1. "침몰하는 타이타닉" 정답코드

```
class Solution {
        public int solution(int[] nums, int m){
                int answer = 0;
                Arrays.sort(nums);
                int left = 0;
                int right = nums.length-1;
                while(left <= right){
                        if(nums[left] + nums[right] <= m){
                                answer++;
                                left++;
                                right--;
                        }
                        else{
                                answer++;
                                right--;
                        }
                }
                return answer;
        }
}
```

2. "이동횟수" 정답코드

```
class Solution {
        public int solution(int[] nums){
                int answer = 0;
                Arrays.sort(nums);
                int left = 0;
                int right = nums.length-1;
                while(left <= right){
                        if(nums[left] + nums[right] <= 5){
                                answer++;
                                left++;
                                right--;
                        }
                        else{
                                answer++;
                                right--;
                        }
                }
                return answer;
        }
}
```

3. "스프링 쿨러" 정답코드

```
class Solution {
        public int solution(int n, int[] nums){
                int answer = 0;
                int[][] line = new int[nums.length + 1][2];
                for(int i = 0; i <= n; i++){
                        line[i][0] = Math.max(0, i - nums[i]);
                        line[i][1] = Math.min(n, i + nums[i]);
                }
                Arrays.sort(line, (a, b) -> a[0] - b[0]);
                int start = 0, end = 0, i = 0;
                while(i < line.length){
                        while(i < line.length && line[i][0] <= start){
                                end = Math.max(end, line[i][1]);
                                i++;
                        }
                        answer++;
                        if(end == n) return answer;
                        if(start == end) return -1;
                        start = end;
                }
                return answer;
        }
}
```

▶▶ Comment :

```
while(i < line.length && line[i][0] <= start){
        end = Math.max(end, line[i][1]);
        i++;
}
```

start 지점의 잔디부터 물을 줄 수 있는 범위가 가장 큰 스프링클러를 찾는 그리디 핵심 코드입니다.

4. "꽃이 피는 최단시간" 정답코드

```
class Solution {
        public int solution(int[] plantTime, int[] growTime){
                int answer = 0;
                int n= plantTime.length;
                int[][] list = new int[n][2];
                for(int i = 0; i < n; i++){
                        list[i][0] = plantTime[i];
                        list[i][1] = growTime[i];
                }
                Arrays.sort(list, (a, b) -> b[1] - a[1]);
                int start = 0, end = 0;
                for(int[] x : list){
                        end = start + x[0] + x[1];
                        answer = Math.max(answer, end);
                        start += x[0];
                }
                return answer;
        }
}
```

▶▶ Comment :

성장기간이 가장 긴 꽃씨부터 먼저 심는 그리디 문제입니다.

end = start + x[0] + x[1]; //현재의 꽃씨를 심고 이 꽃씨가 꽃이 피는 날이 end입니다.

5. "전투게임" 정답코드

```
class Info implements Comparable<Info>{
    public int idx;
    public Character team;
    public int power;
    Info(int idx, Character team, int power){
            this.idx = idx;
            this.team = team;
            this.power = power;
    }
    @Override
    public int compareTo(Info ob){
            return this.power - ob.power;
    }
}
class Solution {
    public int[] solution(String[] students){
            int n = students.length;
            int[] answer = new int[n];
            ArrayList<Info> list = new ArrayList<>();
            for(int i = 0; i < n; i++){
                    Character a = students[i].split(" ")[0].charAt(0);
                    int b = Integer.parseInt(students[i].split(" ")[1]);
                    list.add(new Info(i, a, b));
            }
            Collections.sort(list);
            HashMap<Character, Integer> Tp = new HashMap<>();
            int j = 0, total = 0;
            for(int i = 1; i < n; i++){
                    for( ; j < n; j++){
                            if(list.get(j).power < list.get(i).power){
                                    total += list.get(j).power;
                                    char x = list.get(j).team;
                                    Tp.put(x, Tp.getOrDefault(x, 0) + list.get(j).power);
                            }
                            else break;
                    }
                    answer[list.get(i).idx] = total - Tp.getOrDefault(list.get(i).team, 0);
            }
            return answer;
    }
}
```

6. "최대 인원수" 정답코드

```
class Solution {
        public int solution(int n, int[][] trans, int[][] bookings){
                int answer=0;
                int[] sum = new int[n+1];
                for(int[] x : trans){
                        sum[x[0]] += x[2];
                        sum[x[1]] -= x[2];
                }
                for(int i = 1; i <= n; i++){
                        sum[i] += sum[i-1];
                }
                int bN = bookings.length;
                Arrays.sort(bookings, (a, b) -> a[0] - b[0]);
                LinkedList<Integer> nums = new LinkedList<>();
                int ix = 0;
                for(int i = 1; i <= n; i++){
                        while(!nums.isEmpty() && nums.peek() == i){
                                answer++;
                                nums.pollFirst();
                        }
                        while(ix < bN && bookings[ix][0] == i){
                                nums.add(bookings[ix][1]);
                                ix++;
                        }
                        Collections.sort(nums);
                        while(nums.size() > sum[i]){
                                nums.pollLast();
                        }
                }
                return answer;
        }
}
```

▶▶ Comment :

```
Collections.sort(nums);
while(nums.size() > sum[i]){
        nums.pollLast();
}
```

도착역이 가까운 어린이부터 우선순위로 해서 기차에 태우고 다음 역으로 가는 그리디 핵심 코드입니다.