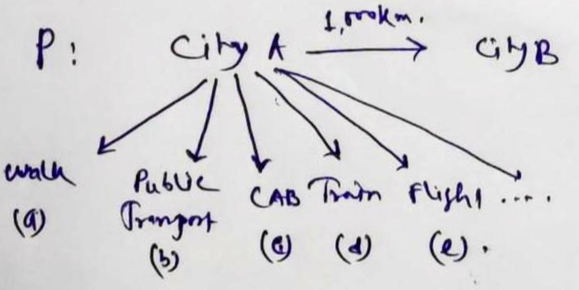
Greedy Method

The main idea of this technique, as the name implies, is to make a series of greedy choices in order to construct an optimal solution for a given problem.

The focus of these lectures is to enrich students with the general idea ~~of the~~ / structure of the greedy method & show how it can be applied to Knapsack & scheduling problems.

Problem → P: City A ⟶ City B

Suppose I have to travel from City A to City B, for any problem (for this one as well) there can be multiple Solution



Now let's add a condition/constraint in the above problem.

→ You have to travel within 24 hrs.

⇒ To satisfy this problem. we have to travel either by (d) Train, or (e) Flight, These are the feasible solution.

Now let's add another criteria, that I/we have to travel with minimum Cost. (≤ 2,000 INR).

⇒. Flight is costly Thus traveling by (d) Train is the only solution. It's a optimal Solution.

⇒ There can be multiple Feasible solution but there will be only one optimial ~~so~~ solution.

⇒ Further these kind of problem which require maximization/ minimization is called "optimization problem."

⇒ Greedy Method mostly deals with the optimization problem.

Algorithm

N = 4.

PS1	PS2	PS3	PS4
1	2	3	4

```
Greedy ( PS, N).
{   J=0;
    for (i=1 to n)
    {      x = selectfrom (PS);
        if ( x is Feasible)
        {
            Solution [J] = ~~Selection~~ x
            J++;
        }
    }
}.
```

Time to Modify the definition / Learn the definition.

#- Defn :- It is a method to solve the optimization problem, that involves searching through a set of configurations to find that one which minimizes or maximizes an objective function defined ~~on~~ on these configurations.

The greedy approach does not always lead to an optimal solution. But there are several problems that it does work optimally for, & such problem are said to posses the greedy choice property. This is the property that a global optimal configuration can be reached by a series of locally optimal choices, starting from a well-defined configuration.

Fractional Knapsack Problem.

object: O:	1	2	3	4	5	6	7
profit: P:	10	5	15	7	6	18	3
weight: w:	2	3	5	7	1	4	1.

There are 7 object, each object have some weight & have ~~som~~ a profit.
Let there is a bag whose capacity is 15 kg. which you will be using to carry objects from location A to B & on transfering you will receive the profit. So it is a contaneous transfer problem.

AIM ⇒ You have to transfer so that the profit is maximum.
⇒ it is a maximization problem.
→ there can be many solution but only one has maximum.

Thus we can apply greedy method to solve the problem.

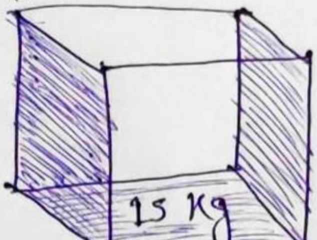
Assumption: The objects are divisible.

Objective:- How to include these object in the bag.
→ Selection criteria.

object: o_i	1	2	3	4	5	6	7
profit: P_i	10	5	15	7	6	18	3
weight: w_i	2	3	5	7	1	4	1.
P/w:	5	1.3	3	1	6	4.5	3

$x \left(\underset{\substack{x_1 \\ (b)}}{1} \quad \underset{\substack{x_2 \\ (d)}}{2/3} \quad \underset{\substack{x_3 \\ (d)}}{1} \quad \underset{x_4}{0} \quad \underset{\substack{x_5 \\ (a)}}{1} \quad \underset{\substack{x_6 \\ (c)}}{1} \quad \underset{\substack{x_7 \\ (e)}}{1} \right)$.

$0 \leq x \leq 1$.



a) $15 - 1 = 14$
b) $14 - 2 = 12$
(c) $12 - 4 = 8$
(d) $8 - 5 = 3$
(e) $3 - 1 = 2$
(f) $2 - 2 = 0$.

$$\boxed{\text{Constraint} \quad \sum_{i=0}^{n} x_i w_i \leq m.}$$

Lets verify if we are taking 15 kg, –

$$\sum_{i=0}^{n} x_i w_i = 1 \times 2 + 2/3 \times 3 + 1 \times 5 + 0 \times 7 + 1 \times 6 + 1 \times 4 + 1 \times 1$$

Again lets calculate the profit ; –

$$\sum_{i=0}^{n} x_i P_i = 1 \times 10 + 2/3 \times 5 + 1 \times 15 + 0 \times 7 + 1 \times 6 + 1 \times 18 + 1 \times 3$$

$$\boxed{\text{objective:} \quad \max\left(\sum_{i=0}^{n} x_i P_i\right).}$$

Thus we have reach our ~~b~~ objective with ~~...~~ the given constraints.

⇒ Now lets formalize the Solution:

Consider the fractional Knapsack problem where we are given a set of n items, such that each item 'i' has a positive benifit bi & a positive weight wi, & we wish to find the maximum-benifit subset that does not exceed the given weight W.

Here, we can break items into fractions arbitrarily.
i.e. we can take an amount x_i of each item 'i' such that

$$0 \leq x_i \leq w_i \text{ for each } i \in S \;\&\; \sum_{i \in S} (x_i) \leq W.$$

The total benifit of the items taken is determined by the objective function

$$\sum_{i \in S} b_i (x_i / w_i).$$

__Algorithm:__

FractionalKnapsack (S, W).

__Input__: Set S of Items, such that each item $i \in S$ has a positive benifit 'b_i' & a positive weight 'w_i' & positive maximum total weight W.

__output__: Amount x_i of each item $i \in S$ that maximizes the total benifit while not exceeding the maximum total weight W.

```
for (each item i ∈ S)
{
    xi ← 0;
    vi ← bi/wi;  [//value index of item i]
}
w ← 0;
while (w < W)
{
    remove item 'i' with highest value index from S.  [//Greedy choice]
    a ← min (wi, W-w)  [// more than W-w cause weight overflow].
    xi ← a;
    w = w + a
}.
```

$\boxed{O(n \log n)}$