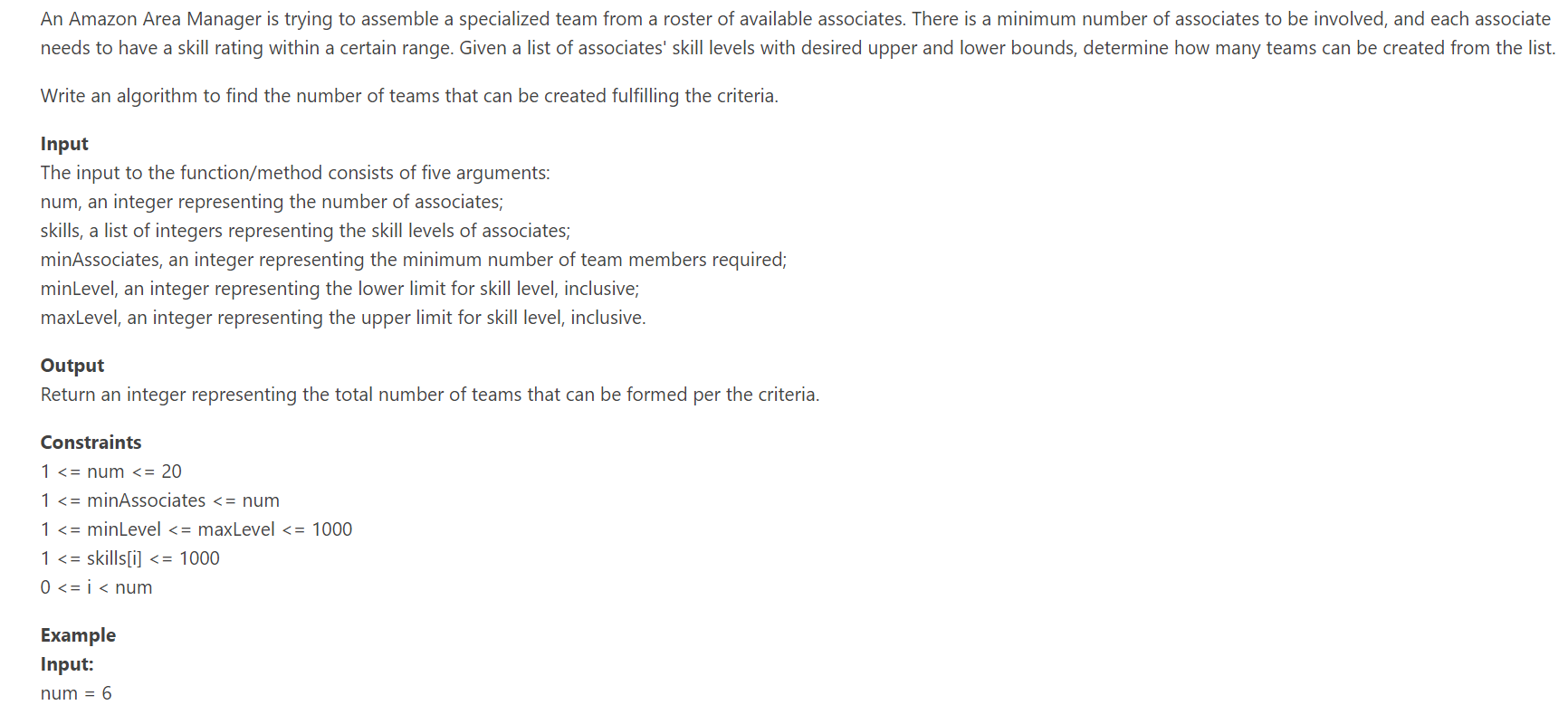
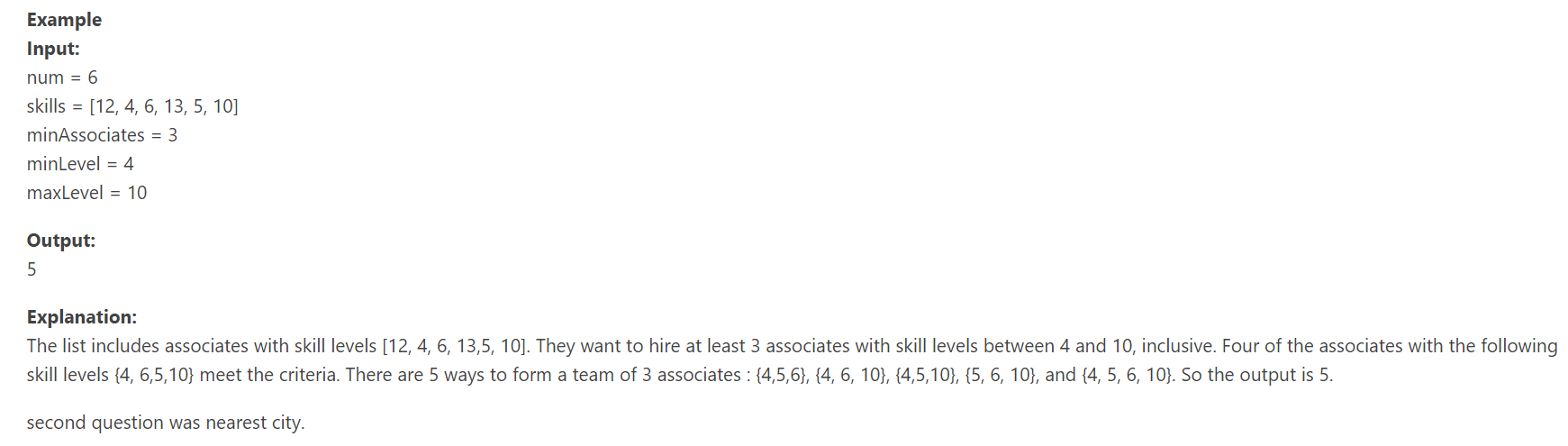
Amazon | OA 2020 | Count Teams

LEET CODE : <https://leetcode.com/discuss/interview-question/903119/>





static int count = 0;

public static int countTeams(int num, int[] skills, int minAssociates, int minLevel, int maxLevel) {

\_countTeams(skills, minAssociates, minLevel, maxLevel, 0);

return count;

}

private static void \_countTeams(int[] skills, int minAssociates, int minLevel, int maxLevel, int currIndex) {

if (minAssociates <= 0) {

count++;

}

if (currIndex == skills.length) {

return;

}

for (int i = currIndex; i < skills.length; i++) {

if(skills[i] >= minLevel && skills[i] <= maxLevel) {

\_countTeams(skills, minAssociates - 1, minLevel, maxLevel, i+1);

}

}

}

The main idea is that for each worker, if they fall within the skill threshold, we can either select them, or skip them. I have provided a recursive solution, but there is an option for memoization so we don't do recursive subproblems.

Recursive Solution:

public static void main(String[] args) {

System.out.println(solve(new int[] {12, 4, 6, 13, 5, 10}, 3, 4, 10));

}

private static int solve(int[] skill, int min, int minLevel, int maxLevel) {

return helper(skill, min, minLevel, maxLevel, 0);

}

private static int helper(int[] skill, int minAssociates, int minLevel, int maxLevel, int idx) {

if (idx >= skill.length) {

if (minAssociates <= 0) return 1;

return 0;

}

int sum = 0;

if (skill[idx] >= minLevel && skill[idx] <= maxLevel) {

sum += helper(skill, minAssociates - 1, minLevel, maxLevel, idx + 1);

sum += helper(skill, minAssociates, minLevel, maxLevel, idx + 1);

} else {

sum += helper(skill, minAssociates, minLevel, maxLevel, idx+1);

}

return sum;

}