GREEDY METHOD :-

→ Used for solving problems, whose solutions are viewed as result of making a set/sequence of Decision.

→ Decisions are made in step wise manner.

→ Each step out of all options, greedily select that option, which satisfies the given criteria of the problem.

Terminology :-

→ Problem Definition.

→ Constraints (Requirements), condition.

implicit — explicit.

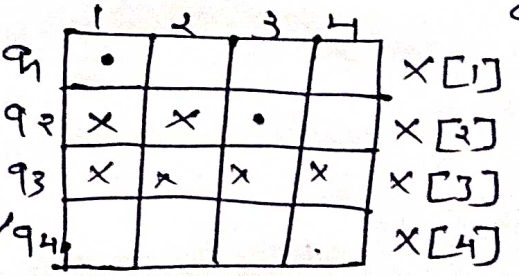
→ **n-Queens :-**

n-queens ($q_1, \ldots, q_n$)

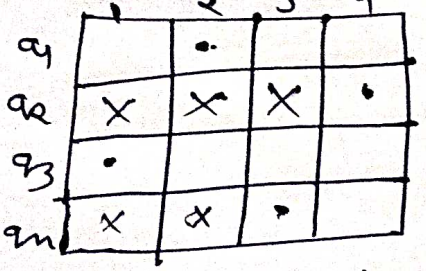
n = 4 ; ($q_1, \ldots, q_4$)

N = 4 → Limit Explicit Constraint

Implicit { No of Same Row, Same Column, Same Diagonal



new



✓ ⟨2, 4, 3, 1⟩

vector = ⟨2, 4, 1, 3⟩

x[1...n]

x[i] = Pos (column) in which Q i is placed.

Explicit $1 \leq x[i] \leq n$

So this is sequence of Decision.

= Greedy method

⇒ Solution space :- All possible ways of organization inputs satisfying any explicit Constraints,

n = 4

× [1,2,3,4]
[1,2,4,3]
[1,3,2,4]
:

yes 1,2,3,4 lies b/w 1 to n.
all possible solution.
So size = 4!

Solution space	=	n queen Game	=	n!

⇒ feasible Solution :

⇒ are those solution in the solution space that satisfy implicit constrain of problem.

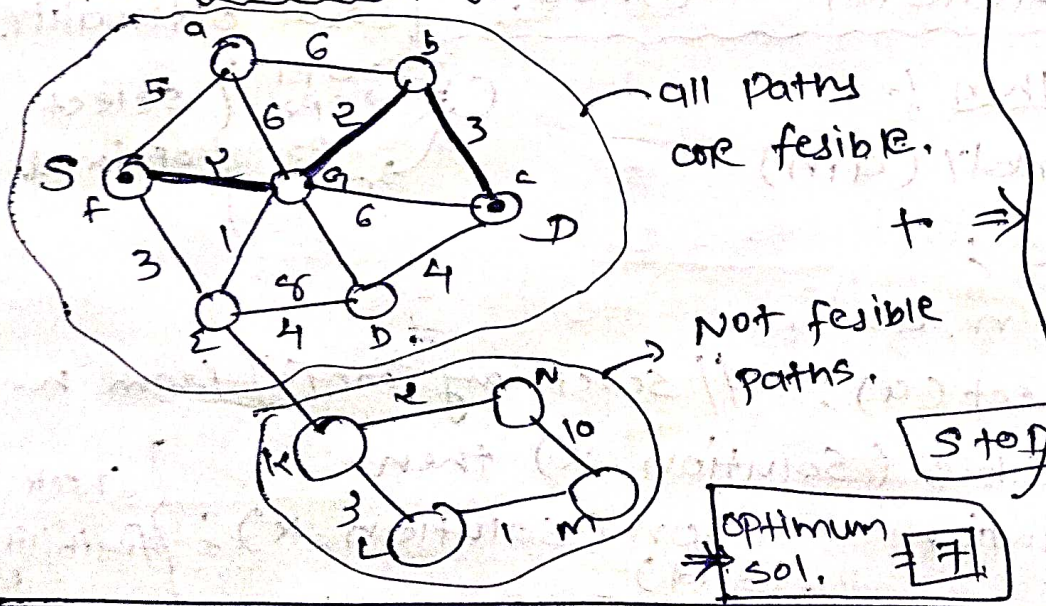
feasible = Correct solution :

⇒ Problems may have multiple feasible solution.

⇒ Objective function :

⇒ few problems may have objective fn. which tries to min/max. a given criteria of problem.

Ex. Shortest Path :



Not all Paths are fesible.

only 1 Optimal sol.

[objective] = minimizing cost of path.

Stop

optimum sol. = 7.

Optimal Solution :

↳ at that fesible solution it satisfies objective fn

↳ optimal solution Always Refers to the Value and hence it always unique. ⇒ (7) (f → s) → G → b → (c → d)

Problem (P)

Decision

→ its Result is always either (T/F) or (0/1) (feasible solution).

Ex:- n-queens.
→ sorting
  ↳ sorted Y/N
→ searching.
  ↳ Exist/Not Exist
→ Bankers Algorithm.
→ Deadlock Process.

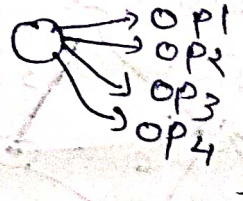
optimization

→ Requirements to determine a max/min value of a given criteria!
→ optimal solution (objective fn).

Ex. → shortest Path.
→ Graph colouring.
→ CPU sheduling.
→ Page Replacement.
→ Disk sheduling.
→ congestion control.
→ Minimize Traffic.

Analysis | Control Abstraction / General Method. | Principal of Local optimality.

GREEDY Algorithm :-



select optimal.

```
⇒ Algorithm Greedy (a, n)
{
  Solution = ∅ ;
  for i = 1 to n do
  {
    x = select (a) ;   // select any one [a] from n.
    if feasible (Solution, x) then
      Solution = union (Solution, x) ;
  }
  Return Solution ;
}
```

O(1) ← select
O(1) ← feasible
Solution = union (Solution, x) ; (Add) O(1) } Check feasibility.

Remember! any Problem which is solved by Greedy method, it may time complexity $T(n)$ is atleast $O(n)$.
⇒ O(1) O(1) O(1) Repeat n times.

① KNAPSACK PROBLEM :

(KNAP)

⇒ Given a KNAPSACK (BAG) of capacity 'M'.

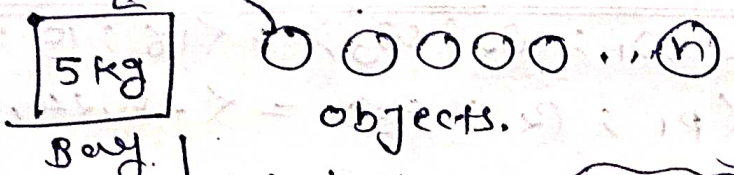
⇒ Given 'n' objects. ($o_1, o_2 \ldots o_n$)
- Weight (w).
- Profit (P).

objective = Maximize Profit.

KNAPSACK get filled up by (w). & we get profit of (P)

you can not put all in bag ≠ capacity.
↳ gain max. Profit.
↳ so we need Decision making.

⇒ Maximize Profit is subject to the condition that the total wt. ~~being~~ being put into the KNAP. Does not exceed its capacity.



↳ limited capacity.

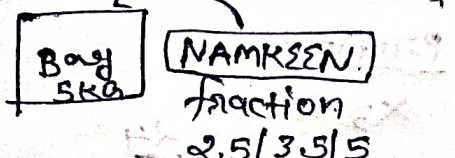
put in bag such as we gain Profit.

Decision making sequence → if i put it i will gain profit or Not.

Problem

Fraction Real
Continuous.
$0 \leq x_i \leq 1$
$< \infty >$

Binary (0/1)
<either include it totally or exclude it>
$x_i = (0/1)$ Put it / No put it.



Ex

① $\sum_{i=1}^{n} w_i \leq M$ ⇒ total Profit = $\sum_{i=1}^{n} P_i$ } Always True.

<if condition true ∴ No - Decision>

② for Solution Solving the Problem

Decision O_1 → $w_i * x_i$, $P_i * x_i$ $\sum_{i=1}^{n} > M$. strictly Greater.

objects. ↳ $x_1 = ?$, ↳ $x_2 = ?$

Total Profit :- $\max \sum_{i=1}^{n} p_i * x_i$ } Maximize Profit

KNAPSACK General Equation

Subject $\sum_{i=1}^{n} w_i * x_i \leq M$ } Condition to bag cap.

where $0 \leq x_i \leq 1$. → Explicit constraints

⇒ Maximize Profit = objective fn.

⇒ Implicit constraint = Not exceed capacity of bag.

Sol. Space

fraction (∞) [Greedy method]

0/1 KNAP 2^n [Dynamic programing.]

① $n=3$, $m=20$; $\langle w_1, w_2, w_3 \rangle = \langle 18, 15, 10 \rangle$

$\langle p_1, p_2, p_3 \rangle = \langle 25, 24, 10 \rangle$

$20 < 18+15+10$

Yes, working Fractional KNAP.

$\langle x_1, x_2, x_3 \rangle$

② Greedy About PROfit :-

$x_1 = 1$ ~> $\langle 18 \rangle$ weight. $\langle 25 \rangle$ high prof.

$x_2 = \frac{2}{15}$ ~> 2 kg Remain.

$x_3 = 0$

$\sum w_i x_i = 20$ total weight

Profit Cal.

$x_1 = 25$

$x_2 = \frac{2}{15} \times 24$

$x_3 = 0$

$\sum p_i x_i = 28.3$ total Profit.

③ Greedy about weight

least weight ↑ enter,

$x_3 = 1$

$x_2 = \frac{10}{15} = \frac{2}{3}$

$x_1 = 0$

$\sum w_i x_i = 20$ full weight

Profit Cal

$x_3 = 10$

$x_2 = \frac{2}{3} \times 24 = 16$

$x_1 = 0$

$\sum p_i x_i = 26$ total Profit.

Try to include more Number

here weight & Profit both are Not optimal. bcz. 18 15 10 weight 25 24 10 Profit. weight are Different,

if 18 18 18 / 25 30 10 } then we can say x_2 has high profit Value.

→ Feasible Solutions: are those Solutions in the Solution Space that Satisfy implicit Constraints of the Problem,

Problems may have multiple feasible solutions,

→ ...on:
...s may have
...which tries to
...given Criterion

Ex: Shortest Path

Criteria