Greedy algorithms

Set Cover (Textbook Section 5.4)

The set cover problem

Problem (Set Cover)

The set cover problem

Problem (Set Cover)

Input:

The set cover problem

Problem (Set Cover)

Input:

- *a set B*

The set cover problem

Problem (Set Cover)

Input:

- *a set* B
- *subsets* $S_1, \ldots, S_n \subseteq B$

The set cover problem

Problem (Set Cover)

Input:

- *a set* B
- *subsets* $S_1, \ldots, S_n \subseteq B$

Output: *a collection of subsets* $S_{i_1}, \ldots, S_{i_m}$ *s.t.* $\bigcup_{k=1}^{m} S_{i_k} = B$

The set cover problem

Problem (Set Cover)

Input:

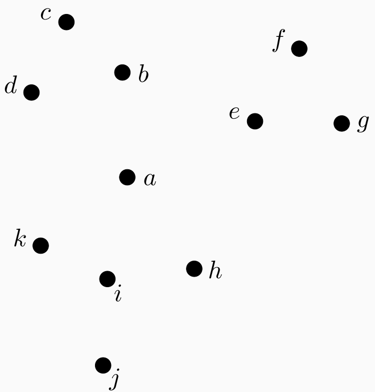
- *a set* B
- *subsets* $S_1, \ldots, S_n \subseteq B$

Output: *a collection of subsets* $S_{i_1}, \ldots, S_{i_m}$ *s.t.* $\bigcup_{k=1}^{m} S_{i_k} = B$

Goal: *minimize the number of selected subsets*

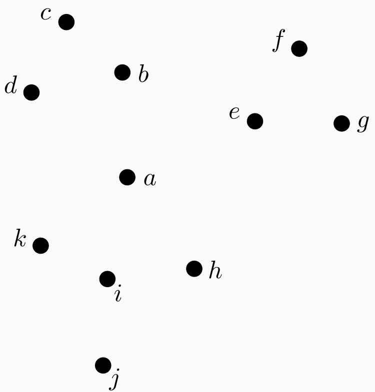
Set cover: example

Example: Each post office can serve 30 miles. Where to build post offices in centre county?



Set cover: example

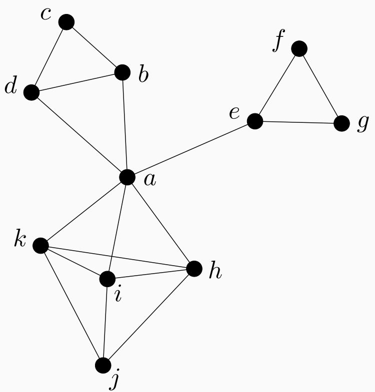
Example: Each post office can serve 30 miles. Where to build post offices in centre county?



Draw an edge if two towns are within 30 miles

Set cover: example

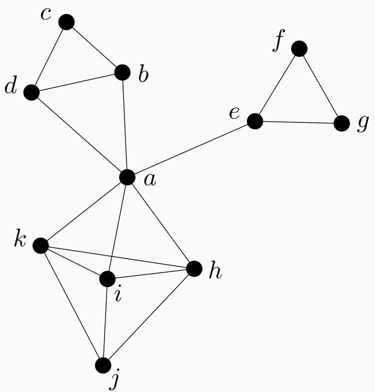
Example: Each post office can serve 30 miles. Where to build post offices in centre county?



Draw an edge if two towns are within 30 miles

Set cover: example

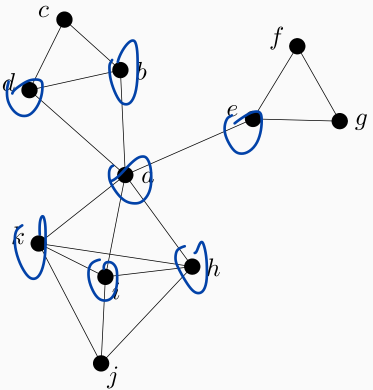
Example: Each post office can serve 30 miles. Where to build post offices in centre county?



$$B = \{a, b, \ldots, k\}$$

Draw an edge if two towns are within 30 miles

Example: Each post office can serve 30 miles. Where to build post offices in centre county?



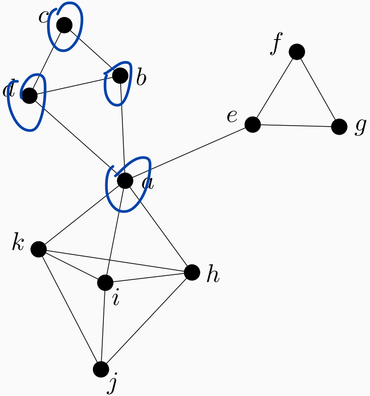
$$B = \{a, b, \ldots, k\}$$

$$S_a = \{a, b, d, e, h, i, k\}$$

Draw an edge if two towns are within 30 miles

Set cover: example

Example: Each post office can serve 30 miles. Where to build post offices in centre county?



$$B = \{a, b, \ldots, k\}$$

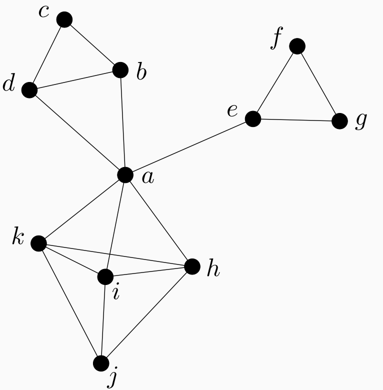
$$S_a = \{a, b, d, e, h, i, k\}$$

$$S_b = \{b, c, a, d\}$$

Draw an edge if two towns are within 30 miles

Set cover: example

Example: Each post office can serve 30 miles. Where to build post offices in centre county?



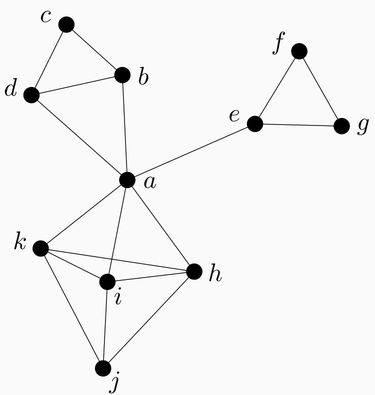
$$B = \{a, b, \ldots, k\}$$
$$S_a = \{a, b, d, e, h, i, k\}$$
$$S_b = \{b, c, a, d\}$$
$$\vdots$$
$$S_k = \{k, a, h, i, j\}$$

Draw an edge if two towns are within 30 miles

Set cover: example

Example: Each post office can serve 30 miles. Where to build post offices in centre county?

S_{i_1} S_{i_2} S_{i_3}



$$B = \{a, b, \ldots, k\}$$

$$S_a = \{a, b, d, e, h, i, k\}$$

$$S_b = \{b, c, a, d\}$$

$$\vdots$$

$$S_k = \{k, a, h, i, j\}$$

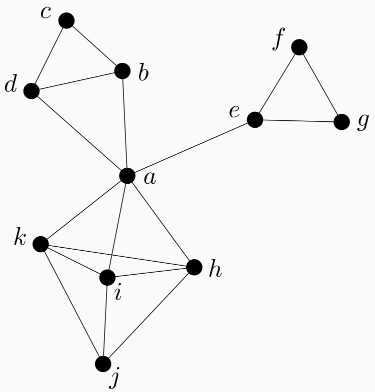
S_x: the towns within 30 miles of x

Draw an edge if two towns are within 30 miles

Set cover: greedy heuristic

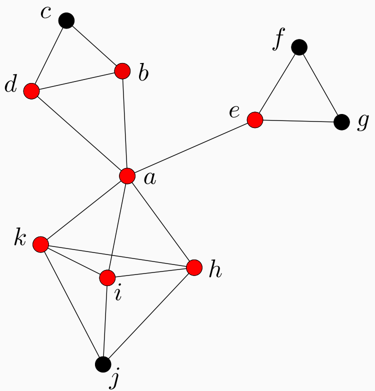
Greedy heuristic: choose the next subset with the most number of uncovered items, until B gets covered

Greedy heuristic: choose the next subset with the most number of uncovered items, until B gets covered



Set cover: greedy heuristic

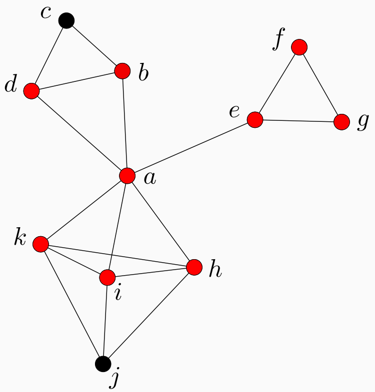
Greedy heuristic: choose the next subset with the most number of uncovered items, until B gets covered



$$S_a = \{a, b, d, e, h, i, k\}$$

Set cover: greedy heuristic

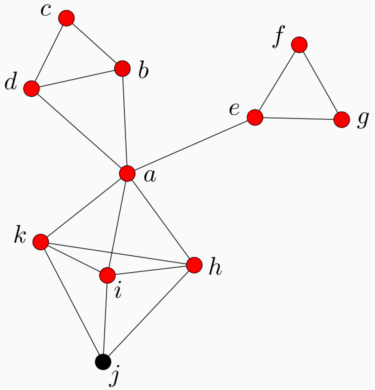
Greedy heuristic: choose the next subset with the most number of uncovered items, until B gets covered



$$S_a = \{a, b, d, e, h, i, k\}$$
$$S_f = \{f, g, e\}$$

Set cover: greedy heuristic

Greedy heuristic: choose the next subset with the most number of uncovered items, until B gets covered



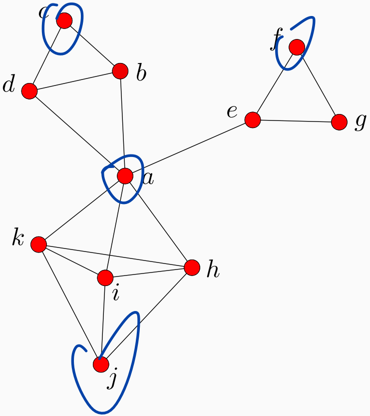
$S_a = \{a, b, d, e, h, i, k\}$

$S_f = \{f, g, e\}$

$S_c = \{c, b, d\}$

Set cover: greedy heuristic

Greedy heuristic: choose the next subset with the most number of uncovered items, until B gets covered



$S_a = \{a, b, d, e, h, i, k\}$

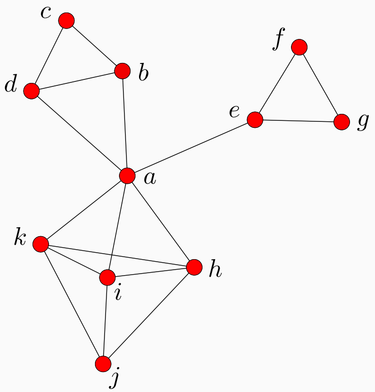
$S_f = \{f, g, e\}$

$S_c = \{c, b, d\}$

$S_j = \{i, k, j, h\}$

Set cover: greedy heuristic

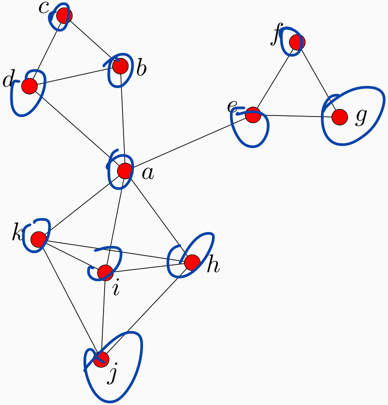
Greedy heuristic: choose the next subset with the most number of uncovered items, until B gets covered



$$S_a = \{a, b, d, e, h, i, k\}$$
$$S_f = \{f, g, e\}$$
$$S_c = \{c, b, d\}$$
$$S_j = \{i, k, j, h\}$$

Is this optimal?

Greedy heuristic: choose the next subset with the most number of uncovered items, until B gets covered



$S_a = \{a, b, d, e, h, i, k\}$

$S_f = \{f, g, e\}$

$S_c = \{c, b, d\}$

$S_j = \{i, k, j, h\}$

Is this optimal?

Optimal solution: S_b, S_e, S_i

Greedy solution is not too bad

Although the greedy solution is not optimal, but it's not off by much

Greedy solution is not too bad

Although the greedy solution is not optimal, but it's not off by much

Theorem

Assume $|B| = n$ and the optimal solution uses k subsets.

$\ln = \log_e$

Although the greedy solution is not optimal, but it's not off by much

Theorem

Assume $|B| = n$ and the optimal solution uses k subsets. Then the greedy algorithm uses at most $k \ln(n)$ subsets

Greedy solution is not too bad

Although the greedy solution is not optimal, but it's not off by much

Theorem

Assume $|B| = n$ and the optimal solution uses k subsets. Then the greedy algorithm uses at most $k \ln(n)$ subsets

$\ln(n)$: *approximation ratio*

Greedy solution is not too bad

Although the greedy solution is not optimal, but it's not off by much

Theorem

Assume $|B| = n$ *and the optimal solution uses* k *subsets. Then the greedy algorithm uses at most* $k\ln(n)$ *subsets*

$\ln(n)$: *approximation ratio*

More about **approximation algorithms**: CSE 565

Proof: Let n_t be the number of elements not covered by the greedy algorithm after t iterations.

Proof: Let n_t be the number of elements not covered by the greedy algorithm after t iterations. These remaining n_t elements are covered by the optimal k subsets.

Proof: Let n_t be the number of elements not covered by the greedy algorithm after t iterations. These remaining n_t elements are covered by the optimal k subsets. So some subsets has $\geq \frac{n_t}{k}$ of these uncovered elements,

Suppose not, all of these subsets have size $< \frac{n_t}{k}$

$\Rightarrow$ total # of such items is $< k \cdot \frac{n_t}{k} = n_t$

Proof: Let n_t be the number of elements not covered by the greedy algorithm after t iterations. These remaining n_t elements are covered by the optimal k subsets. So some subsets has $\geq \frac{n_t}{k}$ of these uncovered elements, and the greedy algorithm will pick a set of size at least $\frac{n_t}{k}$.

$$n_{t+1} \leq n_t - \frac{n_t}{k}$$

Proof: Let n_t be the number of elements not covered by the greedy algorithm after t iterations. These remaining n_t elements are covered by the optimal k subsets. So some subsets has $\geq \frac{n_t}{k}$ of these uncovered elements, and the greedy algorithm will pick a set of size at least $\frac{n_t}{k}$.
So, $n_{t+1} \leq n_t - \frac{n_t}{k} = n_t \left(1 - \frac{1}{k}\right)$

$n_0 = n$

$$n_t \leq n_{t-1} \left(1 - \frac{1}{k}\right) \leq n_{t-2} \left(1 - \frac{1}{k}\right)^2 \leq \cdots \leq n_0 \left(1 - \frac{1}{k}\right)^t$$

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

Proof: Let n_t be the number of elements not covered by the greedy algorithm after t iterations. These remaining n_t elements are covered by the optimal k subsets. So some subsets has $\geq \frac{n_t}{k}$ of these uncovered elements, and the greedy algorithm will pick a set of size at least $\frac{n_t}{k}$.
So, $n_{t+1} \leq n_t - \frac{n_t}{k} = n_t \left(1 - \frac{1}{k}\right)$

Repeatedly applying this:

Proof: Let n_t be the number of elements not covered by the greedy algorithm after t iterations. These remaining n_t elements are covered by the optimal k subsets. So some subsets has $\geq \frac{n_t}{k}$ of these uncovered elements, and the greedy algorithm will pick a set of size at least $\frac{n_t}{k}$.
So, $n_{t+1} \leq n_t - \frac{n_t}{k} = n_t\left(1 - \frac{1}{k}\right)$

Repeatedly applying this:

$$n_t \leq n_{t-1}\left(1 - \frac{1}{k}\right) \leq n_{t-2}\left(1 - \frac{1}{k}\right)^2 \leq \cdots \leq n_0\left(1 - \frac{1}{k}\right)^t = n\boxed{\left(1 - \frac{1}{k}\right)^t}$$

Proof: Let n_t be the number of elements not covered by the greedy algorithm after t iterations. These remaining n_t elements are covered by the optimal k subsets. So some subsets has $\geq \frac{n_t}{k}$ of these uncovered elements, and the greedy algorithm will pick a set of size at least $\frac{n_t}{k}$.
So, $n_{t+1} \leq n_t - \frac{n_t}{k} = n_t \left(1 - \frac{1}{k}\right)$

Repeatedly applying this:

$$n_t \leq n_{t-1}\left(1 - \frac{1}{k}\right) \leq n_{t-2}\left(1 - \frac{1}{k}\right)^2 \leq \cdots \leq n_0 \left(1 - \frac{1}{k}\right)^t = n\left(1 - \frac{1}{k}\right)^t$$

Using the fact: $\boxed{1 - x \leq e^{-x}}$ (equality when $x = 0$)

$$n_t \leq n\left(1 - \frac{1}{k}\right)^t \leq \boxed{ne^{-t/k}}$$

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

Proof: Let n_t be the number of elements not covered by the greedy algorithm after t iterations. These remaining n_t elements are covered by the optimal k subsets. So some subsets has $\geq \frac{n_t}{k}$ of these uncovered elements, and the greedy algorithm will pick a set of size at least $\frac{n_t}{k}$.
So, $n_{t+1} \leq n_t - \frac{n_t}{k} = n_t\left(1 - \frac{1}{k}\right)$

Repeatedly applying this:

$$n_t \leq n_{t-1}\left(1 - \frac{1}{k}\right) \leq n_{t-2}\left(1 - \frac{1}{k}\right)^2 \leq \cdots \leq n_0\left(1 - \frac{1}{k}\right)^t = n\left(1 - \frac{1}{k}\right)^t$$

Using the fact: $1 - x \leq e^{-x}$ (equality when $x = 0$)

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

Greedy algorithm terminates when $n_t < 1$. Let's find out what t makes $n_t < 1$

$\Rightarrow n_t < ne^{-t/k} \leq 1$

Since $n_t < ne^{-t/k}$, it suffices to make $ne^{-t/k} \leq 1$

Since $n_t < ne^{-t/k}$, it suffices to make $ne^{-t/k} \leq 1$

Solving $ne^{-t/k} \leq 1$ $\iff e^{-t/k} \leq \frac{1}{n} \iff -\frac{t}{k} \leq \ln\left(\frac{1}{n}\right)$

$\iff t \geq -k\ln\left(\frac{1}{n}\right)$

$= k\ln(n)$

Since $n_t < ne^{-t/k}$, it suffices to make $ne^{-t/k} \leq 1$

Solving $ne^{-t/k} \leq 1$

$\iff e^{-t/k} \leq \frac{1}{n} \iff -\frac{t}{k} \leq \ln(\frac{1}{n}) \iff t \geq -k\ln(\frac{1}{n}) = \boxed{k\ln(n)}$

then $n_t < 1$

t: # of iterations

= # of subsets picked by the greedy alg.

Since $n_t < ne^{-t/k}$, it suffices to make $ne^{-t/k} \leq 1$

Solving $ne^{-t/k} \leq 1$

$\iff e^{-t/k} \leq \frac{1}{n} \iff -\frac{t}{k} \leq \ln(\frac{1}{n}) \iff t \geq -k\ln(\frac{1}{n}) = k\ln(n)$

At $t = k\ln(n)$, $n_t < 1$. Everything is covered $\square$

Since $n_t < ne^{-t/k}$, it suffices to make $ne^{-t/k} \leq 1$

Solving $ne^{-t/k} \leq 1$

$\iff e^{-t/k} \leq \frac{1}{n} \iff -\frac{t}{k} \leq \ln(\frac{1}{n}) \iff t \geq -k\ln(\frac{1}{n}) = k\ln(n)$

At $t = k\ln(n)$, $n_t < 1$. Everything is covered $\square$

Proof of the fact $\boxed{1 - x \leq e^{-x}}$ (equality when $x = 0$):

Since $n_t < ne^{-t/k}$, it suffices to make $ne^{-t/k} \leq 1$

Solving $ne^{-t/k} \leq 1$

$\iff e^{-t/k} \leq \frac{1}{n} \iff -\frac{t}{k} \leq \ln(\frac{1}{n}) \iff t \geq -k\ln(\frac{1}{n}) = k\ln(n)$

At $t = k\ln(n)$, $n_t < 1$. Everything is covered $\square$

Proof of the fact $1 - x \leq e^{-x}$ (equality when $x = 0$):

Consider $f(x) = e^{-x} - (1 - x) \geq 0$

Since $n_t < ne^{-t/k}$, it suffices to make $ne^{-t/k} \leq 1$

Solving $ne^{-t/k} \leq 1$
$\iff e^{-t/k} \leq \frac{1}{n} \iff -\frac{t}{k} \leq \ln(\frac{1}{n}) \iff t \geq -k\ln(\frac{1}{n}) = k\ln(n)$

At $t = k\ln(n)$, $n_t < 1$. Everything is covered $\square$

Proof of the fact $1 - x \leq e^{-x}$ (equality when $x = 0$):
Consider $f(x) = \boxed{e^{-x} - (1 - x)} \geq 0$
$f'(x) = -e^{-x} + 1$. Critical point at $x = 0$, achieving minimum $\square$