

a) What is meant by P and NP Problems? Explain $P = NP$.

Recall:

$P = \{ \text{problems solvable in polynomial time} \}$ $\xrightarrow{\text{size } n}$

$NP = \{ \text{decision problems solvable in nondeterministic polynomial time} \}$ $\xrightarrow{n^{O(1)}}$

Answers
is YES
or NO

X is NP-complete if $X \in NP$ & X NP-hard.

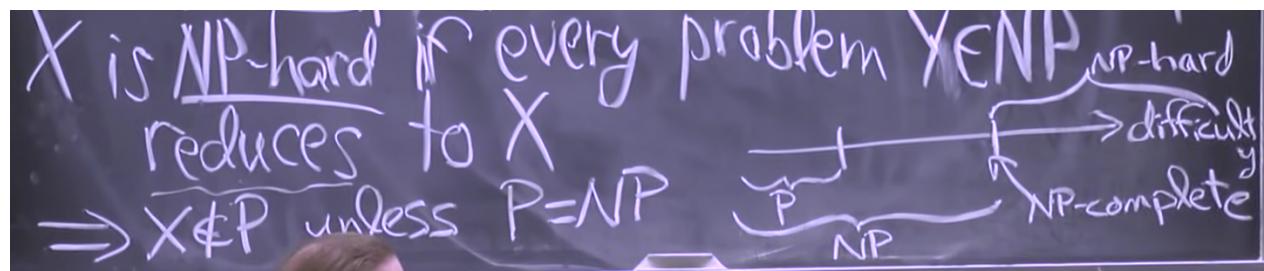
X is NP-hard if every problem $Y \in NP$ $\xrightarrow{\text{NP-hard}}$ reduces to X $\xrightarrow{\text{difficulty}}$

$\Rightarrow X \notin P$ unless $P = NP$ $\xrightarrow{\text{P}} \xrightarrow{\text{NP}} \xrightarrow{\text{NP-complete}}$

b) Why it is important to find approximate solutions for NP-Complete Problems.

ENP: guess $x_1 = T$ or F
guess $x_2 = T$ or F
check formula $\xrightarrow{T: \text{return YES}}$
 $\downarrow F: \text{return NO}$

c) What is the difference between NP-Hard and weakly NP-hard class problems?



Subset Sum: given n integers $A = \{a_1, a_2, \dots, a_n\}$ & target sum t ,
is there a subset $S \subseteq A$ such that

$$\sum_{i \in S} a_i = t$$

- weakly NP-hard

d) What is the 3-SAT problem?

3SAT: given Boolean formula of the form $(x_1 \vee x_3 \vee \bar{x}_6) \wedge (\bar{x}_2 \vee x_3 \vee \bar{x}_7) \wedge \dots$

x_1 \downarrow OR \downarrow NOT \downarrow AND \dots
 literals clause

Can you set the variables $x_1, x_2, \dots \rightarrow \{T, F\}$
 such that formula = T \hookrightarrow satisfying assignment

- e) What is meant by NP-complete problems? How can we prove a problem is NP-complete.
 Explain with an example.

X is NP-complete if $X \in \text{NP} \wedge X$ NP-hard.

How to prove X is NP-complete:

- ① $X \in \text{NP} \hookrightarrow$ nondet. algorithm
certificate + verifier
- ② reduce from known NP-complete
problem Y to X

TODAY: NP-completeness

→ Super Mario Bros.

3SAT

3-Dimensional
Matching

→ Subset Sum
↳ Partition
↳ Rectangle Packing

4-Partition

↳ Rectangle Packing
↳ Jigsaw Puzzles

f) What is Reduction?

Reduction from problem A \rightarrow problem B
= Poly-time algorithm converting
A inputs \rightarrow equivalent B inputs.

g) A problem that is solvable in time complexity of $T(n) = 3 * n^n$ and space complexity of $S(n) = n^2$ and it can be validated in $T(n) = 2^n$ time. Is it an NP-Complete or NP-Hard? Explain ?

It is NP hard problem as it is not solvable and verifiable in polynomial time.

- (d) A problem that is solvable in time complexity of $T(n) = 3 * n^n$ and space complexity of $S(n) = n^2$ and it can be validated in $T(n) = 2^n$ time. Is it a NP-Complete or NP-Hard? Explain

Sol: **NP-Hard** as validated in $T(n) = 2^n$ time.

Question #2

Consider the following APPROX-VERTEX-COVER algorithm. Proof that this algorithm is 2-approximation method for VERTEX-COVER. 10 Points

APPROX-VERTEX-COVER(G)

```
C = ∅;
E' = G.E;
while(E' ≠ ∅ ){
    Randomly choose a edge (u,v) in E', put u and v into C;
    Remove all the edges that covered by u or v from E'
}
Return C;
```

Proof:

- Assume a minimum vertex-cover is U^*
- A vertex-cover produced by **APPROX-VERTEX-COVER(G)** is U
- The edges chosen in **APPROX-VERTEX-COVER(G)** is A
- A vertex in U^* can only cover 1 edge in A
 - So $|U^*| \geq |A|$
- For each edge in A , there are 2 vertices in U
 - So $|U| = 2|A|$
- So $|U^*| \geq |U|/2$
- So $\frac{|U|}{|U^*|} \leq 2$

Question 3

10 Points

An Instance (X, F) of the set-covering problem consists of a finite set X and a family F of subset of X , such that every element of X belongs to at least one subset of F :

$$X = \bigcup_{S \in F} S$$

We say that a subset $S \in F$ covers all elements in X . Our goal is to find a minimum size subset $C \subseteq F$ whose members cover all of X .

$$X = \bigcup_{S \in C} S$$

Algorithm 1: GREEDY-SET-COVER (X, F)

```

1  $U \leftarrow X$ 
2  $C \leftarrow \emptyset$ 
3 While  $U \neq \emptyset$ 
4   do select an  $S \in F$  that maximizes  $|S \cap U|$ 
5    $U \leftarrow U - S$ 
6    $C \leftarrow C \cup \{S\}$ 
7 return  $C$ 
```

Consider each of the following words as a set of letters: {arid, dash, drain, heard, lost, nose, shun, slate, snare, thread}. Show which set cover GREEDY-SET-COVER produces, when we break ties in favor of the word that appears first in the dictionary

Since all of the words have no repeated letters, the first word selected will be the one that appears earliest on among those with the most letters, this is “thread”. Now, we look among the words that are left, seeing how many letters that aren’t already covered that they contain. Since “lost” has four letters that have not been mentioned yet, and it is first among those that do, that is the next one we select. The next one we pick is “drain” because it has two unmentioned letters. This only leave “shun” having any unmentioned letters, so we pick that, completing our set. So, the final set of words in our cover is {*thread, lost, drain, shun*}.

Question 4:

20 Points

Consider following points in 2D

(6,2), (9,5), (-2,2), (-3,4), (-8,8), (-10,4), (-10,3), (-8,-6), (-4,-4), (6,4), (6,-6), (-6,-10), (8,0)

Find the smallest convex set containing all the points using Package Wrap (Jarvis March) and Graham Scan (Show all iterations)

- A) Jarvis March: The points at the hull of convex in ordered form are: $(-6, -10) \rightarrow (6, -6) \rightarrow (8, 0) \rightarrow (9, 5) \rightarrow (-8, 8) \rightarrow (-10, 4) \rightarrow (-10, 3) \rightarrow (-8, -6)$.
- B) Graham Scan: The points are as same as the Jarvis March but the deleted points are: $(6,2) \rightarrow (6,4) \rightarrow (-2,2) \rightarrow (-4,-4) \rightarrow (-3,4)$. The sorted points are: $(-6, -10) \rightarrow (6, -6) \rightarrow (8, 0) \rightarrow (9, 5) \rightarrow (6, 2) \rightarrow (6, 4) \rightarrow (-2, 2) \rightarrow (-4, -4) \rightarrow (-3, 4) \rightarrow (-8, 8) \rightarrow (-10, 4) \rightarrow (-10, 3) \rightarrow (-8, -6)$.

