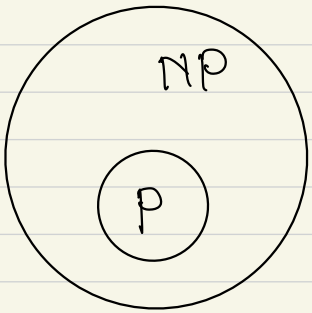
NP-Completeness

a) Argue that $P \subseteq NP$



P = Deterministic and polynomially solvable

NP = Non deterministic but polynomial

So, $P \subseteq NP$

b) Formally define NP-Hard and NP-Complete problem.

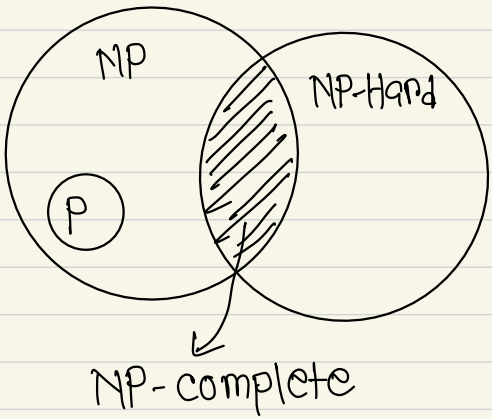
NP-Hard

A problem X is NP-hard if every problem Y in NP can be reduced to X in polynomial time

NP Complete

2 conditions

1. X is in NP
2. X is NP-Hard



Answer the following questions:

i. Prove that 2-CNF SAT is in P.

1)

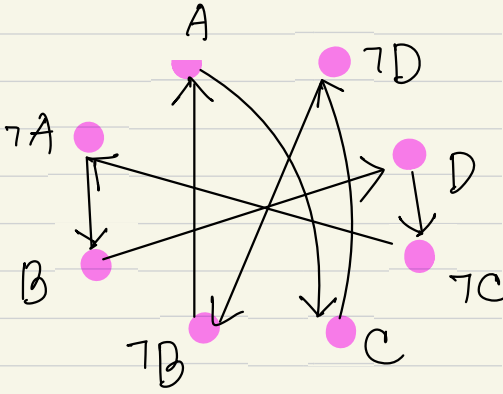
$$F = (A \vee B) \wedge (\bar{A} \vee C) \wedge (B \vee D) \wedge (\bar{C} \vee \bar{D})$$

Step 1 Create 2 vertices for each variable A and $\neg A$

Step 2 Transform each clause into 2 conditional statements.
To do this we add directed edges

$(A \vee B)$: $\bar{A} \to B$, $\bar{B} \to A$
$(\bar{A} \vee C)$: $A \to C$, $\bar{C} \to \bar{A}$
$(B \vee D)$: $B \to D$, $\bar{D} \to \bar{B}$
$(\bar{C} \vee \bar{D})$: $C \to \bar{D}$, $D \to \bar{C}$

Directed graph:



Step 3
Look for Strongly Connected Components.

ii. If 2-CNF SAT is solvable, why is 3-CNF SAT not assumed to be solved?

3-SAT is NP-complete.

If 3-SAT was solved in polynomial time, every NP problem could also be solved in polynomial

Effectively proving, P=NP

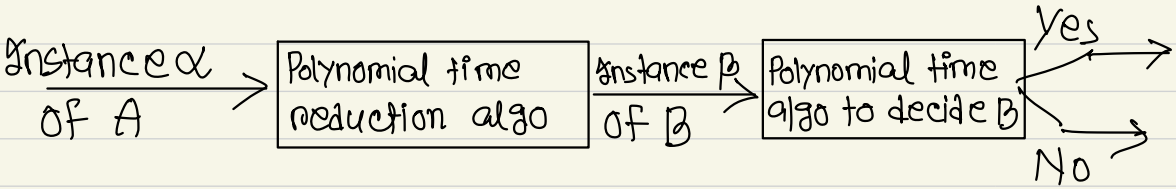
But not solved yet

Briefly describe Reduction and Reducibility in the context of NP-Completeness.

→ Want to solve A in polynimal time
→ Have B that can be solved " "

→ Transform any input α in A to any input β in B

Assumptions
1) Transformation takes linear time
11) The answers are the same.



c) Prove (trivially) that Travelling Salesman Problem (TSP) and Boolean-SAT are NP-Complete.

⇒ TSP can be reduced to Hamiltonian cycle which is known to be NP complete.

So, TSP is NP-complete

⇒ Boolean SAT can be reduced to 0/1 knapsack which is np-complete.

i. Knowing Hamiltonian Circuit problem is NPC, confirm TSP problem is also NPC.
ii. Name two decision problems and two optimization problems that are NPC.

i) 2 conditions

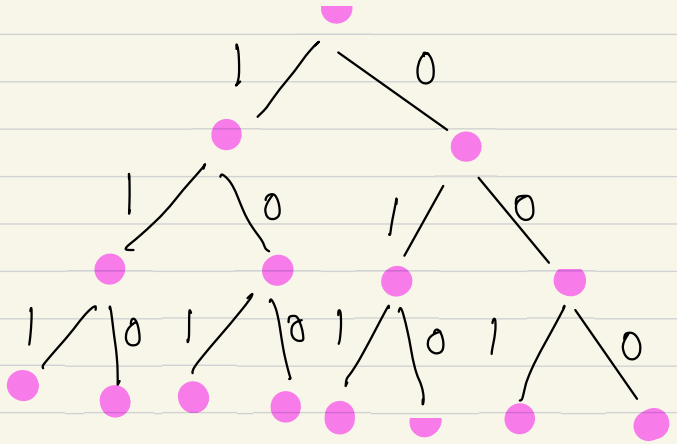
1. TSP is in NP: solution can be verified in polynomial time
2. TSP is NP-Hard
   - → Reduction from Hamiltonian circuit to TSP (Construct weighted graph G')
   - → Verification of reduction in polynomial time

All conditions satisfied. So, NP-complete

ii) Decision 1) SAT 2) Clique problem

Optimization 1) 0/1 knapsack 2) Graph coloring

b) Reduce 0-1 Knapsack problem to CNF-Satisfiability problem.



What is reducibility of a problem? Explain its usage in deciding tractability of a problem.

→ Reducability

Conversion to known polynomial-time algorithm.

→ Based on computational complexity we decide if tractable or untractable.

a) What is traveling-salesman problem? Prove that the traveling-salesman problem is NP-complete.

Shortest route betw a set of points and locations.

Reduction of Hamiltonian Circuit to TSP

1. For each pair of vertices in G, construct G' such that,
   a. $w(u,v) = 1$ if existing edge in G
   b. $w(u,v) = 2$, if no edge
2. Set bound $B = |V|$

So, TSP is NP-complete

Approximation Algo

In graph theory, a vertex cover of a graph is a set of vertices that includes at least one endpoint of every edge of the graph. Finding minimum vertex cover is NP-Complete.

i. Write a 2-approximation algorithm for Minimum-Vertex-Cover problem with its proof.

ii. Can you think of any improvement of the 2-approximation algorithm for Minimum-Vertex-Cover?

1) APPROX-VERTEX-COVER(G)

```
C = ∅
E' = G.E          // Edge set of graph
while E' ≠ ∅
    let (u,v) be an arbitrary edge in E'
    C = C ∪ {u,v}
    remove from E' every edge incident on u or v
return C
```

Time complexity: $O(V+E)$

11) GREEDY_VERTEX_COVER(G):

```
V = ∅
let L be a list of all leaves in G
while V ≠ 0
  if V==1 or 0
    return V'

  let v be a leaf and (u,v) it's incident edge
  V = V - v
  V' = V' ∪ u
  for each edge incident on u:
    if dw == 1
      L.append()
    E = E - {u,w}

  V = V - u
```

Time complexity: $O(V+E)$

Greedy-Set-Cover is a polynomial-time $\rho(n) = H(\max\{|S|: S \in F\})$ approximation algorithm. Explain the complexity in plain language. You do not need to show any proof.

$|X|$ is the number of vertices $= n$
$|F|$ is the number of subsets $= m$

→ Each iteration scans through subsets to find one with max cover
This takes $n \times m$ time

→ The loop must cover all subsets or all vertices.
So, it is bounded by $\min(n, m)$

Total complexity: $O\{nm * \min(n, m)\}$

Prove that randomized MAX-3-CNF algorithm is an ***8/7 -approximation*** algorithm.

n variables and m clauses

set each variable to 1 with $Pr = 1/2$

0 " $" = 1/2$

$Y_i = 1\{\text{clause } i \text{ is satisfied}\}$

Clause i is satisfied if any variable is set to 1

not " " all variables are " " 0

$Pr\{\text{clause not satisfied}\} = (1/2)^3 = 1/8$

$Pr\{\text{clause i is satisfied}\} = 1 - 1/8 = 7/8$

$\therefore E[Y_i] = 7/8$

Let Y = # of total satisfied clauses

$Y = Y_1 + Y_2 + \cdots + Y_m$

$\therefore E[Y] = E\left[\sum_{1}^{m} Y_i\right] = \sum_{1}^{m} E[Y_i] = 7m/8$

$$\text{Approximation ratio} = \frac{m}{7m/8} = 8/7$$

[m is the upper bound on satisfied clauses]

Provide a polynomial time 2-approximation algorithm for traveling salesman problem with triangle inequality. Comment on its complexity also.

APPROX-TSP-TOUR (G, c):
    select a vertex $r \in G$ to be root
    compute an MST from root r

    let H be the list of vertices ordered on pre-order walk
    return the cycle H

Time complexity : $O(V^2)$

→ Create MST for each vertex
→ Traverse MST with every vertex to get optimal path

iii. Why are we interested in approximation algorithms?

a. Unsolvable nature of NP-Hard problems

b. Provides near-optimal solutions

c. Resource constraints

d. Scalability

Trees

How does B+ Tree maintain balance?

Order m:
Each node except root and leaves can have $\lceil m/2 \rceil \rightarrow m$ children

Root
can have 2 to m children

Main operations
i) Splitting : More than capacity? Split into 2

ii) Merging : Less than capacity? Merge with sibling

i. What is a balanced tree? Write two applications of balanced tree.
ii. Why memory based balanced tree could not be used in disk based searching/indexing?
iii. How does Red-Black Tree maintain the balance in the tree?
iv. How does B-Tree maintain the balance in the tree?

I) Height → log n

Height of left subtree - height of right subtree ≤ 1

a. Database indexing b. Memory management

II) Large volume of data
Primary memory very expensive
Inefficient disk I/O

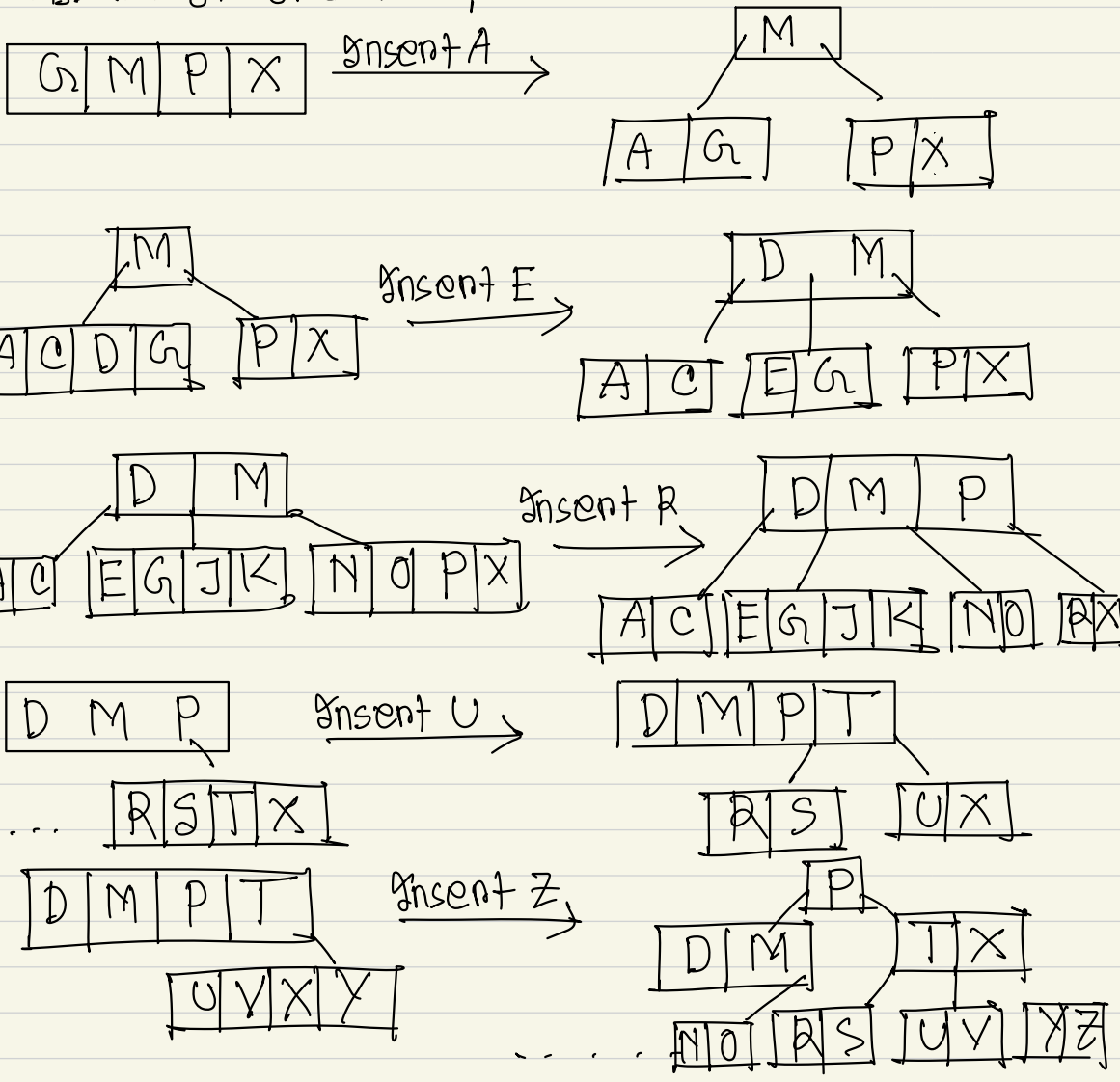
III) Left and right rotate
Decreases height of tree
Moves larger subtrees up

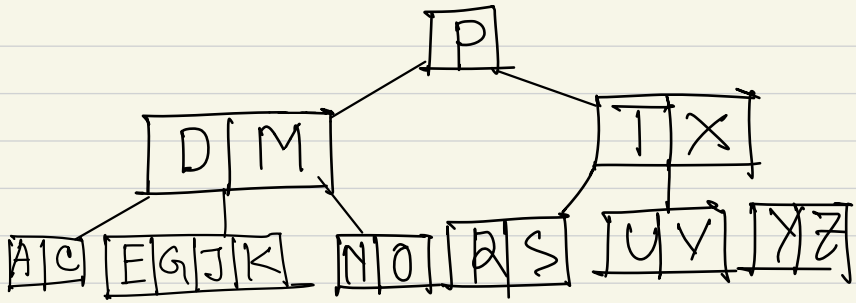
IV) Maintains min number of keys

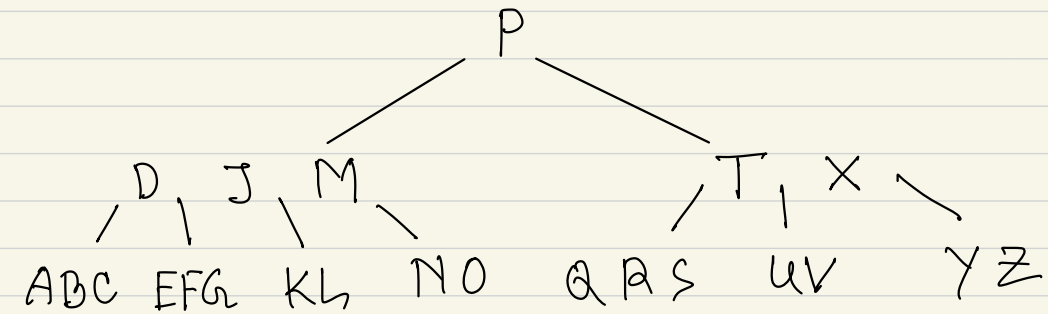
Every node except root must contain t-1 keys.
t = minimum degree

Insert the following keys in a B-tree (assume t is 3):
G M P X A C D E J K N O R S T U V Y Z B Q L F

Assuming, order = 4

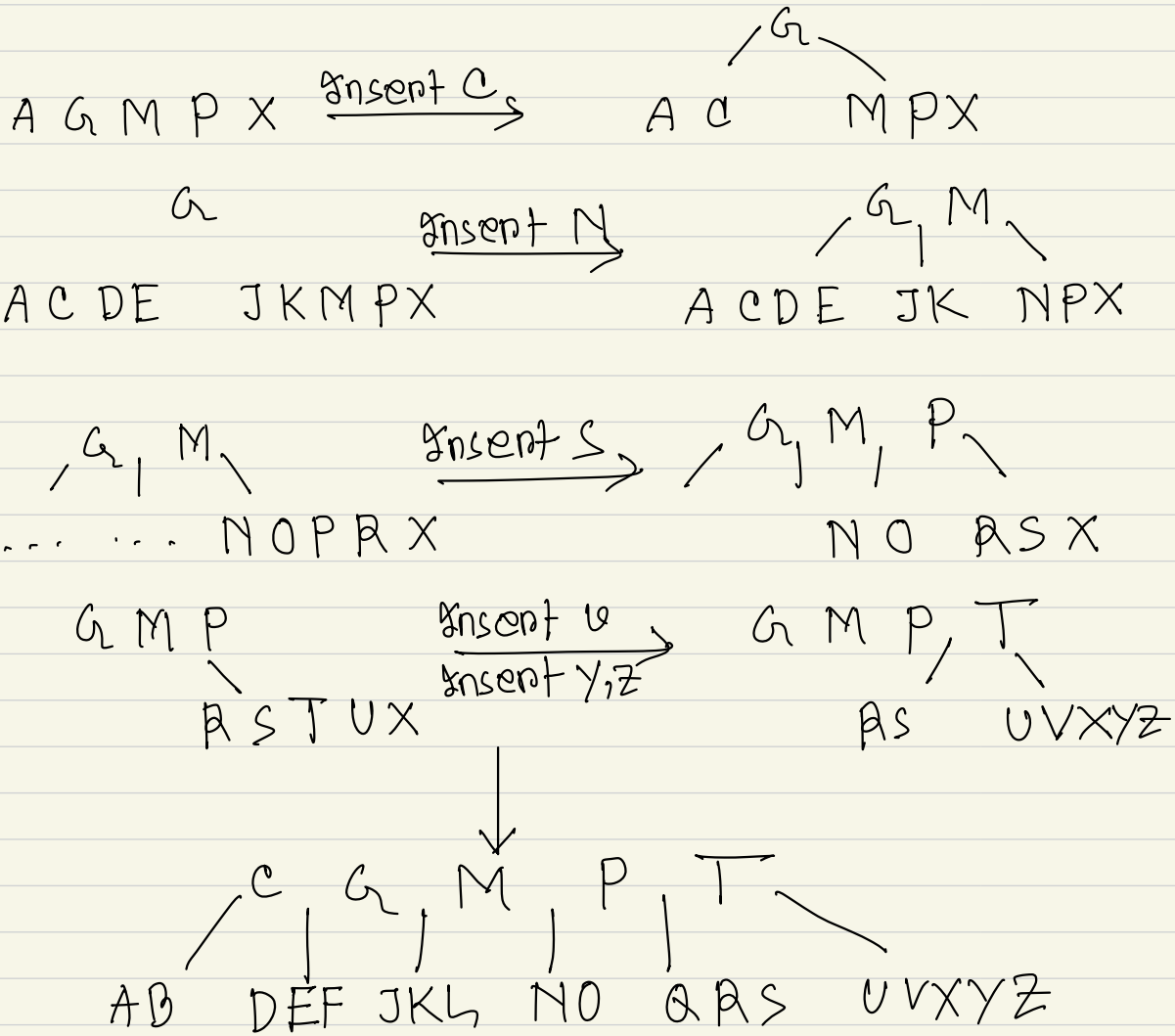



P
D M
T X
A C
E G J K
N O
P S
U V
Y Z


P
D J M
T X
ABC
EFG
KL
NO
QRS
UV
YZ

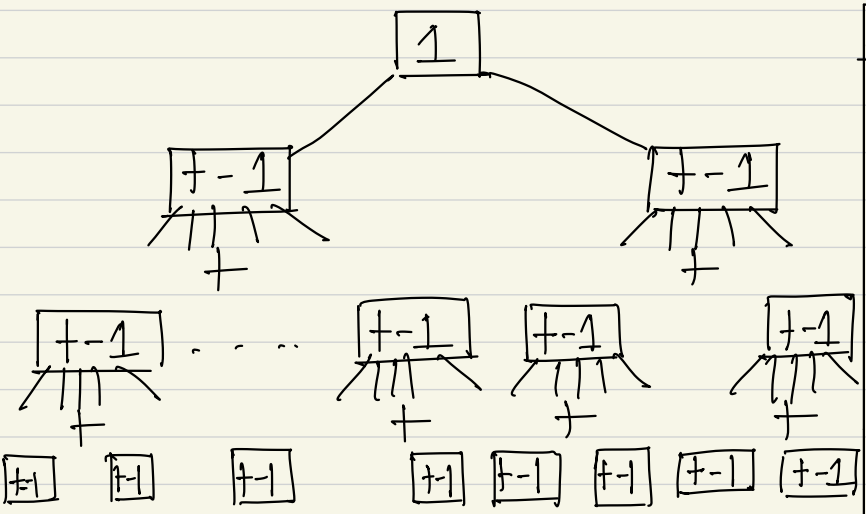
Insert the following keys in a B-tree (assume t is 3):
G M P X A C D E J K N O R S T U V Y Z B Q L F

t = 3, Max nodes = $2t-1 = 5$, children = $2t = 6$



Prove that if $n \geq 1$, then the height h for any n-key B-tree T with minimum degree $t \geq 2$, is

$$h \leq \log_t((n+1)/2)$$



depth	# of nodes
0	1
1	2
2	$2t$
3	$2t^2$

$$n \geq 1 + (t-1)\sum_{i=1}^{h} 2t^{i-1}$$

$$= 1 + 2(t-1)\left(\frac{t^h - 1}{t-1}\right)$$

$$= 2t^h - 1$$

$$\Rightarrow t^h \leq (n+1)/2$$

$$\Rightarrow h \leq \log_t((n+1)/2)$$

[proved]

Greedy

Discuss the greedy heuristic in Dijkstra's single source shortest path algorithm. Why does the heuristic work?

Smallest tentative weight

Works because non-negative edge weights

i. Greedy and Dynamic Programming are applied to problems with similar properties; what are those properties?

ii. How does Johnson's algorithm use Dijkstra Algorithm to solve all pair shortest path?

iii. Can Johnson's Algorithm solve shortest path problem with negative loops in the graph? Explain your answer.

I) a. Optimal substructure
b. Overlapping subproblems

II) a. Reweight the edges using bellman-ford

b. Apply dijakstra's algorithm on every vertex to find all pairs shortest path

c. Adjust edge weights back to previous using inverse formula

III) No
As underlying algorithms are SSSP

Write an algorithm to solve the fractional knapsack problem in $O(n)$ time.

```
fractional-knapsack (W, arr)

    Sort array based on profit/weight ratio

    for items in arr
      if item.weight <= W:
        W -= item.weight
        res += item.profit

      else
        res += item.profit * W/item.weight

    return res
```

Write the optimal substructure property of Optimal Prefix Coding problem. What is the greedy heuristic applied by Huffman Coding Algorithm to find the optimal prefix coding?

Optimal substructure

Breaking down the problem into smaller subproblems and solving them will result in optimal solution

Greedy heuristic

I) Merge 2 least frequent nodes at every step

II) Recursively applying until single tree is formed

i. What is the greedy heuristic for Dijkstra Algorithm? Why does the heuristic work?

ii. What is the greedy heuristic for Activity Selection problem? Why does the heuristic work?

iii. What is the graph property that Bellman Algorithm exploit? Is it a greedy or a dynamic programming algorithm?

II) Heuristic Sort by increasing finishing time

Choosing activity that finishes earliest, gives the option to choose as many activities as possible.

III) Relaxation property

Iterative relax all edges n-1 times to improve shortest path estimates

DP

Suppose there are cells in a rectangular mining field. Each cell may contain gold as reward, or may have bug that reduces the reward earned. We want to find the minimum number of squares in the mining field that will provide the maximum possible reward. Devise an algorithm or pseudo code for the purpose using greedy or dynamic programming technique.

DP

Optimal substructure

$dp[i][j] = \max(dp[i-1][j], dp[i][j-1])$

Overlapping subproblems

Solving $(i-1, j)$ and $(i, j-1)$ before solving (i, j)

DAG

ii. Why does longest simple path finding problem not have a dynamic programming solution like a shortest path finding problem?

lack of optimal substructure

Including longest path might lead to cycles or revisiting nodes

NP-hard problem

b) What do you understand by topological sorting? What is the role of topological sorting in finding shortest paths in DAG?

Topological sort = Linear ordering vertices in DAG

Algorithm,

I) Perform topological sorting

II) Initialize distances to -INF

III) Relax edges by updating distances based on topological order

Blockchain

a) Briefly describe blockchain and its application domains.

i) Supply chain management
ii) Healthcare
iii) Voting systems
iv) Digital identity verification

b) Ethereum has recently switched from Proof of Work (PoW) to Proof of Stake (PoS) consensus mechanism for creating Blocks though Bitcoin is still continuing with PoW. Describe the PoW and PoS with their pros and cons.

PoW

Miners solve complex cryptographical puzzles to validate and add new blocks

Pros	Cons
Security	Energy intensive
Decentralization	Scalability

PoS

Consensus mechanism where validators add and validate new blocks based on their cryptocurrency stake

Pros	Cons
Energy Efficiency	Security concern
Scalability	Wealth accumulation
	51% attack using high stake

c) Answer the following questions in short:

i. How are Nonce calculated in parallel using GPU even though a mining node proposes only one Block in one competition round of PoW?

ii. When you add your personal GPU to a mining node in Bitcoin, what are the tasks that you perform and what reward do you get in return?

iii. Why do we not ask solution to any NP problem instance as a challenge in PoW?

I) Despite testing many nonces in parallel, mining node only propoes one block

II) a. Install mining comptabile software
b. Utilize GPU for mining tasks
c. Hash computation iterating different nonces
d. Submission of hash

Reward
a. Transaction fees
b. Crypto generated from protocol

III) Too computationally extensive compared to matching nonces

i. Write two possible applications of Block Chain other than its use in crypto-currency.
ii. What are the applications of Merkle Tree?
iii. How can block chain be hacked?

ii) a. torrents
b. Blockchain
c. Scalable databases (Cassandra)
d. CA for certifying digital signatures

iii) 51% attack in PoS systems

b) How is double spending restricted in a Block Chain-based crypto-currency?

Through consensus mechanisms and decentralized nature.

i) PoW
ii) PoS
iii) PBFT

e) Explain the role of mining in Block Chain-based crypto-currency. How is the Proof of Work (PoW) ensured?

→ Miners compete to validate transactions and create new blocks

→ Miners must solve a complex cryptographical puzzle

→ Puzzle involves finding a nonce which combined with blocks data is passed through cryptographic function that produces a hash

→ Once miner finds a valid hash, they broadcast the new block to the network

→ Other nodes verify the block and its tx's

Randomized Shit

a) Prove that any comparison sort requires $\Omega(n \lg n)$ comparisons in the worst case.

Information theory go brrrr ----

b) Prove that the expected running time of a randomized quicksort is $O(n \lg n)$

z_i = i^{th} smallest element of $A[1 \cdots n]$

z_{ij} = set of elements between z_i, z_j including them

$X_{ij} = I\{z_i \text{ compared to } z_j\}$

X_{ij} = Indicator random variable for the event that i^{th} and j^{th} smallest are compared

Since each pair of elements are compared at most once,

$$X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}$$

Expected number of comparisons,

$$E[X] = \sum\sum E[X_{ij}]$$

$$= \sum\sum Pr[z_i \text{ compared to } z_j]$$

z_i, z_j will only be compared iff the first element of z_{ij} is picked as the pivot element

$Pr[z_i \text{ or } z_j \text{ is the first pivot chosen from } z_{ij}]$

$= Pr[z_i \text{ being picked}] + Pr[z_j \text{ being picked}]$

$= \frac{1}{j-i+1} + \frac{1}{j-i+1} = \frac{2}{j-i+1}$

$$E[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} = \sum_{i=1}^{n-1} \sum_{K=1}^{n-i} \frac{2}{K+1} < \sum_{i=1}^{n-1} \sum_{K=1}^{n} \frac{2}{K}$$

$$= \sum_{i=1}^{n-1} O(\log n)$$

$$= O(n \log n)$$

c) Write Randomized-Select algorithm to find any order statistics. Prove that expected running time of Randomized-Select algorithm to find any order statistics (particularly the median) is $O(n)$.

```
RANDOMIZED-SELECT(A, p, r, i)
1  if p == r
2       return A[p]
3  q = RANDOMIZED-PARTITION(A, p, r)
4  k = q − p + 1
5  if i == k            // the pivot value is the answer
6       return A[q]
7  elseif i < k
8       return RANDOMIZED-SELECT(A, p, q − 1, i)
9  else return RANDOMIZED-SELECT(A, q + 1, r, i − k)
```

Indicator random variable

$E[X_k] = 1/n$

Recurrence relation

$$T(n) = \sum_{k=1}^{n} X_k \cdot T(\max(k-1, n-k)) + O(n)$$

Expected calculation

$$E[T(n)] = \sum_{k=1}^{n} E[X_k] \cdot E[T(\max(k-1, n-k))] + O(n)$$

$$= \sum \frac{1}{n} E[T(\max(k-1, n-k))] + O(n)$$

$$E[T(n)] \leq \sum_{k=\lceil n/2 \rceil}^{n} 2/n \, E[T(k)] + O(n)$$

Asymptotic analysis

$T(n) = O(1)$ for $n < c$ for some constant c

Non recursive component bounded by an

$E[T(n)] = O(n)$ by induction

Master's Theorem

a) Write the rationale for the three cases of master method.

⇒ There are a total of 3 cases in this theorem

1) If the problem gets easier ⇒ The root is hardest

⇒ $T(n) = \theta(\text{root work})$

⇒ $T(n) = \theta(c(n))$

2) If the problem gets harder ⇒ The leafs are hardest

⇒ $T(n) = \theta(n^{\log_b(a)})$

3) If the hardness is evenly distributed,

⇒ All levels are relevant

⇒ $T(n) = \text{height} * (\text{same level})$

⇒ $T(n) = \log_b(n) . (\text{same level})$

$= \log_b n . \theta(n^{\log_b(a)})$

b) Given a regular recursion: $T(n) = aT(n/b) + f(n)$

If $f(n) = O(n^{\log_b a - \epsilon})$, prove that $T(n) = \theta(n^{\log_b a})$.

$$g(n) = O\left(\sum_{j=0}^{\log_b n - 1} a^j (n/b^j)^{\log_b a - \varepsilon}\right)$$

$$= O\left(n^{\log_b a - \varepsilon} \cdot \sum_{j=0}^{\log_b n - 1} \left(\frac{a}{b^{\log_b a - \varepsilon}}\right)^j\right)$$

$$= O\left(n^{\log_b a - \varepsilon} \sum_{j=0}^{\log_b n - 1} \left(\frac{a}{a \cdot b^{-\varepsilon}}\right)^j\right)$$

$$= O\left[n^{\log_b a - \varepsilon} \sum_{j=0}^{\log_b n - 1} (b^\varepsilon)^j\right]$$

$$= O\left[n^{\log_b a - \varepsilon} \cdot \frac{(b^\varepsilon)^{\log_b n} - 1}{b^\varepsilon - 1}\right] \quad \frac{x^{n+1} - 1}{x - 1}$$

$$= O\left[n^{\log_b a - \varepsilon} \cdot \frac{n^\varepsilon - 1}{b^\varepsilon - 1}\right] \quad \text{(keeping others constant)}$$

$$= O[n^{\log_b a}]$$

$$T(n) = \theta(n^{\log_b a}) + g(n)$$

b) Solve the following recurrence using master method and using recursion tree

$T(n) \leq T(2n/3) + \Theta(1)$

$b = 3/2, \; a = 1$

$\log_b a = \log_{3/2} 1 = 0$

$f(n) = \Theta(1) = \Theta(n^0)$, so, $c = 0$

Case 2 applies,

$T(n) = \Theta(n^0 \log n) = \Theta(\log n)$

Recursion tree

I) At each level, size of the problem reduces by a factor 2/3

II) Cost per level is constant $\Theta(1)$

III) Number of levels $\log_{3/2} n$

$T(n) = \#$ of levels $\times$ cost per level $= \Theta(\log_{3/2} n)$

$\log_{3/2} = \frac{\log_2 n}{\log_2 3/2}$ So, $T(n) = \Theta(\log n)$