① Define NP-Hard and NP-complete Pblms.

NP-hard.

→ "polynomial Time Reduction"

Consider a decision pblm A, which is like to solve in polynomial time.

Consider another pblm B, which is having a polynomial time algorithm.

Suppose that we have a procedure that transform any instance α of A into some instances β to B. with following characteristics.

1) Transformation takes polynomial time.
2) The answers are same.

Such procedures is called polynomial time reduction

class NP-complete

If the prblm is NP as well as NP-hard, then that prblm is NP complete

Eg: CIRCUIT SAT pblm.

2. Prove that CLIQUE pblm is NP-complete.

step 1: write a poly. polynomial time verification algorithm to prove that the given pblm is NP

Algorithm: Let G = (V, E)

→ Test whether V is a set of K vertice

→ check whether for each pair (u,v) ∈ V

This algorithm will execute in polynomial time ∴ CLIQUE problm is a NP problm.

Step 2: write a polynomial time reduction algorithm from 3-CNF-SAT problm to CLIQUE problm.

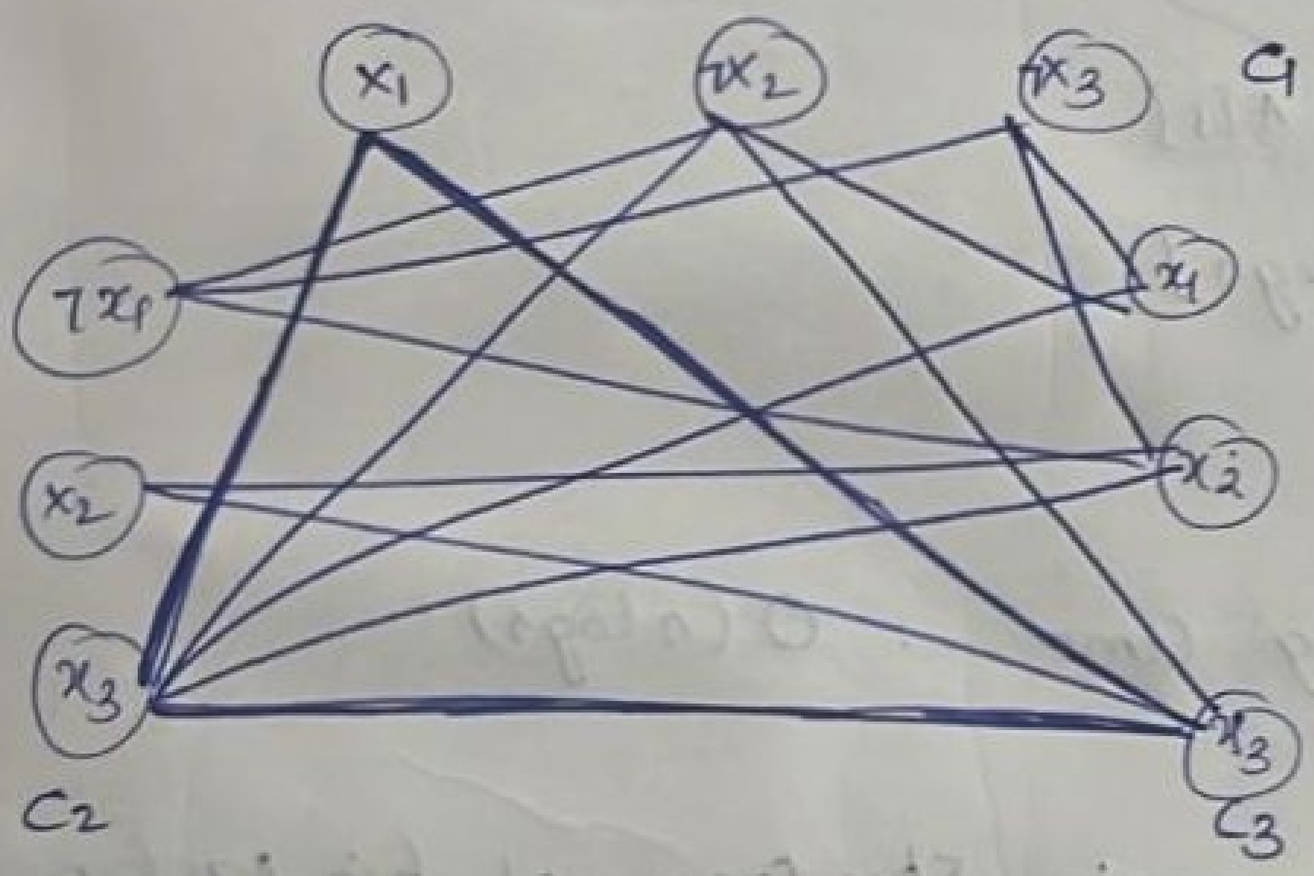
Algorithm

→ Let $\phi = C_1 \wedge C_2, \ldots \wedge C_k$ be a boolean ~~function~~ formula in 3 CNF

→ Each clause C_r has exactly 3 distant literals.

→ construct a graph G such that ϕ is satisfiable

Eg: $\phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$



→ G can easily be constructed from ϕ in polynomial t

→ So CLIQUE problm is NP hard.

Conclusion: CLIQUE problm is NP and NP hard - so it is complete.

3. write randomized quicksort algorithm and perform expected running time analysis.

Algorithm

① RandomizedQuicksort (A, low, high):

if low < high:

pivot ← RandomizedPartition (A, low, high)

call RandomizedQuicksort (A, low, pivot - 1)

call RandomizedQuicksort (A, pivot + 1, high)

RandomizedPartition (A, low, high)

swap A[k] with A[high]

Set pivot ← A[high]

Initialize i ← low - 1

For j from low to high - 1

if A[j] ≤ pivot

increment i

swap A[i] and A[j]

swap A[i+1] and A[high]

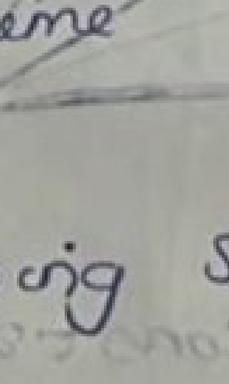
return i+1

∴ Final expected Running time : O(n log n)

④ Explain the first - fit decreasing strategy of bin packing algorithm

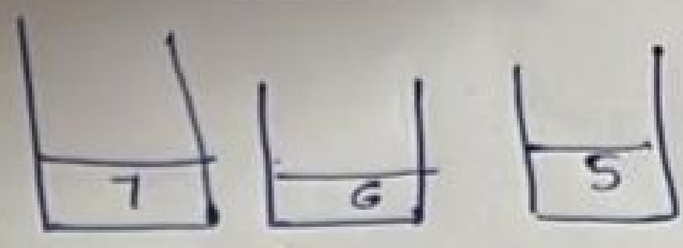
Arrange the items in descending order of the weight : {7, 6, 5, 5, 5, 4, 2, 2, 1}.

7, 6, 5, 5, 5, 4, 2, 2, 1

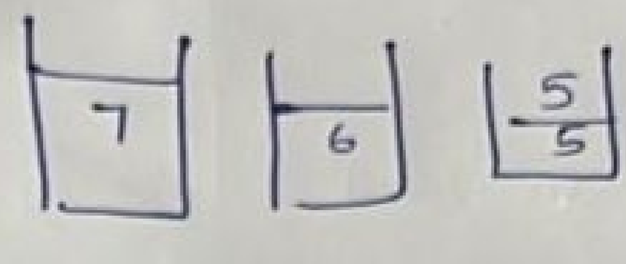


7, 6, 9, 5, 5, 4, 2, 2, 1

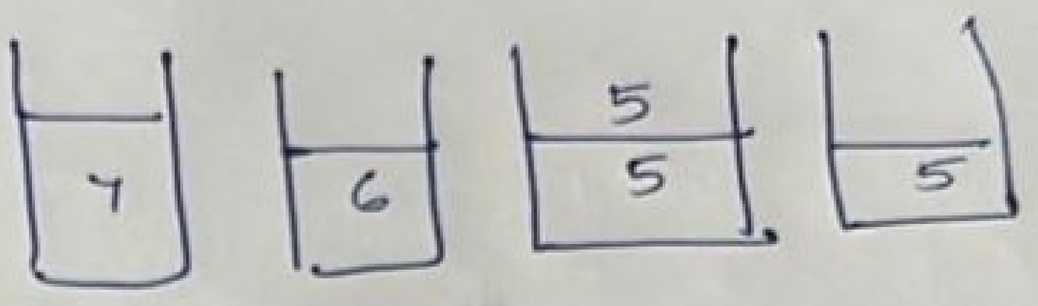
7, 6, 5, 5, 5, 4, 2, 2, 1



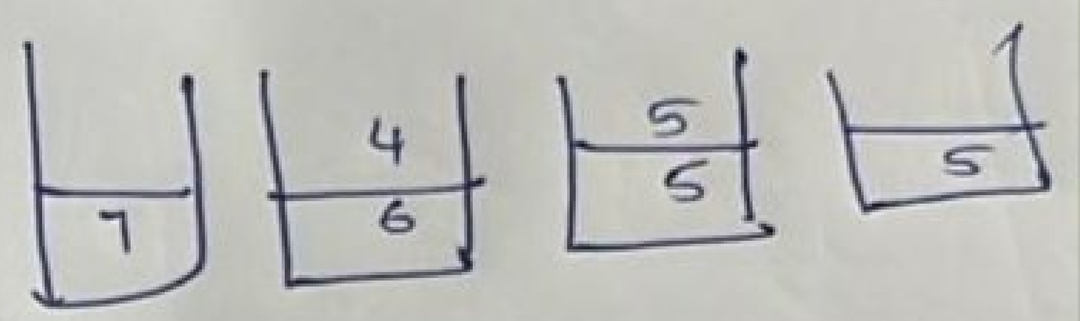
7, 6, 5, 5, 9, 5, 4, 2, 2, 1



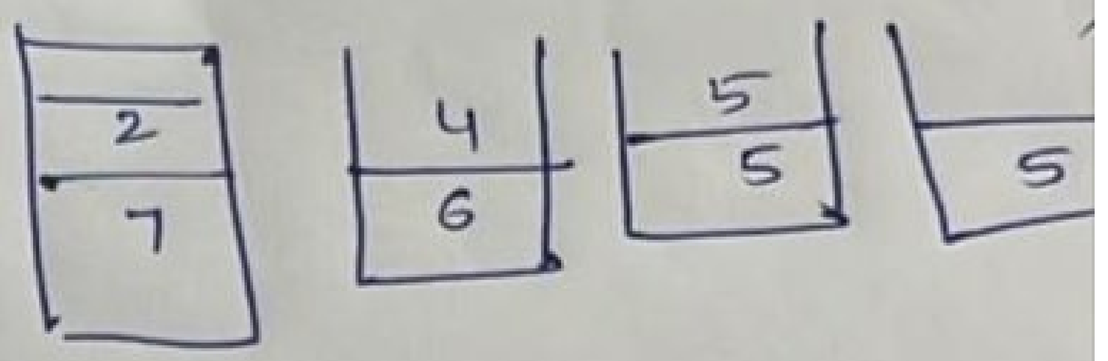
7, 6, 9, 5, 5, 4, 2, 2, 1



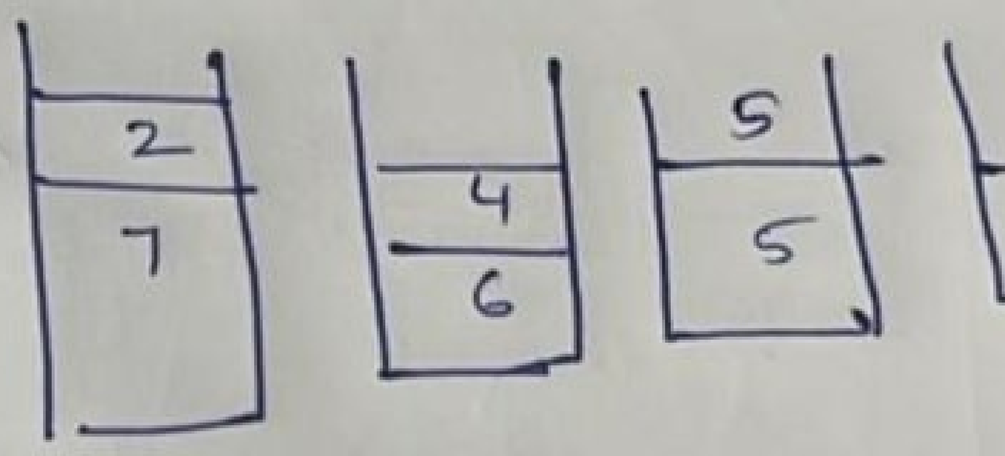
7, 6, 5, 5, 5, 4, 2, 2, 1



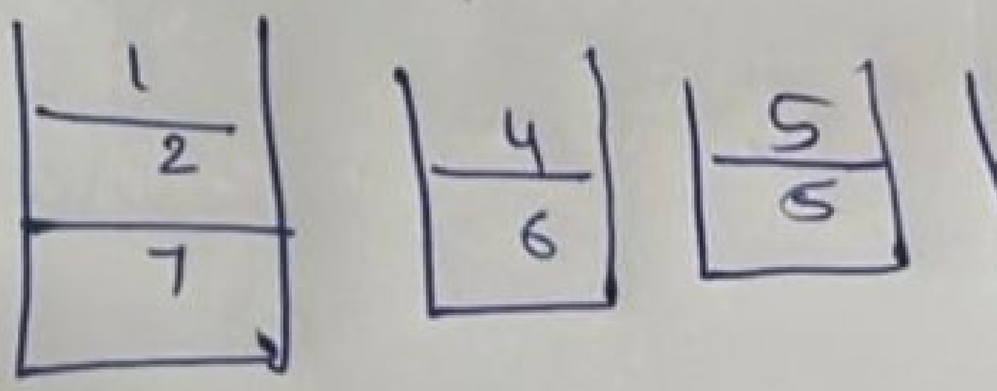
7, 6, 5, 5, 5, 4, 2, 2, 1



7, 6, 5, 5, 5, 4, 2, 2, 1



7, 6, 5, 5, 5, 4, 2, 2, 1



No. of bins required = 4