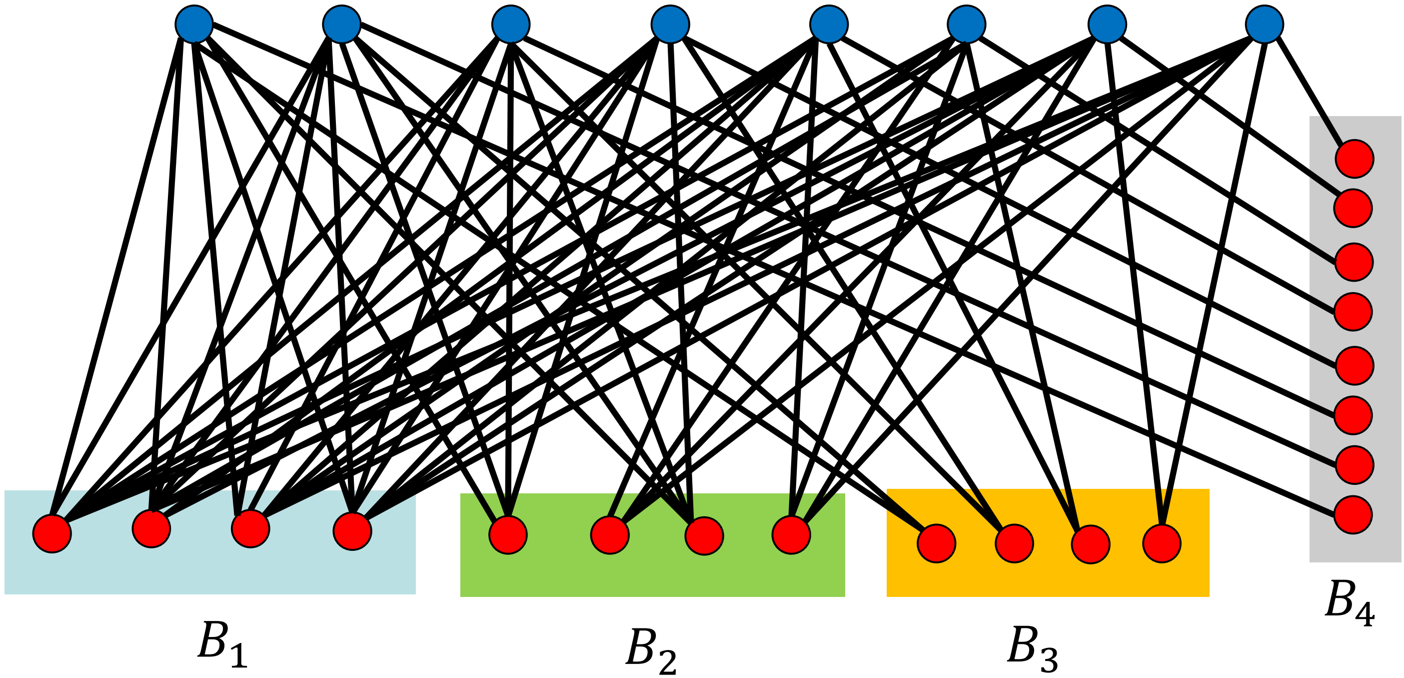
Greedy (1): Pick vertex that covers the most



Greedy (1): Pick vertex that covers the most



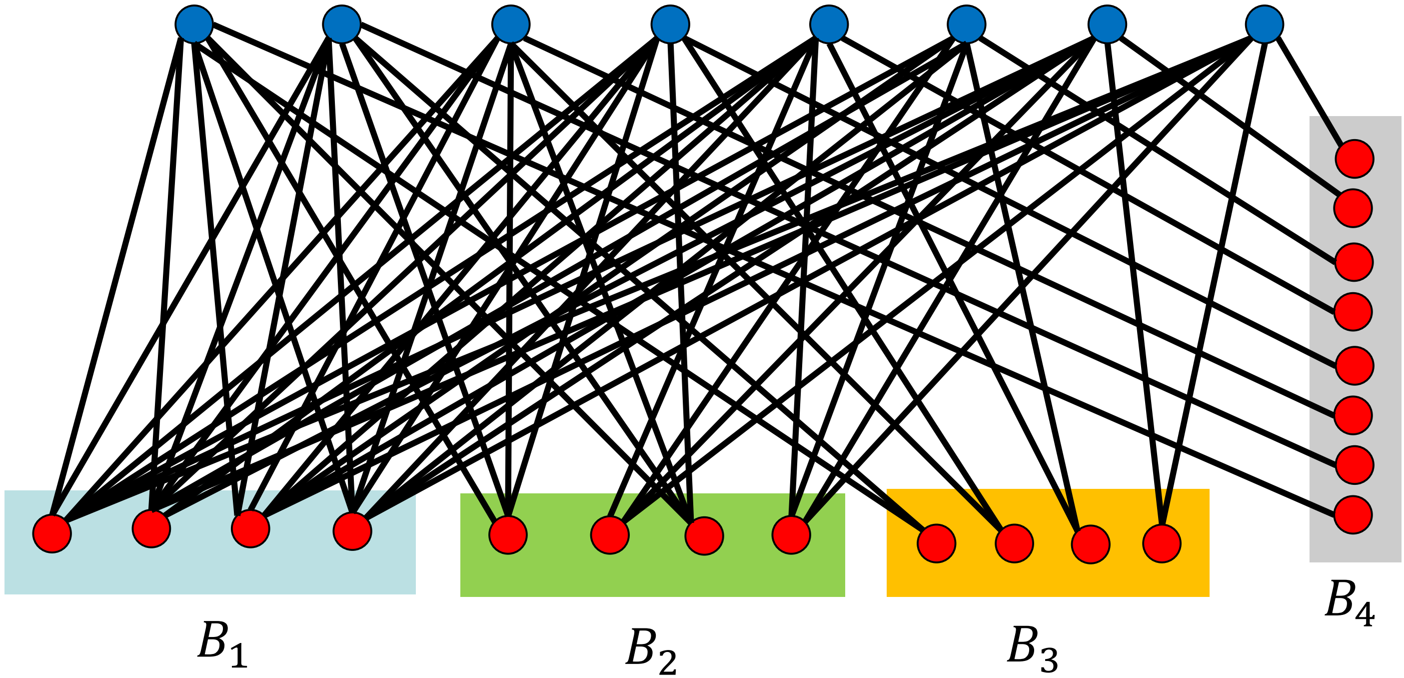
Greedy (1): Pick vertex that covers the most



Greedy Vertex cover = 20

OPT Vertex cover = 8

Greedy (1): Pick vertex that covers the most

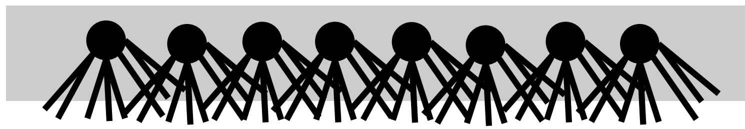


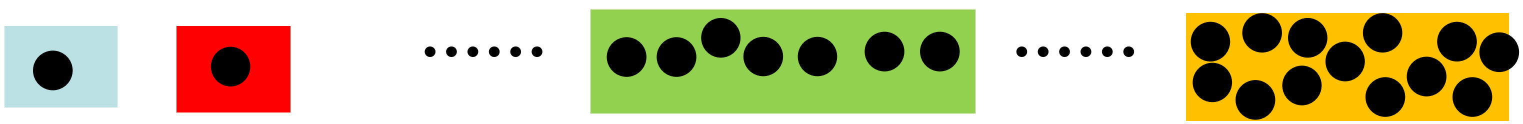
Greedy Vertex cover = 20

OPT Vertex cover = 8

Greedy (1): Pick vertex that covers the most

n vertices. Each vertex has one edge into each B_i





B_n B_{n-1} $|B_i| = n/i$ B_1

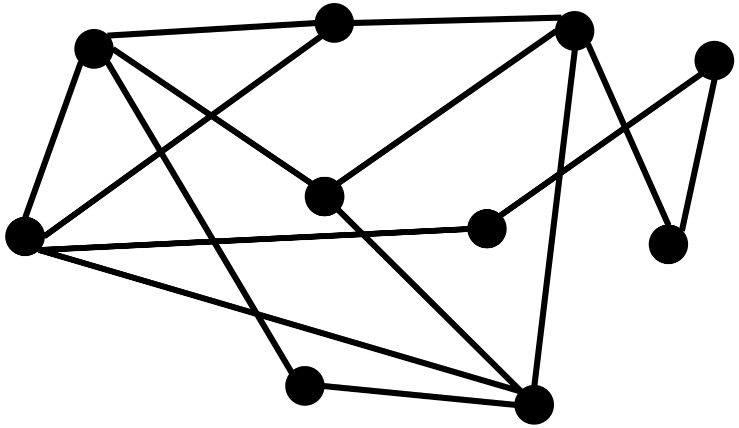
Greedy pick bottom vertices = $n + \frac{n}{2} + \frac{n}{3} + \cdots + 1 \approx n \ln n$

OPT pick top vertices = n

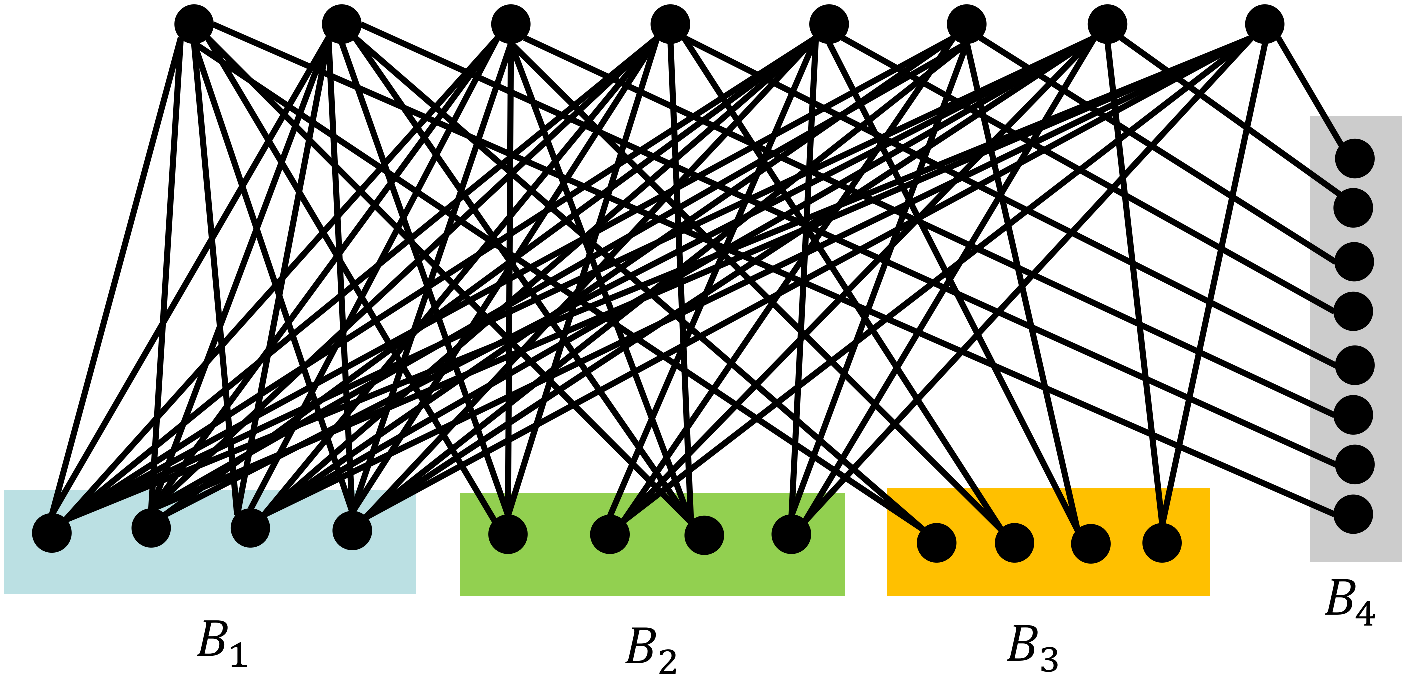
A Different Greedy Rule

Greedy 2: Iteratively, pick both endpoints of an uncovered edge.

Vertex cover = 6



Greedy 2: Pick Both endpoints of an uncovered edge



Greedy vertex cover = 16

OPT vertex cover = 8

Greedy (2) gives 2-approximation

Thm: Size of greedy (2) vertex cover is at most twice as big as size of optimal cover

Pf: Suppose Greedy (2) picks endpoints of edges $e_1, \ldots, e_k$.
Since these edges do not touch, every valid cover must pick one vertex from each of these edges!

i.e., $OPT \geq k$.

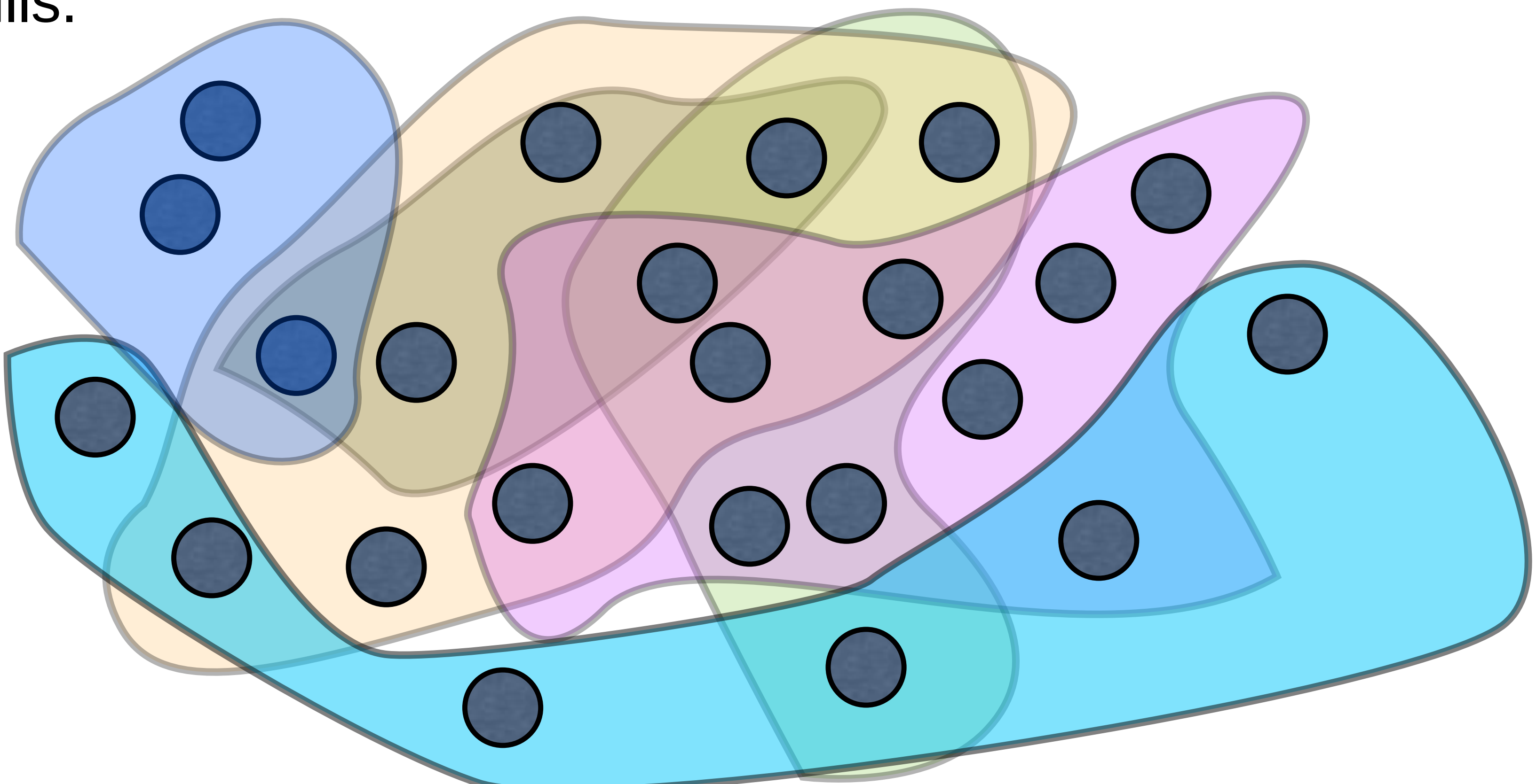
But the size of greedy cover is 2k. So, Greedy is a 2-approximation.

Set Cover

Given a number of sets on a ground set of elements,

Goal: choose minimum number of sets that cover all.

e.g., a company wants to hire employees with certain skills.

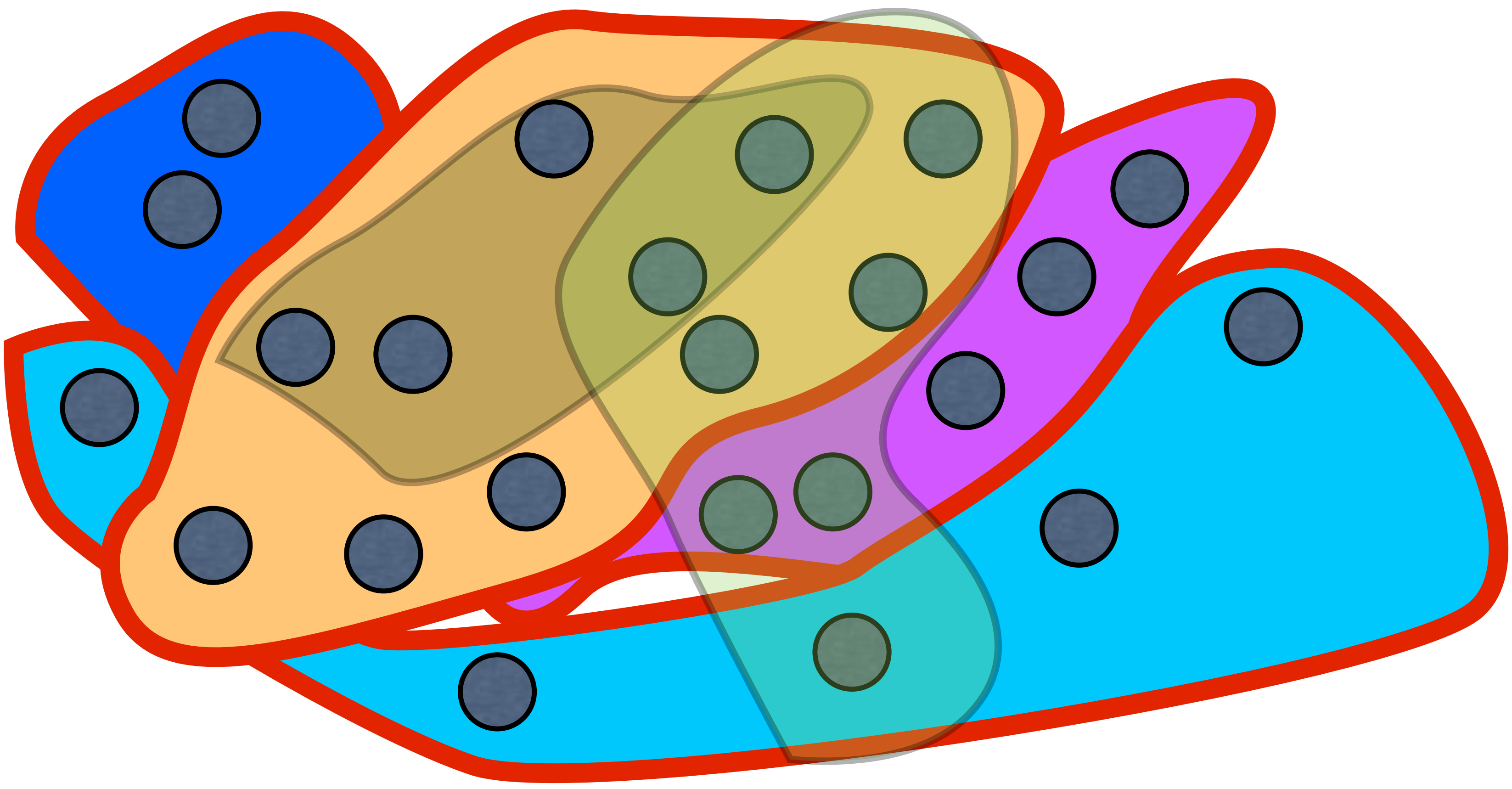


Set Cover

Given a number of sets on a ground set of elements,

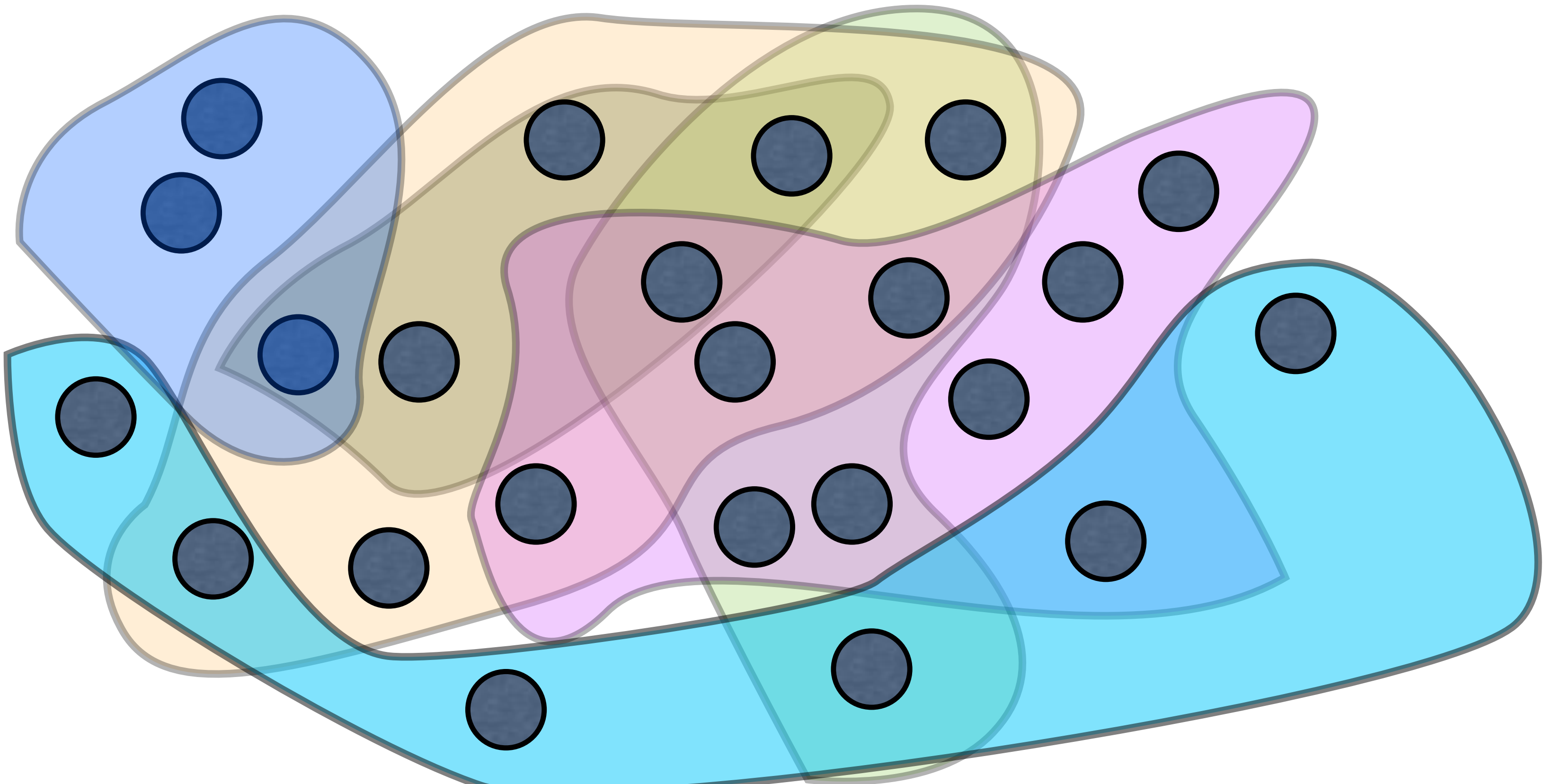
Goal: choose minimum number of sets that cover all.

Set cover = 4



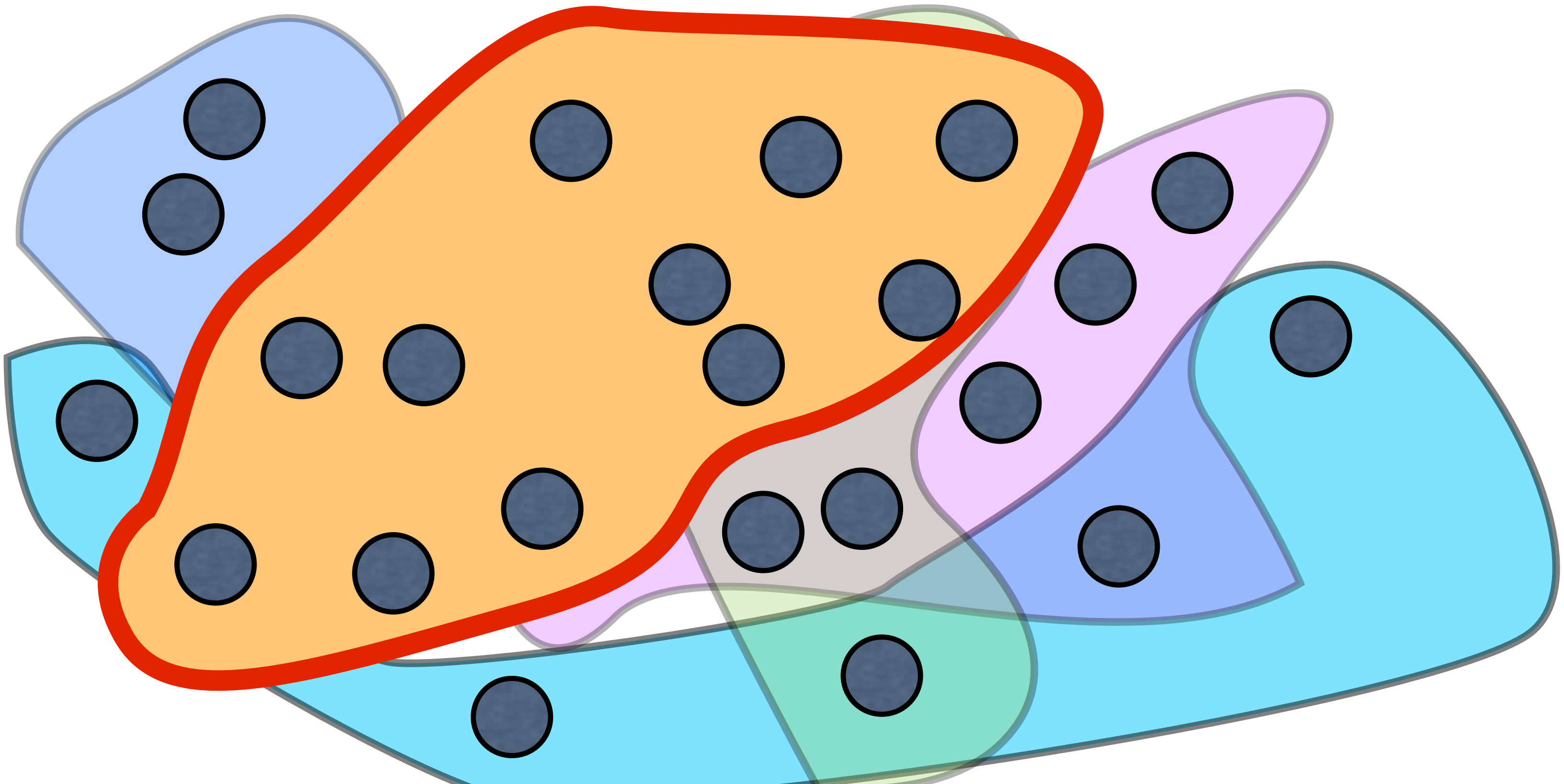
A Greedy Algorithm

Strategy: Pick the set that maximizes # new elements covered



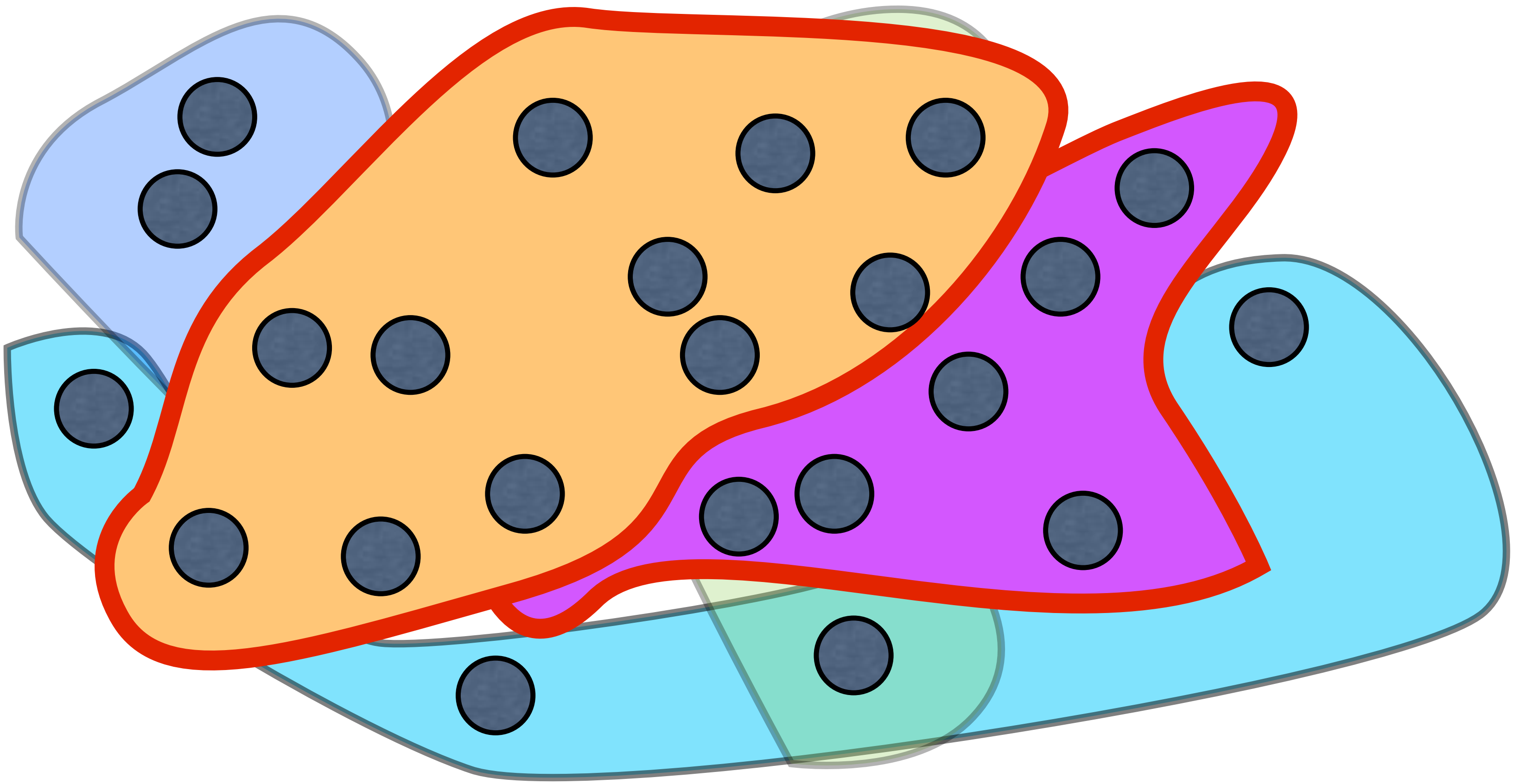
A Greedy Algorithm

Strategy: Pick the set that maximizes # new elements covered



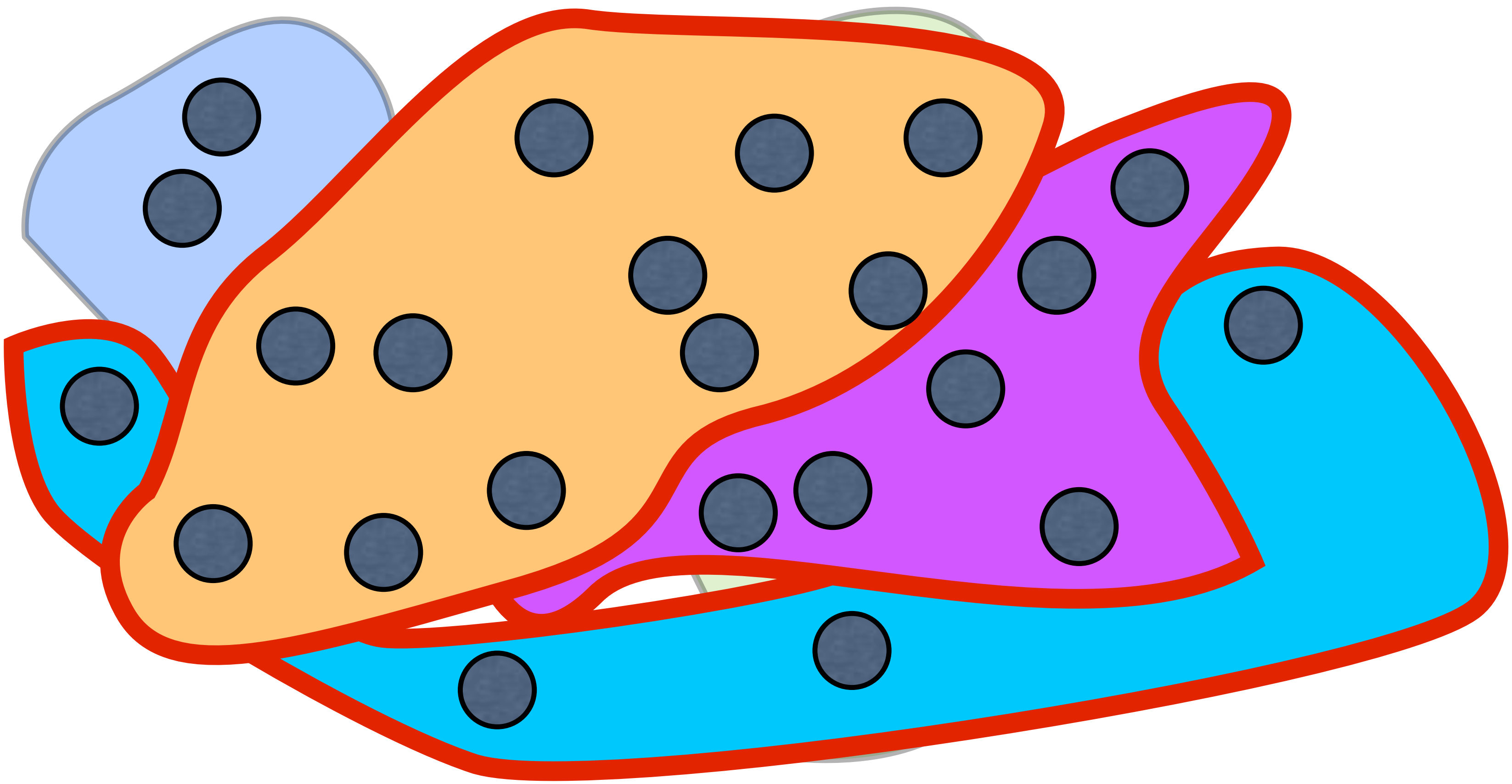
A Greedy Algorithm

Strategy: Pick the set that maximizes # new elements covered



A Greedy Algorithm

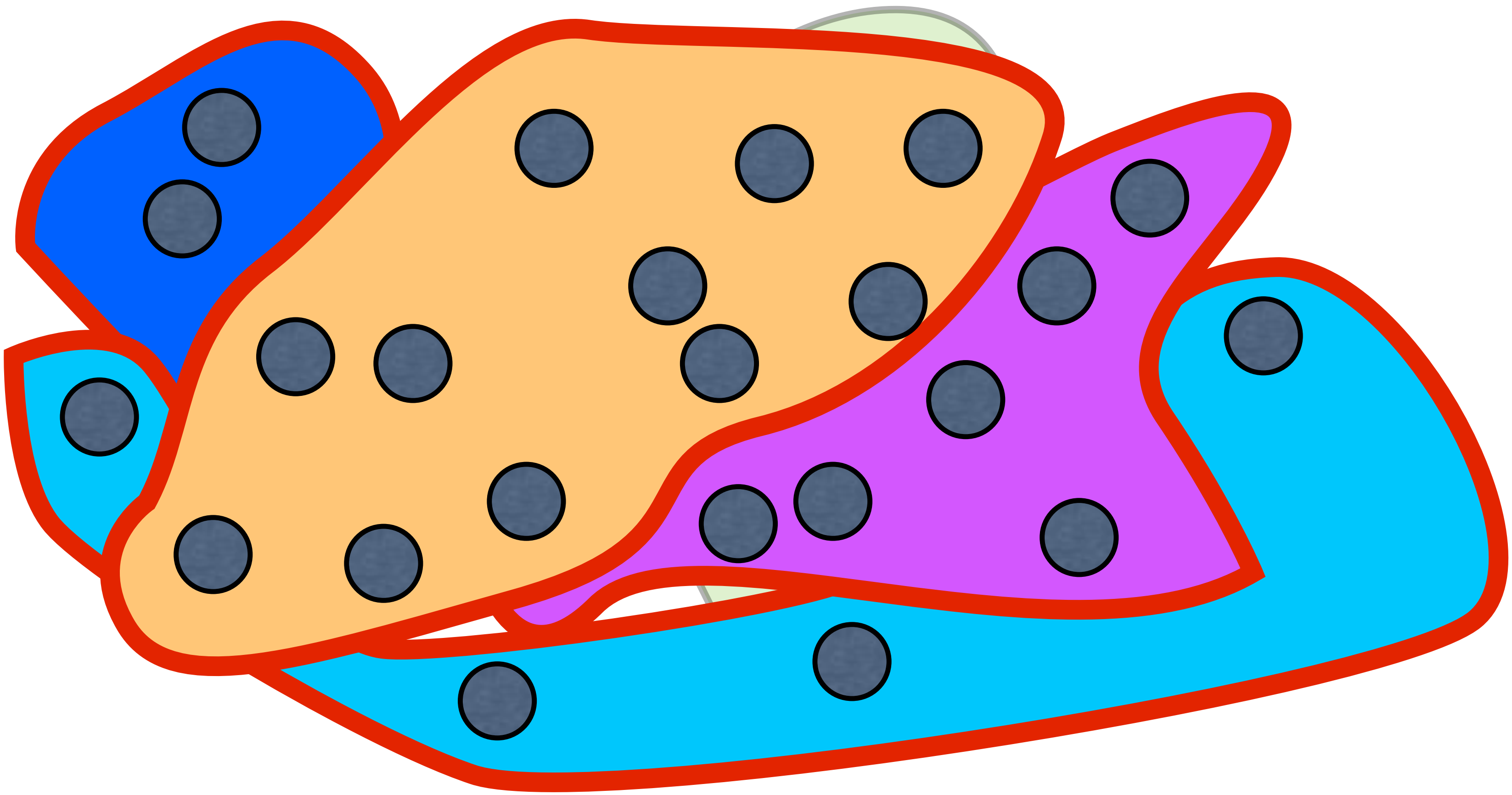
Strategy: Pick the set that maximizes # new elements covered



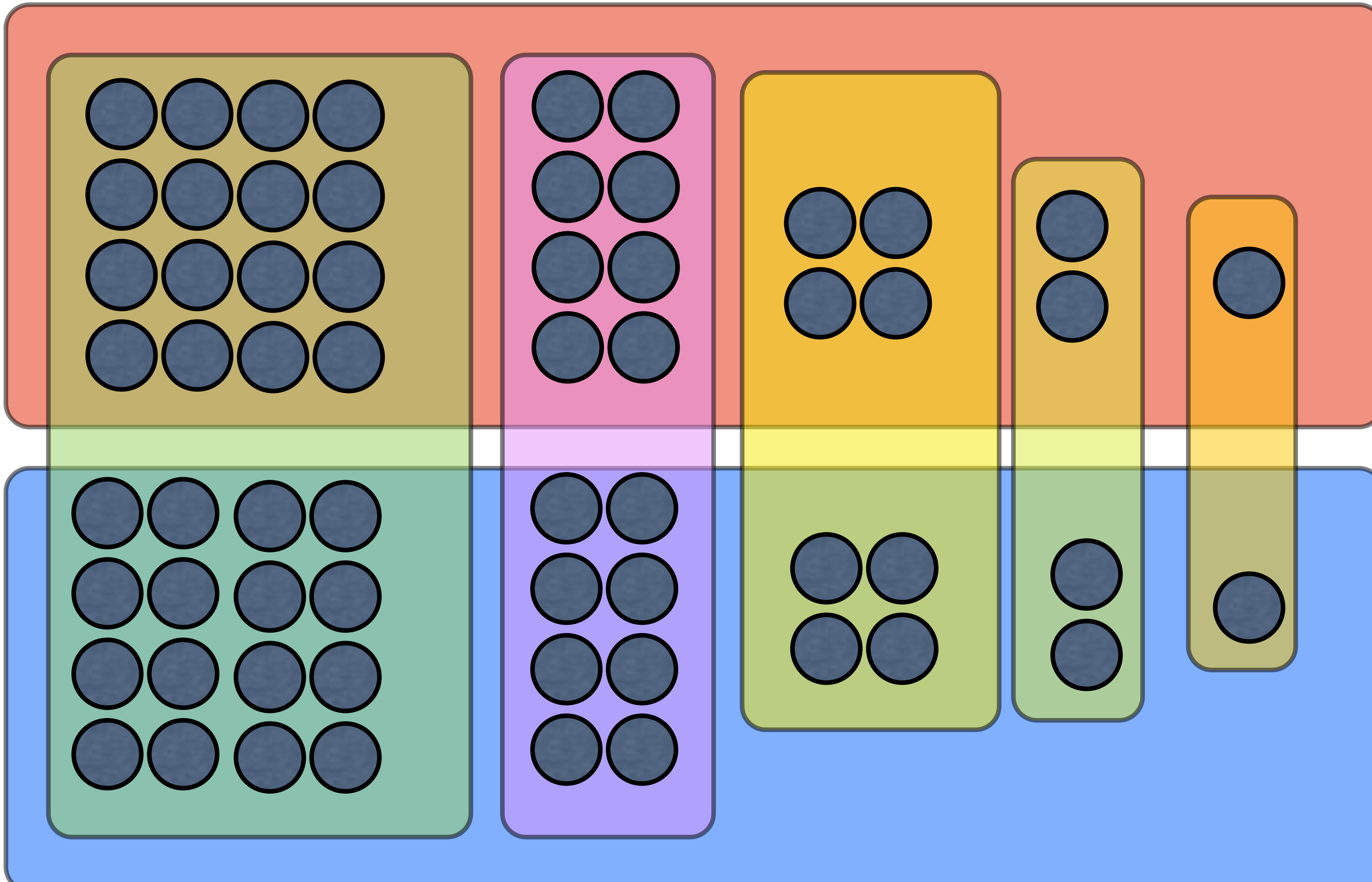
A Greedy Algorithm

Strategy: Pick the set that maximizes # new elements covered

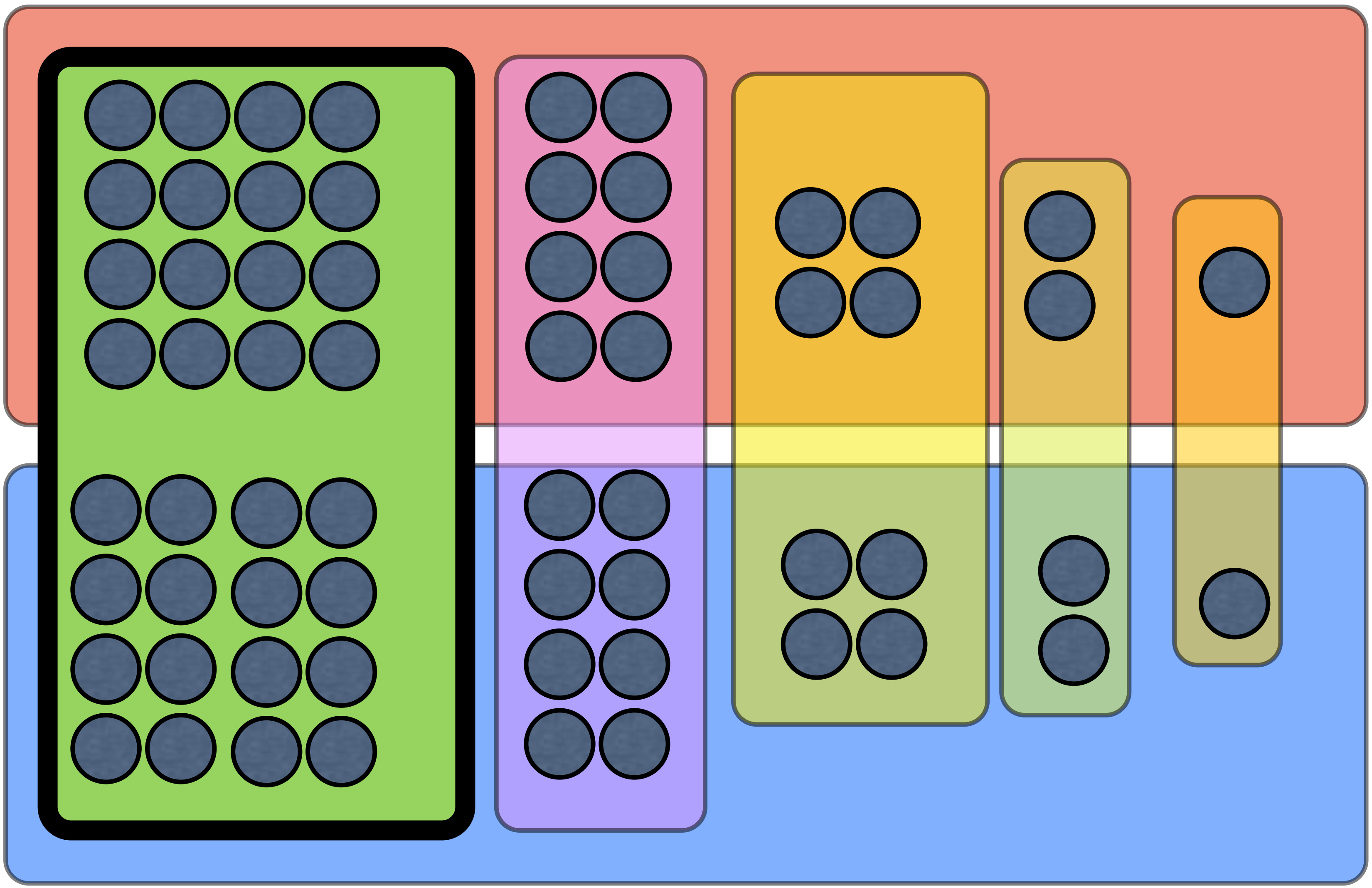
Thm: Greedy has ln n approximation ratio



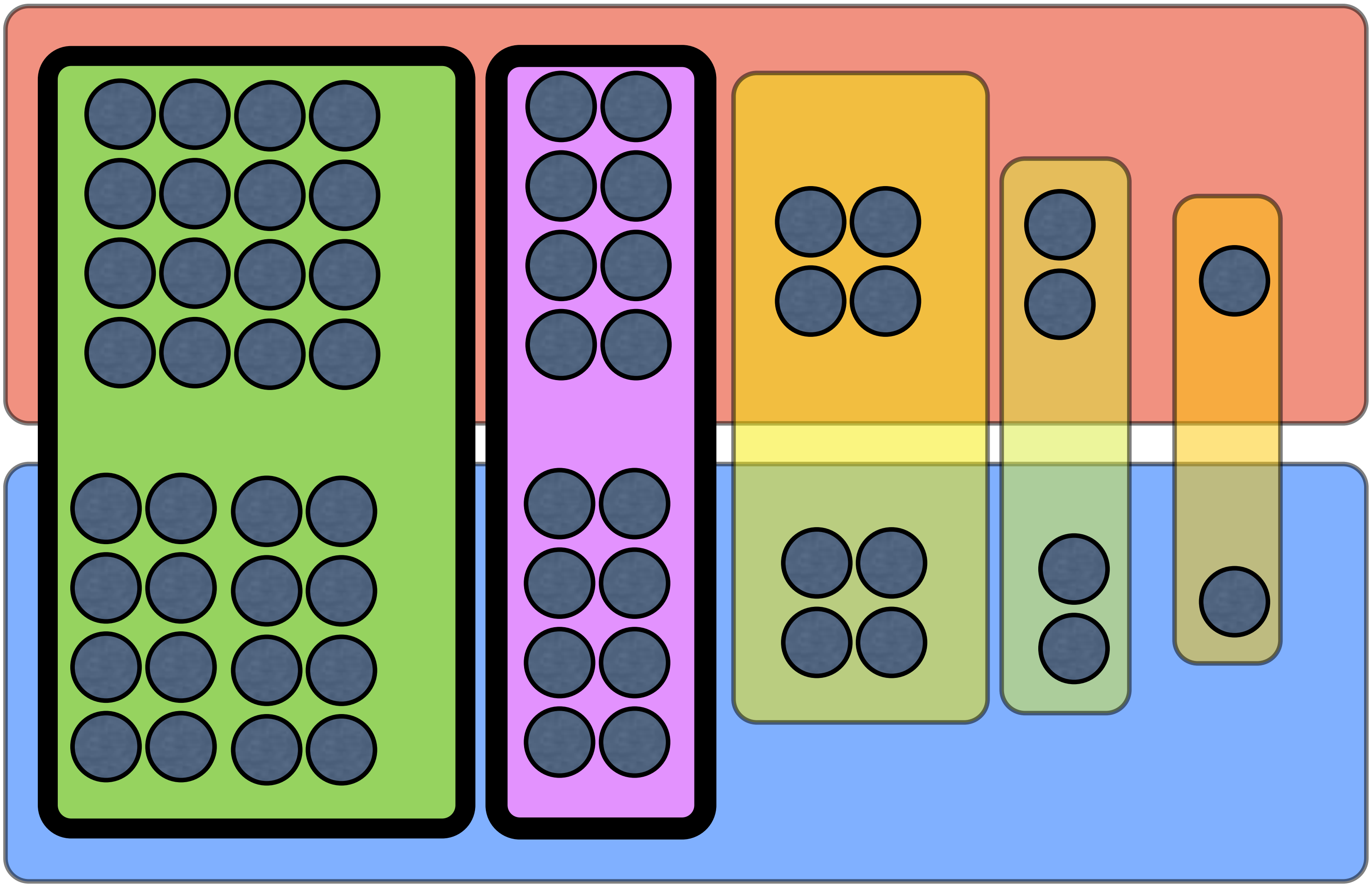
A Tight Example for Greedy



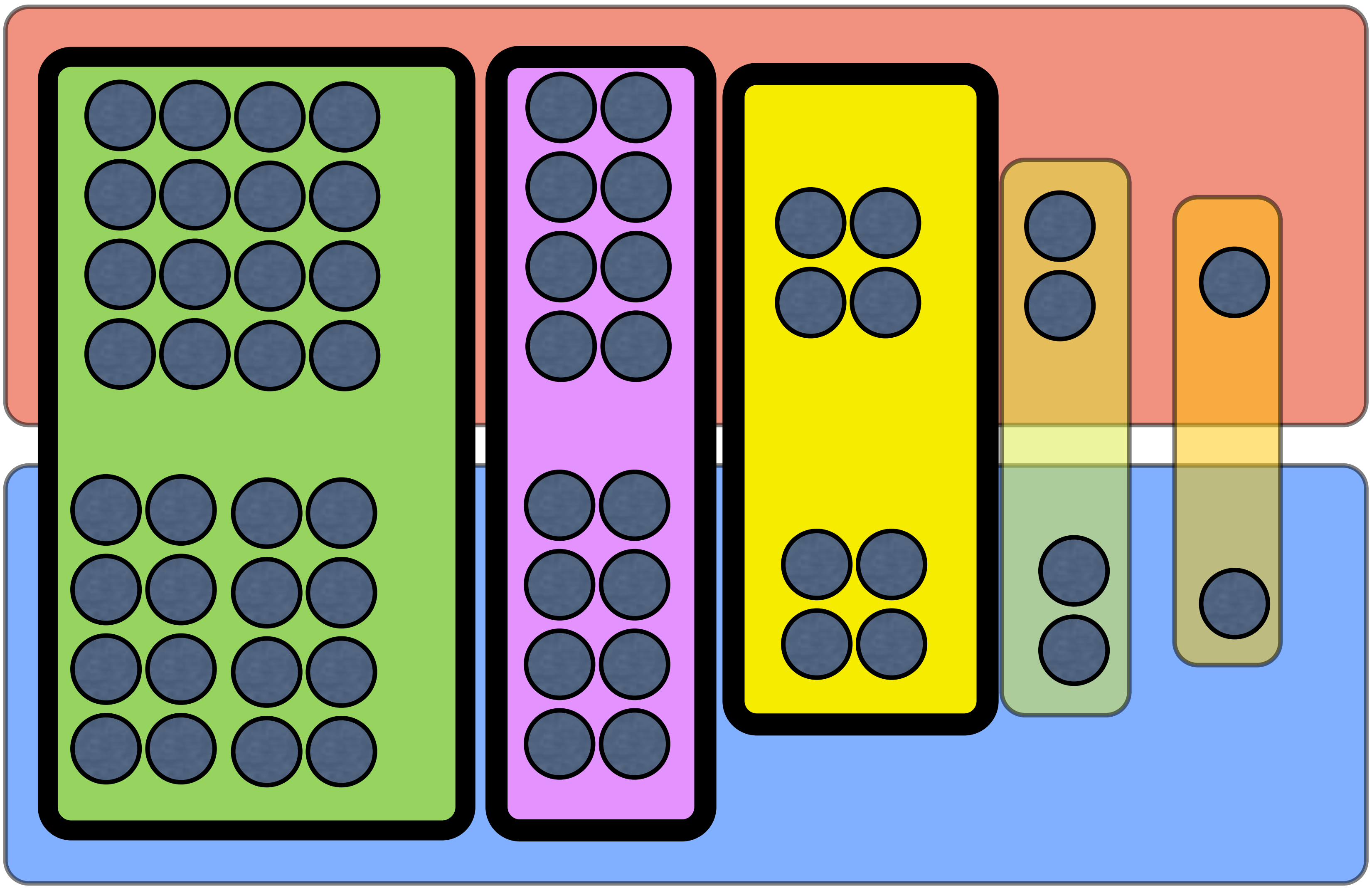
A Tight Example for Greedy



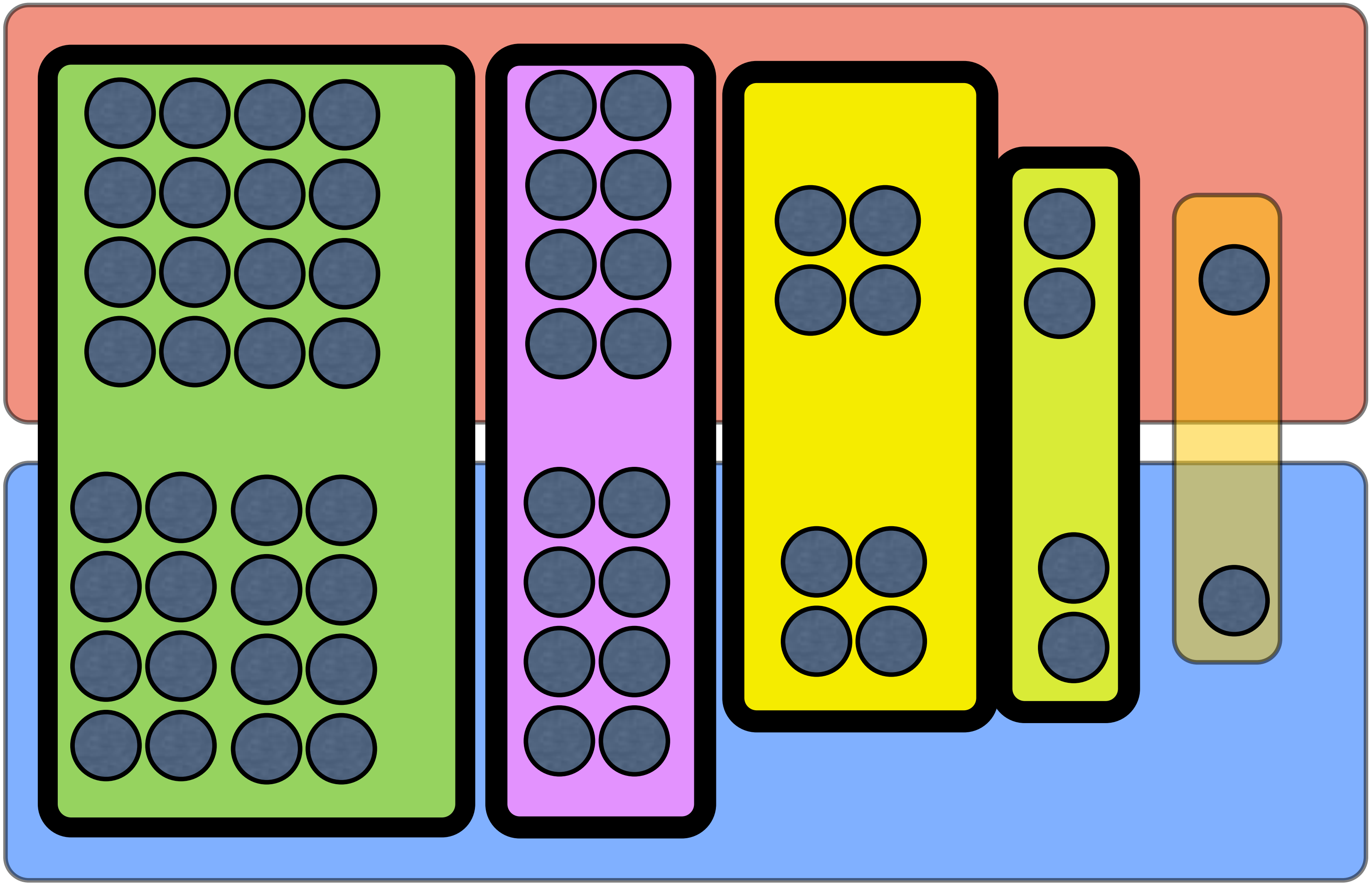
A Tight Example for Greedy



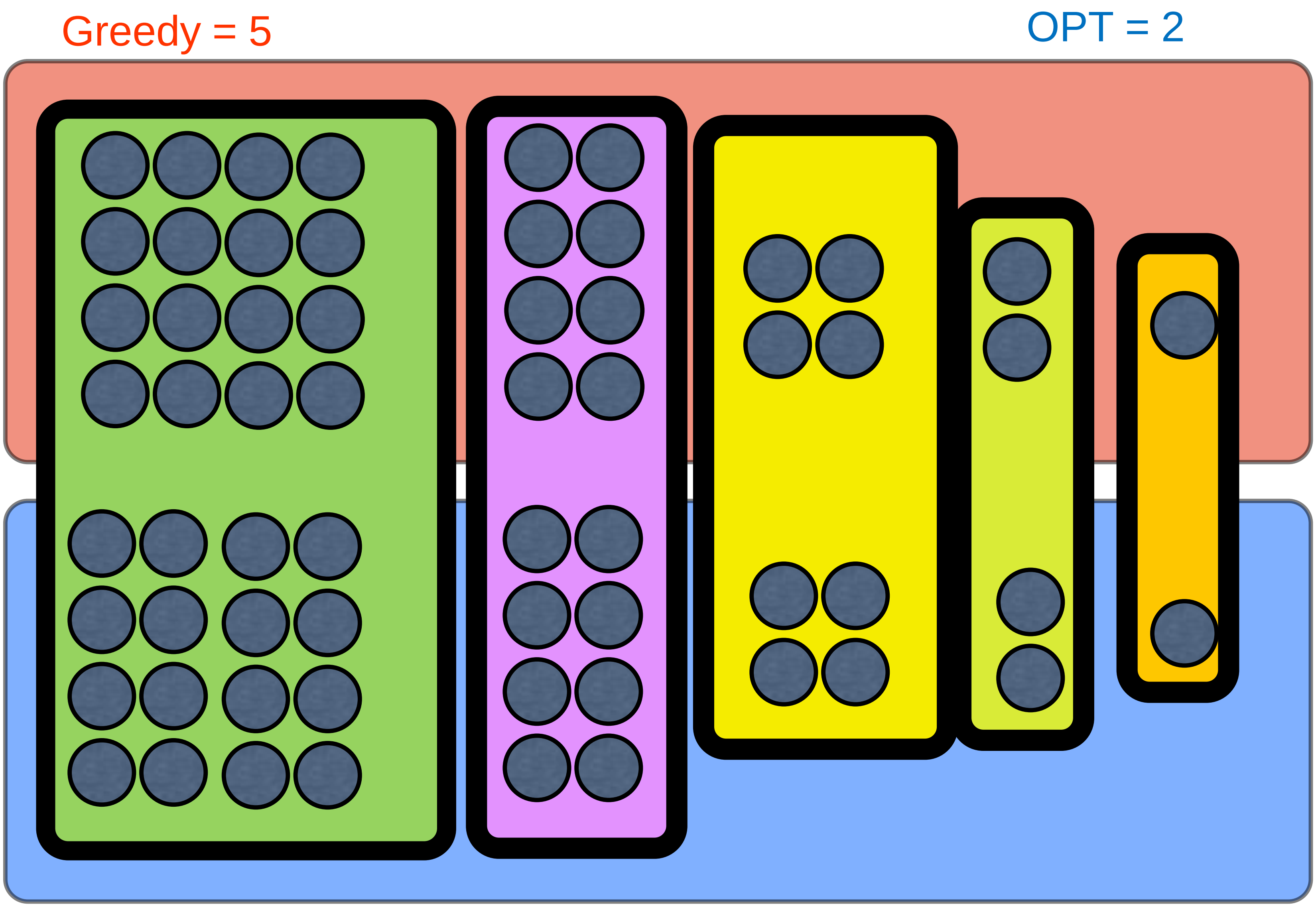
A Tight Example for Greedy



A Tight Example for Greedy



A Tight Example for Greedy



Greedy Gives O(log(n)) approximation

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

Pf: Suppose OPT=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

$$\leq n\left(1-\frac{1}{k}\right)^{t} \leq n e^{-\frac{t}{k}}$$

So after $t = k \ln n$ steps, # uncovered elements < 1.

Approximation Algorithm Summary

- The best known approximation algorithm for set cover is the greedy.
  - It is NP-Complete to obtain better than ln(n) approximation ratio for set cover.
- The best known approximation algorithm for vertex cover is the greedy.
  - It has been open for 40 years to obtain a polynomial time algorithm with approximation ratio better than 2
- There is a long list of questions we do not know the best approximation algorithm.
- https://www.youtube.com/watch?v=n7v9psW3Qwo
- https://en.wikipedia.org/wiki/Unique_games_conjecture