NP-complete and NP-hard problems

- Transitivity of polynomial-time many-one reductions
- Concept of Completeness and hardness for a complexity class
- Definition of complexity class NP
  - NP-complete and NP-hard problems

$\Pi_1 \leq_p \Pi_2$ & $\Pi_2 \leq_p \Pi_3 \rightarrow \Pi_1 \leq_p \Pi_3$

- Let R_1 be the reduction used to prove $\Pi_1 \leq_p \Pi_2$
- Let R_2 be the reduction used to prove $\Pi_2 \leq_p \Pi_3$
- Let x be an input to Π_1
- Define $R_3(x)$ to be $R_2(R_1(x))$

Answer-preserving argument

- Because R_1 is a reduction between Π_1 and Π_2, we know that $R_1(x)$ is a yes input instance of Π_2 iff x is a yes input instance of Π_1
- Because R_2 is a reduction between Π_2 and Π_3, we know that $R_2(R_1(x))$ is a yes input instance of Π_3 iff $R_1(x)$ is a yes input instance of Π_2
- Applying transitivity of iff, we get that $R_3(x)$ is a yes input of Π_3 iff x is a yes input instance of Π_1

Polynomial-time Argument

- Let R_1 take time n^{c1}
- Let R_2 take time n^{c2}
- Let n be the size of x
- Then the R_1 call of R_3 takes time at most n^{c1}
- Furthermore, $R_1(x)$ has size at most $\max(n,n^{c1})$
- Therefore, the R_2 call of R_3 takes time at most $\max(n^{c2}, (n^{c1})^{c2}) = \max(n^{c2}, n^{c1\ c2})$
- In either case, the total time taken by R_3 is polynomial in n

NP-complete and NP-hard problems

- Transitivity of polynomial-time many-one reductions
- Concept of Completeness and hardness for a complexity class
- Definition of complexity class NP
  - NP-complete and NP-hard problems

Utility of Relative Classification Results

- Consider only a pair of problems Π_1 and Π_2
- What does $\Pi_1 \leq_p \Pi_2$ mean?
  - If Π_1 is not in P, then Π_2 is not in P
  - Intuitively, Π_2 is at least as hard as Π_1
- What does $\Pi_1 \leq_p \Pi_2$ and $\Pi_2 \leq_p \Pi_1$ mean?
  - If either Π_1 or Π_2 is not in P, then the other is not in P
  - Intuitively, these two problems are equivalent in difficulty
- In isolation, these results have relatively little impact unless you care about these two specific problems

Utility of Relative Classification Results cont'd

- Consider only a set of problems C
- What does "for all Π' ε C Π' $\leq_p$ Π mean?
  - If any problem in C is not in P, then Π is not in P.
  - If Π is in P, then all problems in C are in P.
  - Intuitively, Π is the hardest problem in C union {Π}
- What does "for all Π,Π' ε C Π' $\leq_p$ Π mean?
  - If any one of the problems in C is not in P, then they all are not in P.
  - If any one of the problems in C is in P, then they all are in P.
  - Intuitively, the problems in C are roughly equivalent in complexity.
- The importance of these results depends on the class C

Definition of C-hard and C-complete

- Let C be a set of problems
- C-hard definition
  - A problem Π is C-hard if for all $\Pi' \in C$ $\Pi' \leq_p \Pi$ holds.
  - Intuitively, Π is the hardest problem in C union $\{\Pi\}$
- C-complete
  - A problem Π is C-complete if
  - Π is C-hard and
  - Π is in C
  - That is, Π is in C and is the "hardest" problem in C (with respect to being in P)

Observations

- All C-complete problems are equivalent in difficulty with respect to being in P
- Proving a new problem Π is C-hard
  - If there is a known C-hard problem Π' (usually a C-complete problem), then we can prove Π is C-hard by showing that $\Pi' \leq_p \Pi$
    - This follows from transitivity of poly-time reductions
  - If there is no known C-hard problem Π', we require some method for proving that all problems in C polynomial-time many-one reduce to Π

NP-complete and NP-hard problems

- Transitivity of polynomial-time many-one reductions
- Concept of Completeness and hardness for a complexity class
- Definition of complexity class NP
  - NP-complete and NP-hard problems

Motivation for Complexity Class NP

- It includes many interesting problems
- It seems unlikely that P = NP
- We can show many interesting problems have the property of being NP-complete

Definition of NP-hard and NP-complete

- A problem Π is NP-hard if
  - for all $\Pi' \varepsilon$ NP $\Pi' \leq_p \Pi$ holds.
- A problem Π is NP-complete if
  - Π is NP-hard and
  - Π is in NP
- Proving a problem Π is NP-complete
  - Show Π is in NP (usually easy step)
  - Prove for all $\Pi' \varepsilon$ NP $\Pi' \leq_p \Pi$ holds.
    - Table method, simulation based method: Cook's Thm
    - *Show that $\Pi' \leq_p \Pi$ for some NP-hard problem Π'*

Importance of NP-completeness Importance of "Is P=NP" Question

- Practitioners view
  - There exist a large number of interesting and seemingly different problems which have been proven to be NP-complete
  - The P=NP question represents the question of whether or not all of these interesting and different problems belong to P
  - As the set of NP-complete problems grows, the question becomes more and more interesting

Importance of NP-completeness Importance of “Is P=NP” Question

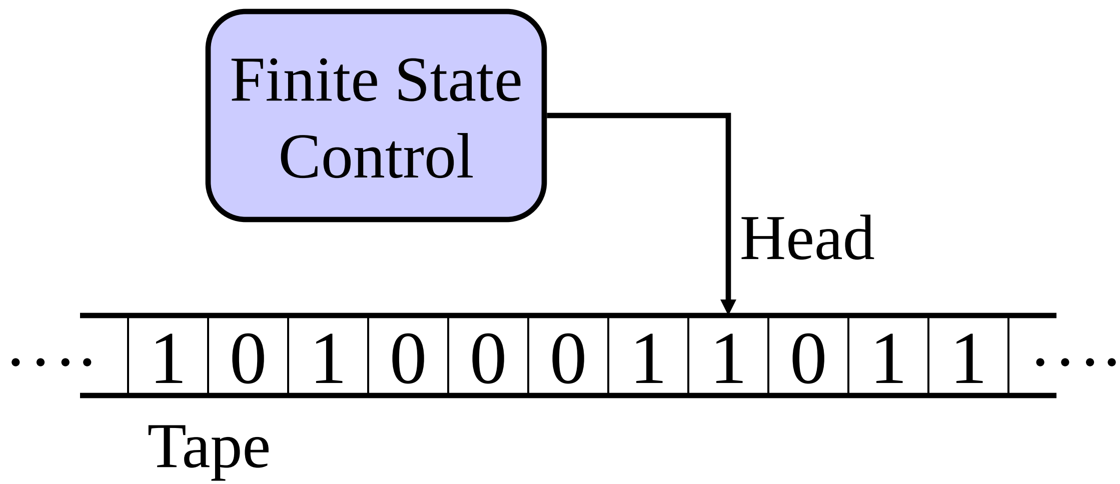
- Theoretician’s view
  - We will show that NP is exactly the set of problems which can be “verified” in polynomial time
  - Thus “Is P=NP?” can be rephrased as follows:
    - Is it true that any problem that can be “verified” in polynomial time can also be “solved” in polynomial time?
- Hardness Implications
  - It seems unlikely that all problems that can be verified in polynomial time also can be solved in polynomial time
  - If so, then P is not equal to NP
  - Thus, proving a problem to be NP-complete is a hardness result as such a problem will not be in P if P is not equal to NP.

Traditional definition of NP

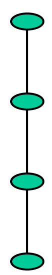
- Turing machine model of computation
  - Simple model where data is on an infinite capacity tape
  - Only operations are reading char stored in current tape cell, writing a char to current tape cell, moving tape head left or right one square
- Deterministic versus nondeterministic computation
  - Deterministic: At any point in time, next move is determined
  - Nondeterministic: At any point in time, several next moves are possible
- NP: Class of problems that can be solved by a nondeterminstic turing machine in polynomial time

Turing Machines

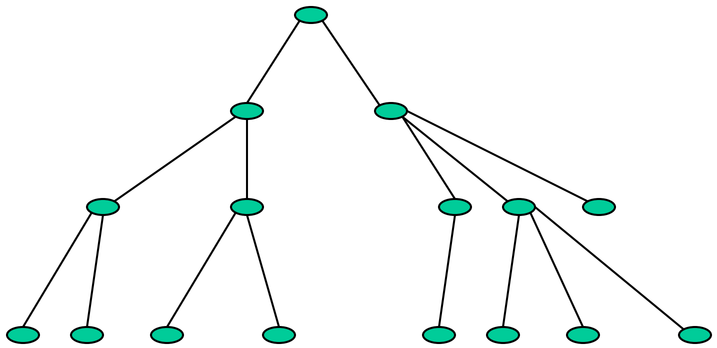
A Turing machine has a finite-state-control (its program), a two way infinite tape (its memory) and a read-write head (its program counter)



Nondeterministic Running Time



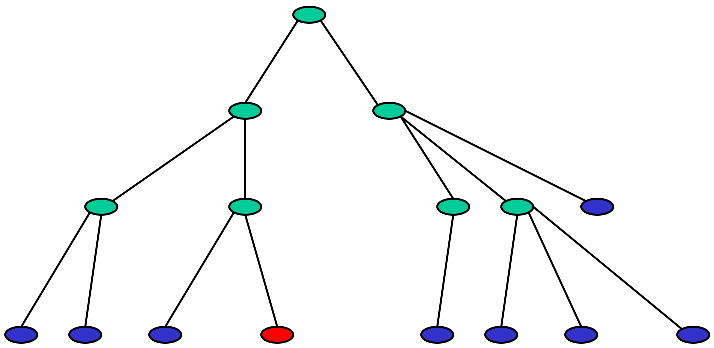
Deterministic Computation



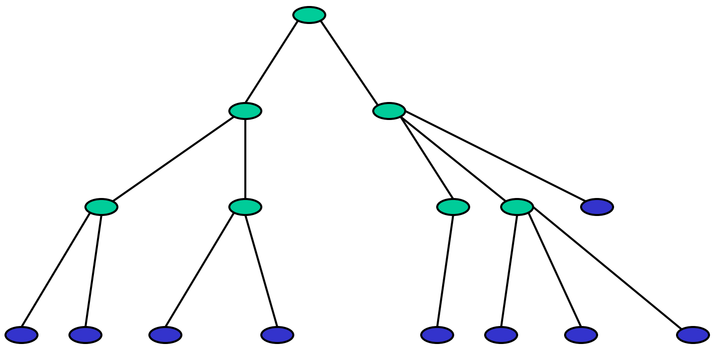
Nondeterministic Computation

- We measure running time by looking at height of computation tree, NOT number of nodes explored
- Both computation have same height 4 and thus same running time

ND computation returning yes



Yes Result



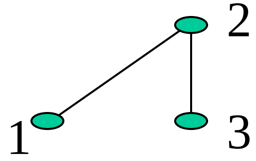
No Result

- If any leaf node returns yes, we consider the input to be a yes input.
- If all leaf nodes return no, then we consider the input to be a no input.

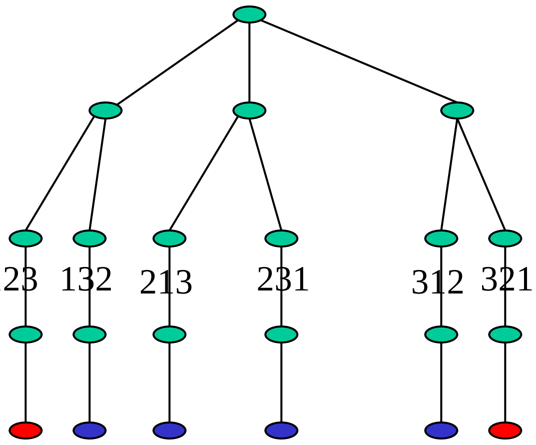
Showing a problem is in NP

- Hamiltonian Path
  - Input: Undirected graph G = (V,E)
  - Y/N Question: Does G contain a HP?
- Nondeterministic polynomial-time solution
  - Guess a hamiltonian path P (ordering of vertices)
    - V! possible orderings
    - For binary tree, V log V height to generate all guesses
  - Verify guessed ordering is correct
  - Return yes/no if ordering is actually a HP

Illustration

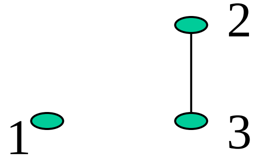


Yes input graph

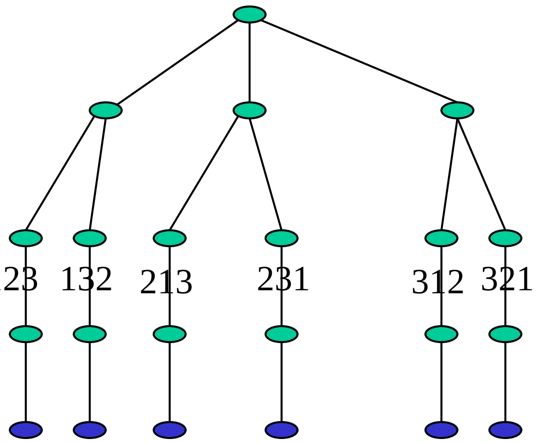


Guess Phase
Nondeterministic

Verify Phase
Deterministic



No input graph



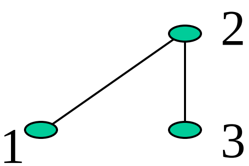
Guess Phase
Nondeterministic

Verify Phase
Deterministic

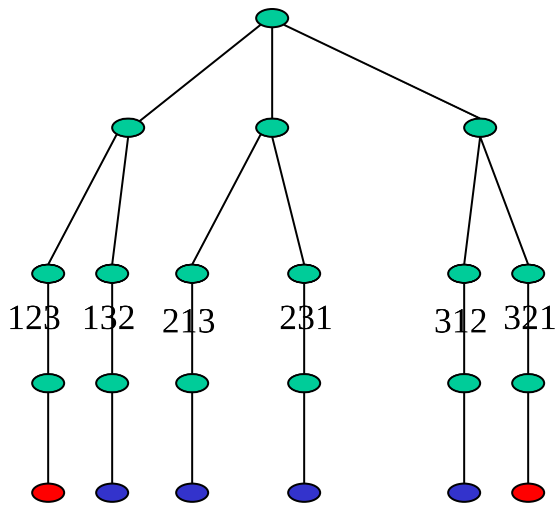
Alternate definition of NP

- Preliminary Definitions
  - Let Π be a decision problem
  - Let I be an input instance of Π
  - Let $Y(\Pi)$ be the set of yes input instances of Π
  - Let $N(\Pi)$ be the set of no input instances of Π
- Π belongs to NP iff
  - For any I ε $Y(\Pi)$, ***there exists*** a "certificate" [solution] C(I) such that a deterministic algorithm can verify I ε $Y(\Pi)$ in polynomial time with the help of C(I)
  - For any I ε $N(\Pi)$, no "certificate" [solution] C(I) will convince the algorithm that I ε $Y(\Pi)$.

Connection

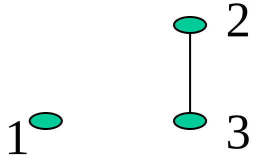


Yes input graph

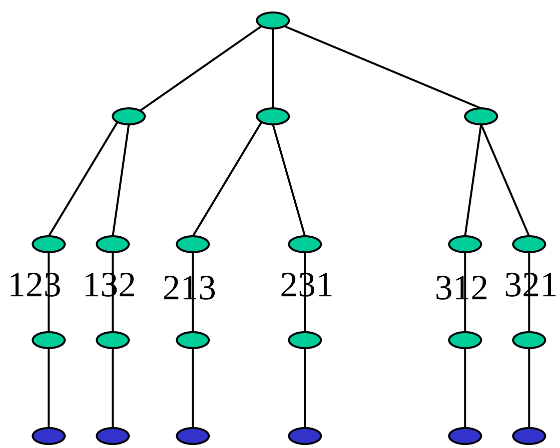


Guess Phase
Nondeterministic

Verify Phase
Deterministic



No input graph



Guess Phase
Nondeterministic

Verify Phase
Deterministic

- Certificate [Solution]
- A Hamiltonian Path
- $C(I_1)$: 123 or 321
- $C(I_2)$: none
- Verification Alg:
- Check for edge between all adjacent nodes in path

Example: Clique Problem

- Clique Problem
  - Input: Undirected graph $G = (V,E)$, integer k
  - Y/N Question: Does G contain a clique of size $\geq k$?
- Certificate
  - A clique C of size at least k
- Verification algorithm
  - Verify that all nodes in C are connected in E

Proving a problem is in NP

- You need to describe what the certificate C(I) will be for any input instance I
- You need to describe the verification algorithm
  - usually trivial
- You need to argue that all yes input instances and only yes input instances have an appropriate certificate C(I)
  - also usually trivial (typically do not require)

Example: Vertex Cover Problem

- Vertex Cover Problem
  - Input: Undirected graph G = (V,E), integer k
  - Y/N Question: Does G contain a vertex cover of size ≤ k?
    - Vertex cover: A set of vertices C such that for every edge (u,v) in E, either u is in C or v is in C (or both are in C)
- Certificate
  - A vertex cover C of size at most k
- Verification algorithm
  - Verify that all edges in E contain a node in C

Example: Satisfiability Problem

- Satisfiability Problem
  - Input: Set of variables X and set of clauses C over X
  - Y/N Question: Is there a satisfying truth assignment T for the variables in X such that all clauses in C are true?
- Certificate?
- Verification algorithm?

Example: Unsatisfiability Problem

- Unsatisfiability Problem
  - Input: Set of variables X and set of clauses C over X
  - Y/N Question: Is there no satisfying truth assignment T for the variables in X such that all clauses in C are true?
- Certificate?
- Verification algorithm?

Key recap

- Proving a problem Π is NP-complete
  - Show Π is in NP (usually easy step)
  - Prove for all $\Pi' \in NP$ $\Pi' \leq_p \Pi$ holds.
    - Assuming we have an NP-hard problem Π'
    - *Show that $\Pi' \leq_p \Pi$ for some NP-hard problem Π'*
- For this to work, we need a "first" NP-hard problem Π
  - Cook's Theorem and SAT