Examples (Greedy Algorithms) 1

1) Assign cookies :-

greed = [1, 5, 3, 3, 4] s = [2, 2, 1, 3, 4, 1]

Ans = 3

greed = [1, 3, 3, 4, 5] s = [1, 1, 2, 2, 3, 4]

return i;

```
func (greed, size) {
    n = greed.size , m = size.size
    l = 0 , r = 0
    sort (greed) → O(nlogn)    sort (size) → O(nlogn)
    While (l < m) —— O(m)
    {
        if (greed[r] <= size[l])
            { r = r+1;
            }
        l = l+1;
    }
    return r;
}
```

T(n) = O{(nlogn) + (nlogn) + (m)}

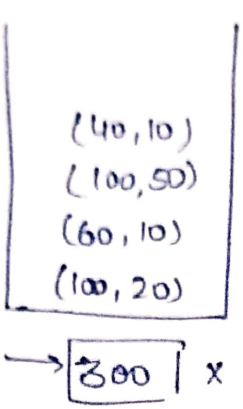
SC :- O(1)

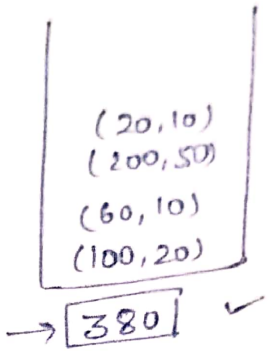
→ fractional knapsack Problem comparator :-

```
bool comp(Item val1, Item val2) {
    if (val1.value / val1.weight >= val2.value / val2.weight)
        return true
    return false
}
```

Q) Fractional Knapsack problem :-

arr[ ] = [(100, 20) (60, 10) (100, 50) (200, 50)] W = 3





⇒ Calculate per unit max (100/20) = 5

arr [(100, 20) (60, 10) (100, 50) (200, 50)]
5 6 2 4

⇒ Pseudo code :-

```
double fcn (Item arr[ ], W) {
    sort (arr, comp) ⟶ O(NlogN)
    total = 0
    for(i=0 ⟶ n) { ⟶ O(1)
        if(arr[i].weight <= W) {
            total += arr[i].value
            W = W - arr[i].weight
        }
        else {
            total += int(arr[i].value / arr[i].weight) x W   (⟶ Typecast)
            break
        }
    }
    return total
}
```

TC :- O(NlogN+

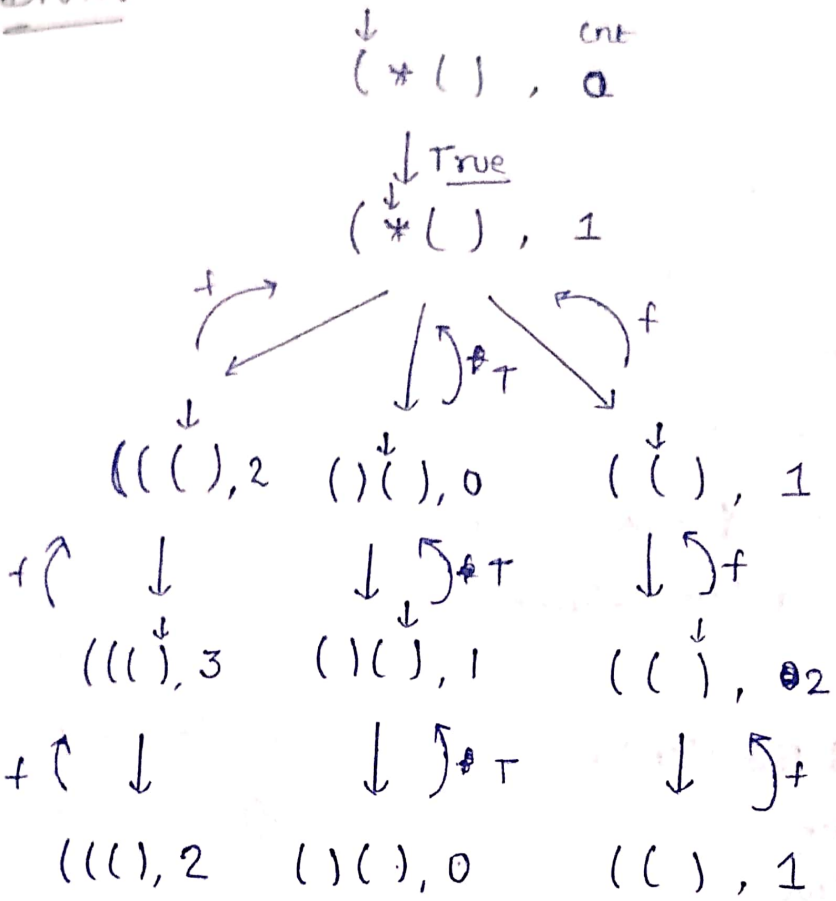
SC :- O(1)

4) valid parentheses :-

(*) or ((*) or (**(

* can be -, (,).

→ Brute :-




```
fc^n (s, ind, cnt) {
  if(cnt<0) return false
  if (ind==n) {
      return (cnt==0)
  }
  if (s[ind] == '(') return fc^n (s, ind+1, cnt+1)
  If (s[ind] == ')') return fc^n (s, ind+1, cnt-1)
  return fc^n (s, ind+1, cnt+1) or fc^n (s, ind+1, cnt-1) or
         fc^n (s, ind+1, cnt)
```

TC :- 3^n

SC :- O(N)

3) Lemonade change :-

bills = [5 5 5 10 20]

5 → 0 1 2 3 4 ...
10 → 0 1 0
20 → 0 1

```
bool func(arr){
    five=0, ten=0
    for(i=0 ⟶ n){
        if(arr[i]==5){
            five++; }
        else if(arr[i]==10){
            if(five){ five-- ten++
            else return false
        else{
            if(five && ten){
                ten--
                five--
            }
            else if(five>=3){
                five-=3;
            }
            else
                return false
        }
    return
}
return true
}
```

TC :- O(N)

SC :- O(1)