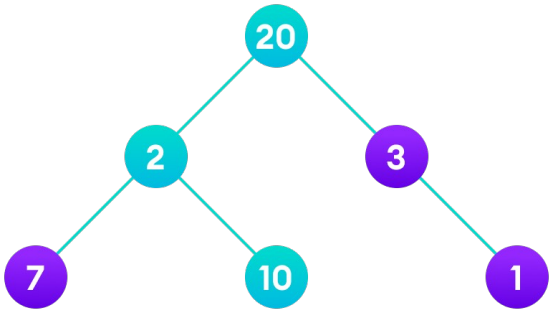
Greedy Approaches

- A greedy algorithm is a simple, intuitive algorithm that is used in optimization problems.
- The algorithm builds up a solution piece by piece, always choosing the next piece that offers the most obvious and immediate benefit.
- Therefore the approach searches for the best option available at the moment. It doesn't worry whether the current best result will bring the overall optimal result.
- This algorithm may not produce the best result for all the problems. It's because it always goes for the local best choice to produce the global best result.



Greedy Properties of Problems

1. Greedy Choice Property

If an optimal solution to the problem can be found by choosing the best choice at each step without reconsidering the previous steps once chosen, the problem can be solved using a greedy approach. This property is called greedy choice property.

2. Optimal Substructure

If the optimal overall solution to the problem corresponds to the optimal solution to its subproblems, then the problem can be solved using a greedy approach. This property is called optimal substructure.

Greedy Approaches

```
Greedy(a, n) {          // a is an input with n items.
    solution = ø;
    for i = 1 to n do {
        x = select(a);
        if feasible(solution, x) then
            solution = Union(solution, x);
    }
    return solution;
}
```

Problem: Selection Sort

```
private static void sort() {
    for (int i = 0; i < inputs.length - 1; i++) {       //  n
        int minValueNdx = i;                            //  n-1
        for (int j = i + 1; j < inputs.length; j++)     //  (n-1)*n/2 + n-1
            if (inputs[j] < inputs[minValueNdx])        //  (n-1)*n/2
                minValueNdx = j;
        int tempValue = inputs[i];                      //  n-1
        inputs[i] = inputs[minValueNdx];                //  n-1
        inputs[minValueNdx] = tempValue;                //  n-1
    }
}
```

Problem: Coins to be returned

Greedy problem: You have to make a change of an amount using the smallest possible number of coins.

Note: There is no limit to the number of each coin you can use.

Problem: Minimum product subset of an array

Greedy problem: Given array a, we have to find the minimum product possible with the subset of elements present in the array.

Note: The minimum product can be a single element also.

Problem: Maximum product subset of an array

Greedy problem: Given an array a, we have to find the maximum product possible with the subset of elements present in the array.

Note: The minimum product can be a single element also.

Problem: Activity selection

Greedy problem: You are given n activities with their start and finish times. Select the maximum number of activities that can be performed by a single person, assuming that a person can only work on a single activity at a time.

Problem: Job Sequencing

Greedy problem: Given an array of jobs where every job has a deadline and associated profit if the job is finished before the deadline. It is also given that every job takes a single unit of time, so the minimum possible deadline for any job is 1. Maximize the total profit if only one job can be scheduled at a time.

Problem: Fractional Knapsack Problem

Greedy problem: Given the weights and values of N items, in the form of {value, weight} put these items in a knapsack of capacity W to get the maximum total value in the knapsack. In Fractional Knapsack, we can break items for maximizing the total value of the knapsack.