
# Assumption University

**Vincent Mary School of Science and Technology**
Department of Computer Science

CS3201 ALGORITHM DESIGN
TERM PROJECT REPORT

**1656 Far Away Kingdom's Army**
Timus Online Judge

**Submit to**
**Asst. Prof. Dr. Thitipong Tanprasert**

**by**
5911694 Rachatha Pramualsuk

Semester 2/2017

# OUTLINES

# Problem Definition

1656 Far Away Kingdom's Army (Difficulty: 194)
*Vladimir Yakovlev (idea by Magaz Asanov),*
*NEERC 2008, Eastern subregion quarterfinals*

*Time limit: 1.0 second*
*Memory limit: 64MB*

Solovey-Razboynik invaded Far Away Kingdom, and nobody could be safe from him. Then King called his Army Commander and ordered him to send his brave soldiers against the foe, and not come back without the victory. Commander went to Baba Yaga to ask for her advice.

"To conquer the enemy, you need a special formation," she said." Put your soldiers into squares but remember that if two soldiers stand in the same row or column on the same side from the center of that row or column, then the soldier who is closer to the center must be at least as tall as the other. Only then will you be able to defeat Solovey.
If you don't do as I say, he'll blow your army away."

Commander assigned the soldiers to several squares and measured their height. However, to place them as Baba Yaga had said wasn't easy. Help him to place the soldiers as required.

## Input

In the first line you are given the size *n* of the square; it is an odd number from 3 to 9. Each of the next $n^2$ lines contains the height of a soldier in centimeters. The army of Far Away Kingdom enrolls men whose height is from 170 to 200 centimeters.

## Output

Output an $n \times n$ table with cells showing the heights of soldiers from the given list. In each row and in each column, the largest number must be in the center, and the numbers mustn't increase in the directions from the center to the ends.

Sample Input & Output

| input | output |
| --- | --- |
| 3<br>170<br>170<br>180<br>175<br>170<br>175<br>175<br>170<br>175 | 170 175 170<br>175 180 175<br>170 175 170 |

# Problem Analysis

The number of people (heights) is $n^2$ where n is an odd number that is also greater than or equal to 3 but less than or equal to 9. The problem stated that the largest height must be in the center and the heights decreases as the index goes outward (but does not implies the height to distribute equally as it disperses outward.

Sorting of the list of heights is required as it is mandatory to specify the tallest person to be placed in the middle. Also, for the sake of simplicity since the problem restrict *n* to be odd. We can deduce that there will always be the middle *m* for each row and column (as we need to place the tallest at $row_m$ $column_m$)

# Problem Solution

*Programming language: Python 3.5*

For ease of indexing, I sorted the list height right away as soon as I read the input.

```
14  n = int(input())
15  h = [201]*99
16  for i in range(1,(n**2)+1):
17      h[i] = (int(input()))
18
19  h.sort(key=lambda x:x)
```

Noted the value 201 is out of the bound of the problem, I did it on purpose to save time amortizing and to sort it easily.

Prepare the 2-dimensional list, size (10 *10) filled with -1 as invalid values

```
23  tb = [[-1] * (10) for i in range(10)]
```

Again, I declared the length to be 10*10 as it is out of bound of the problem to save time amortizing space.

Next, I created a list of 1s and -1s to use later in the computation of indexes will be demonstrated later.

```
21  oe = [1, -1, 1, -1, 1, -1, 1, -1, 1, -1]
```

Now that everything is set. I did a double for loop to iterate through each row and column and in side, I devise a formula to specify which index will which height goes to.

```
25  for i in range(1,n+1):
26      for j in range(1,n+1):
27          r = int((n+1)/2+oe[i]*int(i/2))
28          c = int((n+1)/2+oe[j]*int(j/2))
29          tb[r][c] = h[n*n-(i*n+j-n)]
```

Noted that I start my list at 1 not 0, this is due to the structure of python capable of accessing index -1 (which is the last element of the list), I did just like that to be safe.

r and c represent the indexes of rows and columns in tables, and the logic in computing that are n+1/2 which is the middle + oe[i] (the shift to the left and right (if i) or up down (if j) from the middle. Then I update table tb with the height which goes from n-1 to 0.

Finally, we print table tb which values have operated on.

```
for i in range(1,n+1):
    for j in range(1,n+1):
        print(tb[i][j], end=' ')
    print()
```

The full source code is shown below.

```
# http://acm.timus.ru/problem.aspx?space=1&num=1656

# 3
# 170
# 170
# 180
# 175
# 170
# 175
# 175
# 170
# 175

n = int(input())
h = [201]*99
for i in range(1,(n**2)+1):
    h[i] = (int(input()))

h.sort(key=lambda x:x)
# print(h)
oe = [1, -1, 1, -1, 1, -1, 1, -1, 1, -1]

tb = [[-1] * (10) for i in range(10)]

for i in range(1,n+1):
    for j in range(1,n+1):
        r = int((n+1)/2+oe[i]*int(i/2))
        c = int((n+1)/2+oe[j]*int(j/2))
        tb[r][c] = h[n*n-(i*n+j-n)]
        # print(" "+ str(r) + " " + str(c) + " " + str(tb[r][c]))

for i in range(1,n+1):
    for j in range(1,n+1):
        print(tb[i][j], end=' ')
    print()

```

# Submission Result

| ID | Date | Author | Problem | Language | Judgement result | Test # | Execution time | Memory used |
|---|---|---|---|---|---|---|---|---|
| 7877172 | 19:10:33<br>9 May 2018 | Rachatha Pramualsuk | 1656 | Python 3.6 | Accepted | | 0.062 | 324 KB |