9c NP
Friday, February 21, 2020 2:16 PM

Goals:
- Feel slightly (as opposed to very) confused about NP

Questions: Memoization vs Tabulation, Example

Is there a satisfying assignment?

ORs ANDs

$(v_1 \vee v_2 \vee v_3) \wedge (v_1 \vee v_2 \vee v_4) \wedge (v_1 \vee v_3 \vee v_4) \wedge (v_1 \vee \neg v_2 \vee \neg v_3) \wedge$
$(v_1 \vee \neg v_2 \vee \neg v_4) \wedge (v_1 \vee v_3 \vee v_4) \wedge (\neg v_1 \vee v_3 \vee v_4) \wedge (v_1 \vee v_2 \vee v_4)$

A) Yes B) No

Is v_1=T, v_2=F, v_3=T, v_4=F a satisfying assignment?

$(v_1 \vee v_2 \vee v_3) \wedge (v_1 \vee v_2 \vee v_4) \wedge (v_1 \vee v_3 \vee v_4) \wedge (v_1 \vee \neg v_2 \vee \neg v_3) \wedge$
$(v_1 \vee \neg v_2 \vee \neg v_4) \wedge (v_1 \vee v_3 \vee v_4) \wedge (\neg v_1 \vee v_3 \vee v_4) \wedge (v_1 \vee v_2 \vee v_4)$

$\boxed{\text{A) Yes}}$ B) No

NP idea: A problem is in NP if checking whether a potential solution is valid is easy. * No comment on whether solving problem is difficult... might be hard to solve.

$$\boxed{\begin{array}{l}\text{Is there a satisfying assignment?}\\ (v_1 \vee v_2 \vee v_3) \wedge (v_1 \vee v_2 \vee v_4) \wedge (v_1 \vee v_3 \vee v_4) \wedge (v_1 \vee \neg v_2 \vee \neg v_3) \wedge \\ (v_1 \vee \neg v_2 \vee \neg v_4) \wedge (v_1 \vee v_3 \vee v_4) \wedge (\neg v_1 \vee v_3 \vee v_4) \wedge (v_1 \vee v_2 \vee v_4)\end{array}}$$

in NP — 3SAT (define precisely later)

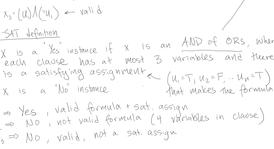
Contrast:

P idea: A problem is in P if finding a solution is easy.

Why do we care?
- Many real world problems are in P or NP (easy, hard)
- Useful for understanding difficulty of problem
  - Practical: Know when to stop finding algs.
  - Fundamental: What is computational power of computers
    - What are some problems different and others easy.

Easy? $\Rightarrow$ Polynomial Time $\Rightarrow O(n), O(n^2), O(n^{100}), O(n^{\log n})$
$\not\Rightarrow O(2^n)$ exponential time

Which Picture is Correct? Why? Place Problems?



1 Million Dollar

P = NP
P ≠ NP

P vs NP

$P \subseteq NP$ because if you can find a solution easily, must be able to check that it is valid easily.

3SAT

Example inputs:

$\Rightarrow X_1 = (v_1 \vee v_2 \vee v_3) \wedge (v_1 \vee v_2 \vee v_4) \wedge (v_1 \vee v_3 \vee v_4) \wedge (v_1 \vee \neg v_2 \vee \neg v_3) \wedge$
$(v_1 \vee \neg v_2 \vee \neg v_4) \wedge (v_1 \vee v_3 \vee v_4) \wedge \underbrace{(\neg v_1 \vee v_3 \vee v_4)}_{\text{clause}} \wedge \underbrace{(v_1 \vee v_2 \vee v_4)}_{\text{clause}}$ (ORs, ANDs)

$\Rightarrow X_2 = (v_1 \vee v_2 \vee v_3 \vee v_4) \leftarrow$ 4SAT

$\Rightarrow X_3 = (v_1) \wedge (v_1) \leftarrow$ valid

3-SAT definition
- X is a "Yes" instance if x is an AND of ORs, where each clause has at most 3 variables and there is a satisfying assignment $\leftarrow (v_1 = T, v_2 = F, \ldots v_n = T)$ that makes the formula True
- X is a "No" instance

$X_1 \Rightarrow$ Yes, valid formula + sat. assign
$X_2 \Rightarrow$ No, not valid formula (4 variables in clause)
$X_3 \Rightarrow$ No, valid, not a sat. assign

NP: Standard def

$$\boxed{\begin{array}{l}\text{NP = Nondeterministic Polynomial Time}\\ \text{A problem is in NP if}\\ \cdot \text{"Yes" Instances} \quad \cdot \text{"No" Instances}\\ \cdot \text{There is a polynomial time algorithm } M \text{ s.t.}\\ \quad -\text{If } x \text{ is a "Yes" instance} \rightarrow \exists y \text{ s.t. } M(x,y) = 1\\ \quad -\text{If } x \text{ is a "No" instance} \rightarrow \forall y \text{ s.t. } M(x,y) = 0\end{array}}$$

"verifier" = checker
"witness" = possible solution
$|x|$ = size of input

Goals:
- Learn how to prove a problem is in NP (P)
- Practice proving a problem is in NP (P)

Questions:
- AND/OR structure
- 2SAT, 4SAT? 3SAT $\in$ P, k5AT for any $k \geq 3$ k5AT $\in$ NP
- Yes-No part of definition
- P vs NP in real world
- NP-Hard, NP-Complete?
- Memoization?

3SAT $\in$ NP

Ex:

1. Describe M
Let $M(x,y)$ be the algorithm that checks
(a) If x is a valid 3-sat formula (AND of ORs, 3 variables/clause)
(b) If y is an assignment of T, F to each variable
(c) If each clause in x is true with y's assignment.
If all checks pass, return 1 Else: return 0

2. Check M satisfies NP def
- If x is "Yes" then let y be the satisfying assignment. $\Rightarrow M(x,y) = 1$ w/c will pass all tests
- If x is "No", then no y will cause 3rd clause to pass, or x is not valid $\rightarrow \forall y, M(x,y) = 0$

3. Check M's runtime
(a) $O(|x|)$
(b) $O(|x|)$ Seems like $O(|y|) = O(|x|)$ (# of variables)
(c) $O(|x|)$

$y = (v_1 = T, v_2 = F, v_3 = F, \ldots)$

M is polynomial time in the input size.

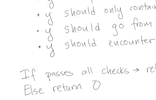
Thus 3SAT $\in$ NP

Hamiltonian Path

Given a graph G, vertices s,t, is there a path from s to t in G that hits each vertex exactly once?

X
- Yes: X is an adj matrix of a graph $G = (V,E)$, and vertices $s, t \in V$ and there is a path from s to t in G that goes through each vertex exactly once.
- No: Otherwise

ex:



Hamiltonian Path $\in$ NP

Ex: 1. Describe M

$M(x,y)$ checks if n = # of vertices $O(n^2)$
- x is (G,s,t)
- y is path of length $n-1$ $O(n)$
- y should only contain edges in G $O(n)$
- y should go from s to t $O(1)$
- y should encounter each vertex $O(n)$ $O(n^2)$

If passes all checks $\rightarrow$ return 1
Else return 0

2. Check M satisfies NP def

If x is "Yes"

If x is "No"

3. M's runtime

To Prove a Problem in P
- Write down pseudocode
- Prove correct
- Analyze runtime $\rightarrow$ polynomial in input size

For example: Knapsack
- runtime $O(nC)$

What is $|x|$ (input size) for Knapsack?

A) $O(\log(n)\log C)$ B) $O(\log(n) + C)$

$\boxed{\text{C) } O(n + \log(C))}$ D) $\cancel{O(n + C)}$

$|x| \approx \log(C)$ runtime $\approx C$ exponential! runtime

Knapsack $\in$ NP