Bellman-Ford

\[ |V| = n \]
\[ |E| = m \]
\[ O(nm) \text{ time} \]

Dijkstra's
\[ O \left( (n+m) \log n \right) \]

\textbf{APSP :} Run either BF or Dijkstra's from every vertex
\[ \Rightarrow O(n^2m) \lor \left( (n+m) n \log n \right) \]

An algebraic approach, where the adjacency matrix \( A \) is "multiplied" with itself \( n \) times \( A^n \) under (addition and min)
\[
(a_1, a_2, a_3, \ldots, a_n) \cdot (b_1, b_2, \ldots, b_n)
\]
\[ \min \left( a_1 + b_1, a_2 + b_2, a_3 + b_3, \ldots, a_n + b_n \right) \]
Correct new of Dijkstra's shortest path algorithm

By induction on the order the vertices get added to $S$ (It adds the vertices in increasing order of the shortest)

$|V| = n$
$|E| = m$

Shortest paths found

$S = A, G, H$

$T = B, C, D, E, F, G, H, I$

Case 1
$x$ doesn't have the right value of shortest path
Handling - ve cycles in shortest path problems

Adding a large - ve number to all the weights would affect paths of different (edge) lengths differently.

\[ \text{Is there a path (cycle) connecting all vertices in the graph where every vertex is visited exactly once?)} \]

\[ \text{(Every tour visits every edge exactly once)} \]

\[ \rightarrow \text{Hamilton cycle} \]

\[ \text{No polynomial time algorithm is not known for H.C.} \]

\[ \rightarrow \text{as obviated & not possible) } \]
Prover
(Has the secret algorithm)

Verifier
(Must verify if the prover is correct)

Verifying H.C. is easy

TSP is not known to be easy

Is the given graph 3-colorable?
Opt version: find chromatic no.

NP: not polynomial
non deterministic
Non det. algorithm

- If the certificate is verified then answer is YES

- If certificate is not correct can we conclude NO?

If the given instance has a HC ⇒ there is a short proof/certificate

If there is no HC, then ??

```
  guess 1 → guess 2 → guess 3 → [Yes]

  guess 1 → [No]

  guess 1 → guess 2 → guess 3 → guess 4 → [Yes]
```

[Yes]
The class of decision problems for which there is a polynomial-time verification (when the answer is YES) is called Non-deterministic Polynomial Set or all graphs.

Polynomial in size of input

Polynomial Time: Is the class of problems that have an algorithm running in \( O(n^c) \) time for some integer \( c \) and input size \( n \).
If we can show that H.C. problem cannot (lower bound) have polynomial algorithm,

\[ P \neq NP \quad \text{i.e.} \quad P \subset NP \]
What if there is a polynomial time algorithm for \( H \cdot C \)?

What does it mean for a problem \( \Pi_1 \) to be reducible to problem \( \Pi_2 \)?

\[
\Pi_1 \leq \Pi_2
\]

Given an instance \( \tau \) of input size \( n \), we first run a mapping algorithm that takes, say \( g(n) \) time where \( g \) is a polynomial function and produces an instance \( f(\tau) \) [\( |\tau| = n \)].
We run the algorithm for $\Pi_2$ on the instance $f(x)$ that takes time $g(f(x))$; $g$ is some function corresponding to the running time of $\Pi_2$.

So the overall algorithm for $\Pi_1$ takes $g(n) + g(|f(x)|)$.

If $g$ is a polynomial function, then this is polynomial time.

Since $|f(x)| \leq g(n)$.

Reductibility is a relation

$$(\Pi_1, \Pi_2) \quad \Pi_1 \leq \Pi_2$$

polynomial-time reductibility is a special kind of reductibility, and we write $\Pi_1 \leq_{\text{poly}} \Pi_2$. 


If \( \pi_1 \leq_{poly} \pi_2 \)
and \( \pi_2 \leq_{poly} \pi_3 \)

\[ \therefore \pi_1 \leq_{poly} \pi_3 \] (by transitivity)

Since \( \pi \) a polynomial function of polynomial time.

\[ \text{NP-complete problem:} \]
\[ \pi \in \text{NPC} \text{ if } \exists \pi \text{ in class } NP \]

2. All problems \( P \in NP \) are reducible to \( \pi \) in polynomial time.

\[ \exists \text{ If } \pi \text{ in NPC and there is a poly-time algorithm for } \pi \text{ then } P = NP \]