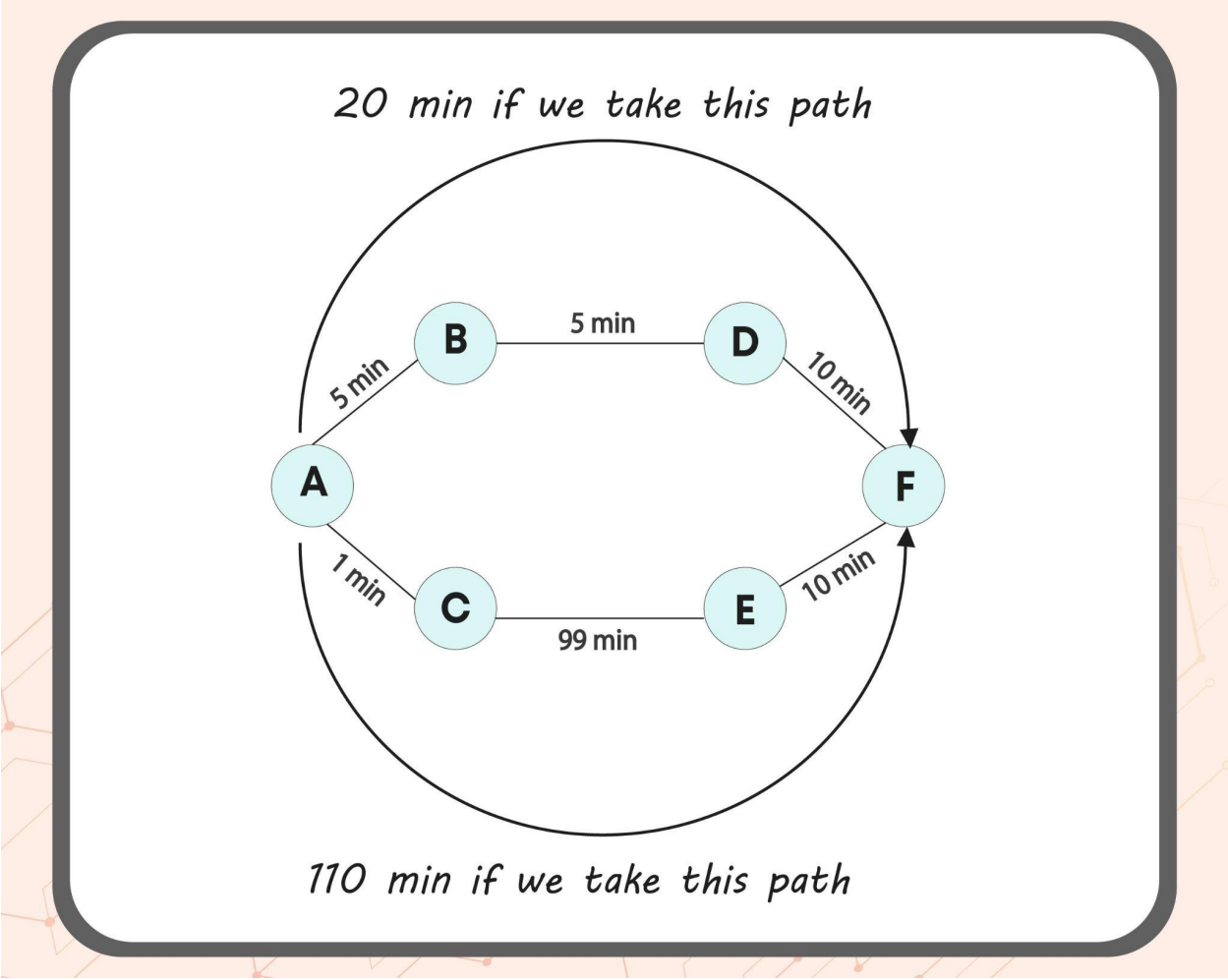
Greedy Problems

Introduction

Suppose we have 6 places A, B, C, D, E, F and we are given the time to reach from one destination to another, and we have to tell the shortest time required to reach from A to F.



Now, if we take the path A → B → D → F that will be our DP solution since we know that it is the overall shorter path, but if we take path A → C → E → F, then it will be greedy approach since we only know and judge our decision on the basis of path A → B and path A → C.

Greedy solution is not giving us the right answer in this problem but it might be a useful approach in others.

Activity Selection

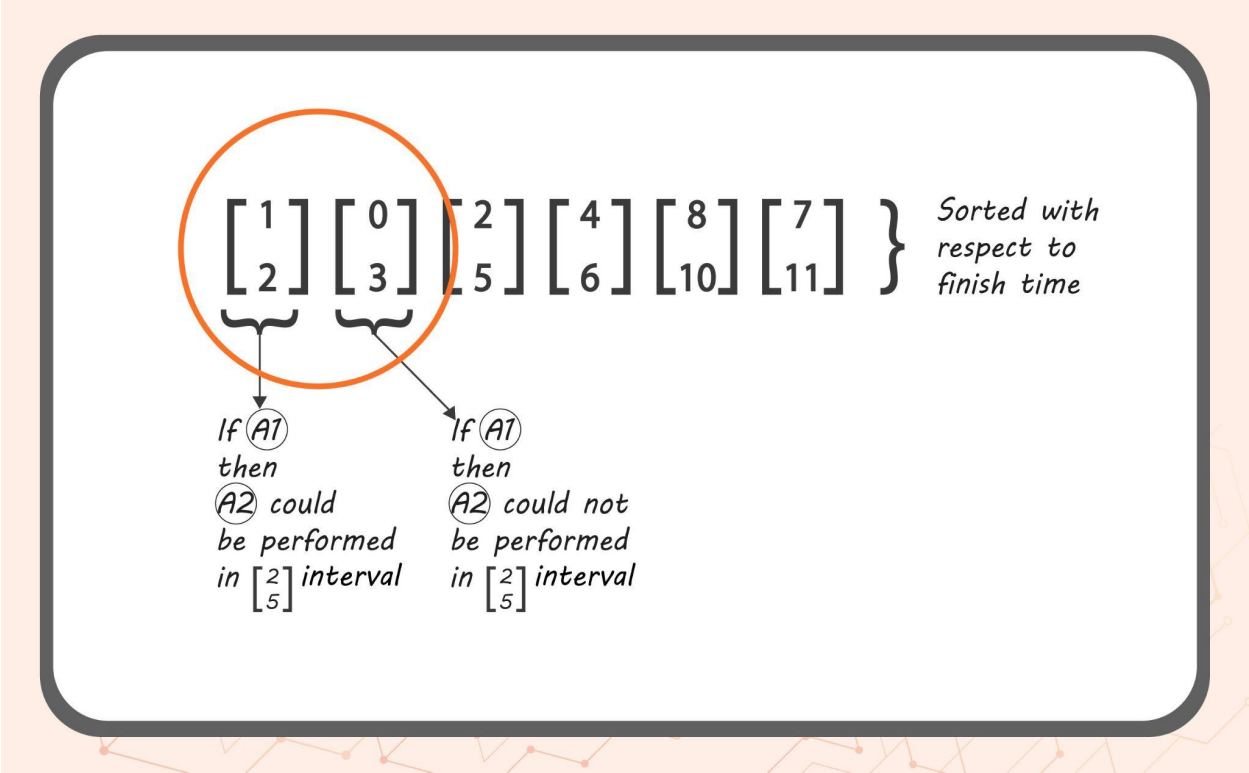
Problem Statement: We are given the starting time and the finishing time of activities and we have to tell how many activities can be completed in a given time interval. We can not do more than one activity simultaneously.

Example:

Given time interval to perform the activities: **0 to 11**.

Output: **3** (because the first activity will be performed in the time interval **1 to 2**, the second activity will be performed in the time interval **2 to 5** and the third activity will be performed in interval **8 to 10**.

The greedy approach to solving this problem starts by thinking to finish an activity as soon as possible and start with a new one. So we sort these two arrays on the basis of their finishing time or rather than saying that we sort the two arrays, we can say that we make a structure that has both the start time and finish time and we sort that on the basis of the finish time.



Now, A1 could be finished early if we do it in [1, 2] time interval which leaves us the option to start A2 in [2, 5] time interval.

So we say that greedy technique is applied when we start from an optimal answer from the first step and continue to look for the optimal answers in the next steps also.

The code to this solution is left for the reader as an exercise.

Minimum Absolute Difference in Array

Problem Statement: We are given an array and we have to tell the minimum absolute difference between two elements.

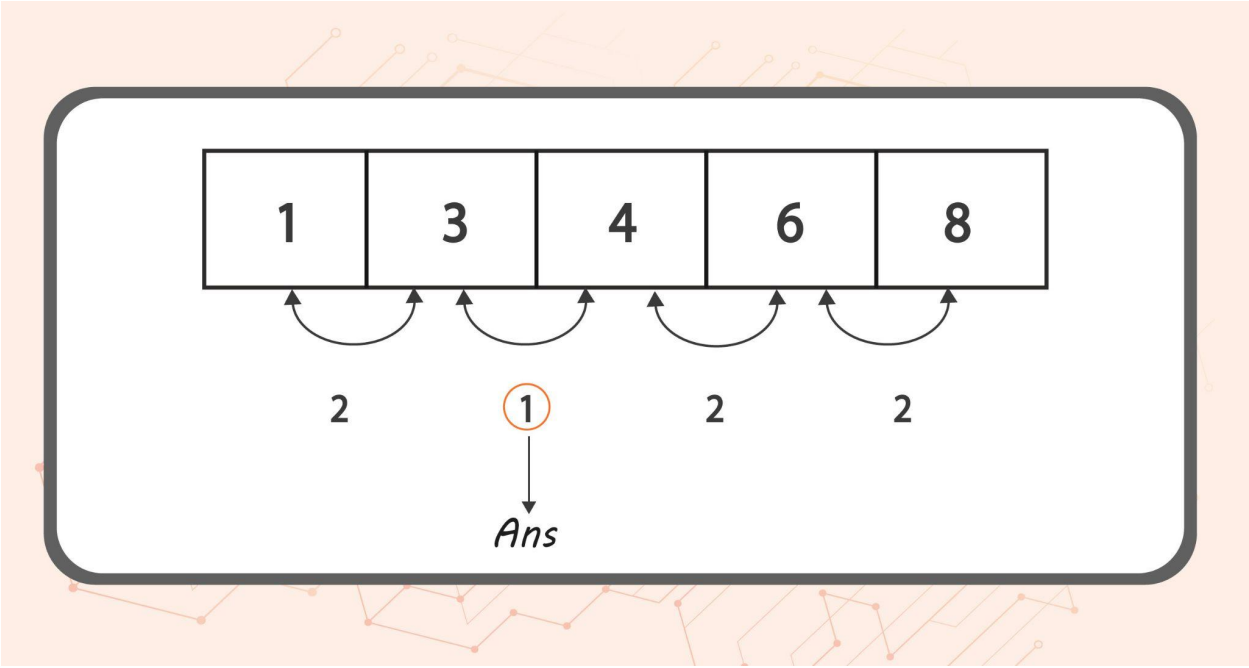
Explanation:

For example, **Arr = [1, 3, 2, 5, 4, 11, 7]**. Now, the difference between 1 and 3 is 2 and the difference between 3 and 2 is 1. So in this example the minimum absolute difference is 1.

Another example is, **Arr = [1, 4, 6, 3, 8]** then also the answer will be 1 i.e., the difference between 3 and 4.

Basic Approach: The basic approach will take $O(N^2)$ time as we will have to generate all the differences of elements by traversing using two loops.

Greedy Approach: If we sort this array then we can think of the greedy technique. After sorting, the array becomes **Arr = [1, 3, 4, 6, 8]**. Now if we start traversing the elements we need not use two loops because we just have to check the minimum difference in the consecutive elements only.



The code to this solution is left for the reader as an exercise.

Fractional Knapsack

Problem Statement: This problem is just like the original knapsack problem in which we were given two arrays: one with weights of the items and the other with the values of these items and we were given a maximum weight that we can put in the knapsack. Now in this we can take the fractional weight of an item by adding only a fraction of the item.

Explanation:

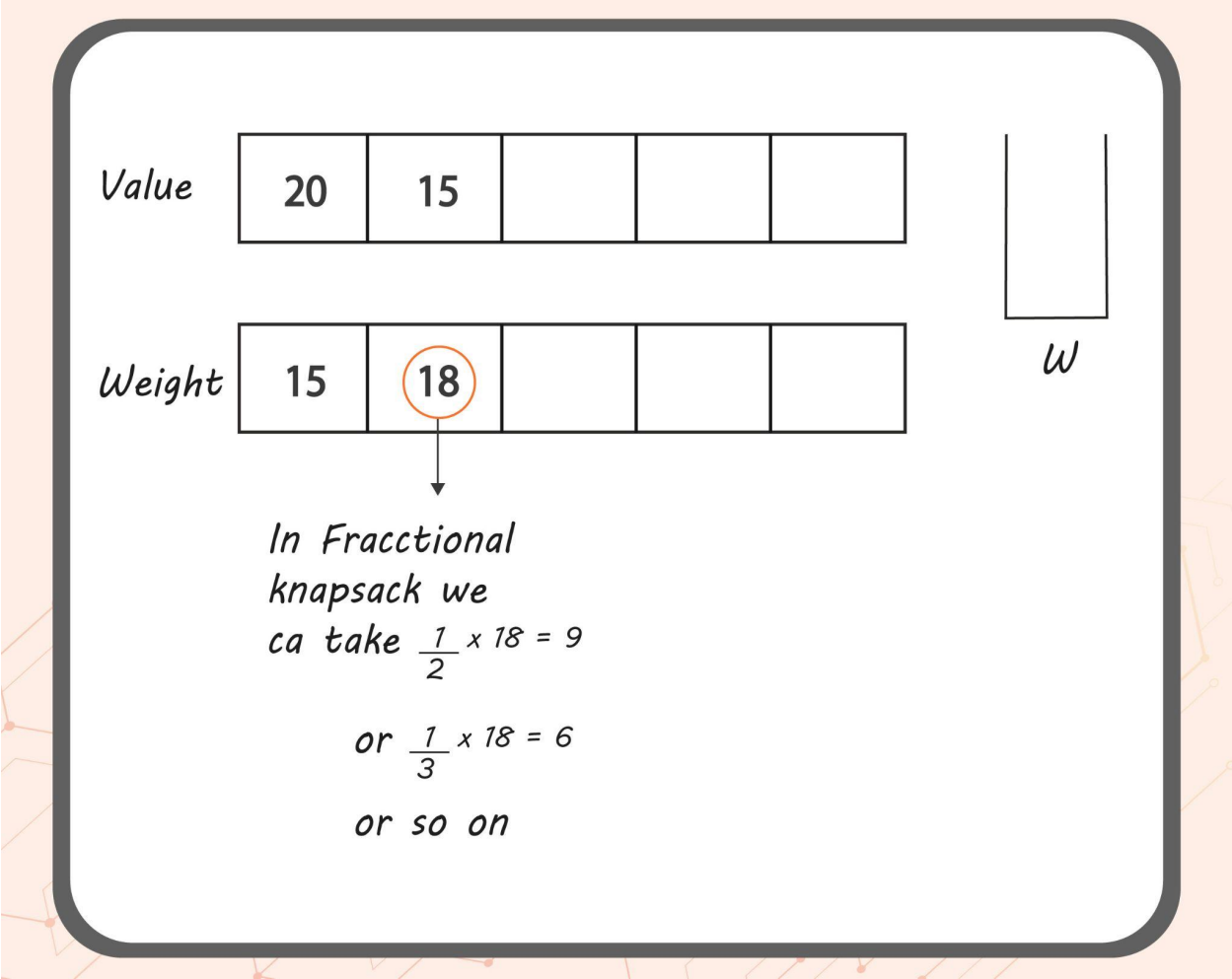
Example 1:

Let us consider the different items that need to be put in the knapsack as dry fruits like Cashews, Raisins, Almonds in quantities 6 kg, 3 kg, and 4 kg respectively and we have a knapsack with weight capacity as 5 kg. The relative prices of these are in the order Cashew > Raisins > Almonds. Now, as we can include fractional weight of items, hence, we will obviously want to fill our knapsack with the items with maximum relative price i.e. Cashews.

Now, let's think for a second that what if these items were not dry fruits and were statues, then we would not have the option to take a part of the total weight of the item as breaking a statue into parts will result in the loss of its value and we could not have chosen the 6 kg statue with the maximum value because our knapsack's capacity is only 5 kg. Therefore, if we have statues as items, then it becomes a case of 0-1 Knapsack.

But since dry fruits can be easily divided and measured according to the weight, that means we now have the option to take only 5 kg out of 6 kg Cashews and still maximize the value of items in the knapsack.

Let's take a look at the image below with another example to understand fractional knapsack.



So, in this problem, we will have to make an array that contains the value per weight of the items and sort that in descending order and then start filling the knapsack accordingly. Therefore we are applying a greedy approach by choosing the items that have the maximum value per weight.

Let's look at the dry run of another example:

Weight of Knapsack = 60 kg

ITEM	WEIGHT	VALUE	VALUE / WEIGHT
I1	5	30	6
I2	10	40	4

I3	15	45	3
I4	22	77	3.5
I5	25	90	3.6

Now we sort all the items in decreasing order of their value per weight.

I1	I2	I5	I4	I3

We are ready to start filling the knapsack

Remaining Knapsack Weight	**Items in Knapsack**	**Cost**
60	NIL	0
55	I1	30
45	I1, I2	70
20	I1, I2, I5	160

Now, the remaining knapsack weight is 20kg but I4 weighs 22kg. Therefore we add only 20 kg of I4 which makes the total cost of our knapsack to be 160 + (20/22)*77 = 230.