Recall \( P \neq NP \)

\[ \begin{array}{c}
P \\
\circ
\end{array} \]

\( \circ \) \( NP \)

\( ? \) \( ? \) \( ? \) \( ? \)

- factoring?
- graph isomorphism?
- few other natural examples.

NP-complete - hardest problems in NP
- open - to find poly time algs.
- i.e. can't be correct and fast.

(Ladner proved: if \( P \neq NP \) then there are inf. many problems between \( P \) and \( NP \)-complete.

Major theoretical and practical question: is \( P = NP ? \)

Previous section of course - does randomness help? open
(though it's a good tool to have in your arsenal, and
gives faster/simpler algorithms)

What to do with NP-complete problems?
- heuristics
- exact exponential time (as efficient as possible)
- other models of computing (quantum, natural)
- approximation algns
- parameterized complexity (next section of course)

give up on exact
but ask for guarantee on quality of solution
An approximation alg. for optimization problem finds in poly-time a solution "close to" the opt.

"close to" - difference is small

or

- ratio of solution to opt. is good

An extreme example

**Edge-colouring in graph**: Given a graph colour the edges s.t. if two edges are incident, they have different colours

This problem is NP-complete:

Given G, k ∈ N, can you edge-colour G with k colours?

**Vizing's Thm**

\[ \Delta \leq \min \text{# colours} \leq \Delta + 1 \]

\[ \Delta = \max \text{ degree} \]

\[ \{ \text{t edges} \} \text{ need at least t colours.} \]

Furthermore, there is poly-time alg. to colour any graph with \( \Delta + 1 \) colours.

So we can approximate within \( +1 \) of opt.

This is rare, usually we will just get good ratio of approx to opt.
**Vertex Cover**: Given a graph $G = (V, E)$ find a min. size vertex cover — a set $U \subseteq V$ s.t. every edge has at least one endpoint in $U$.

![Graph](image)

This is min. vertex cover, size 3

Application — monitor all links in network.

(decision version of) Vertex Cover is NP-complete.

$3SAT \leq_p$ Independent Set $\leq_p$ Vertex Cover.

Find max set of vertices, no two joined by edge.

**Argument**: $G$

$U$ is a vertex cover iff $V \setminus U$ is an ind. set.
**EX:** Prove that if there is an approx alg. for vertex cover that's good within additive constant (as for edge colouring) then $P = NP$.

**Greedy Algorithm**

```
C ← φ
repeat
    C ← C ∪ {vertex of max degree},
    remove covered edges
until no edges remain.
```

We will show $|C| = O(\log n) \cdot |OPT|$

**EX:** Show the greedy alg. can give $\frac{|C|}{|OPT|} \leq \frac{1}{\log n}$

**more general problem:**

**Set Cover Problem:** Given a collection of sets $S_1, S_2, \ldots, S_k$, $S_i \subseteq \{1, \ldots, n\}$

find a min. size subcollection s.t. every element $1 \ldots n$ is covered

i.e. $C \subseteq \{1, \ldots, k\}$

$\forall i \in \{1, \ldots, n\}$, $i \in S_j$ for some $j \in C$.

e.g. elements $1, 2, 3, 4$

$S_1 = \{1, 2\}$ $S_1, S_3$ cover

$S_2 = \{1, 3\}$

$S_3 = \{2, 3, 4\}$

$S_4 = \{1, 4\}$

**Application:** Set = type of pizza elements = people

element $\in$ set = person eats that type of pizza

find min. # of pizza types to feed all.
Vertex Cover is the special case of Set Covering where element = edge set = edges incident to a vertex

\[ S(v_1) = \{1, 3, 4\} \]
\[ S(v_2) = \{1, 2\} \]

Is every Set Cover Problem a Vertex Cover Problem?
No. The special property of Vertex Cover is that each element is in only 2 sets.

---

**Greedy Alg. for Set Cover.**

\[
\begin{align*}
C &\leftarrow \emptyset \\
\text{while there are uncovered elements} & \\
S_i &\leftarrow \text{a set that covers max # uncovered elements} \\
C &\leftarrow C \cup \{ S_i \} \\
\text{end}
\end{align*}
\]

Ex.

- pick set 1
- pick set 2
- pick set 3

`greedy is not opt. here.`
**Theorem** The greedy alg. is poly. time with approx. factor \( O(\log n) \).

i.e. \( |c| \leq O(\log n) |OPT| \) where \( OPT = \min \text{ set cover} \).

**Proof:** From Vazirani’s book - better than pf. in CLRS.

Distribute cost of choosing a set \( S_i \) over the newly covered elements. Note: cost of choosing a set is 1.

Let \( c(e) = \text{cost of element } e \)

e.g. if \( e \) is covered by first chosen set \( S \), \( c(e) = \frac{1}{|S|} \)

Since \( S \) is max size set \( \geq \text{max size set in } OPT \)

\[ \geq \text{avg size set in } OPT \geq \frac{n}{|OPT|} \quad \text{clever!} \]

\[ \therefore c(e) = \frac{1}{|S|} \leq \frac{|OPT|}{n} \]

more generally, let \( e_1, e_2, \ldots, e_n \)

be ordering of elements as they are covered. (lots of ties)

Consider \( e_i \) first covered by \( S' \) say

Suppose \( S' \) covers \( t \) new elements

\[ \otimes c(e_i) = \frac{1}{t} \]

Previous sets cover \( \leq i-1 \) elements

\[ e_1 \ldots \ldots e_i \ldots \ldots e_n \]

newly covered by \( S' \)

\[ \therefore \text{# uncovered elements when we choose } S' \text{ is } \geq n-i+1 \]

\[ S' \text{ covers } t, \text{ which is } \max \geq \max \geq \text{avg - } \geq \frac{n-i+1}{|OPT|} \]

from \( \otimes c(e_i) \leq \frac{|OPT|}{n-i+1} \).

\[ \text{avg (cover sets in OPT)} \]

\[ \text{# uncovered elements} \]

\[-125-\]
Suppose that the optimum cover has \( k \) sets. Then at a stage where \( r \) items remain to be covered, some set covers \( r/k \) of them. Therefore, choosing the set that covers the most new points reduces the number of uncovered points to \( r - r/k = r(1 - 1/k) \). If we start with \( n \) points and repeat this process \( n \) times, \( r \leq n(1 - 1/k) \). Since \( r \) is an integer, we are done when \( n(1 - 1/k) \leq 1 \). This happens when \( j = O(k \log n) \). Hence, \( \alpha = j/k = O(\log n) \).

Next Day: An approximate alg. for vertex cover with approx factor 2.