Data Science Bootcamp: Computing

Programming Test

Problem 1. Valid Path in Graph (graph.ipynb)

You are given a bi-directional graph with n vertices labeled from 0 to n - 1.
The graph is represented by an array of edges, where each edge is denoted as edges[i] = (ui, vi), indicating a connection between vertex ui and vertex vi.
Your goal is **to check if there is a valid path from a given source vertex to a destination vertex**.
If such a path exists, return true; otherwise, return false.

[Constraints]

1. The number of the vertices is between 1 and 2 * 10^5. (1 <= n <= 2 * 10^5)
2. The length of the edges array is between 1 and 2 * 10^5. (1 <= edges.length <= 2 * 10^5)
3. The vertex ui and vi range from 0 to n - 1. (0 <= ui, vi <= 2 * 10^5 - 1)
4. The vertices specified in the edges are distinct from each other. (ui != vi)
5. All pairs of vertices (ui, vi) are distinct.

[Test Case]

Test Case 1)

```
n = 4
edges = [(0,1),(0,2),(1,3),(2,3)]
source = 0
destination = 3

g = Graph()
result = g.getresult(n, edges, source, destination)
result
```

```
True
```

Test Case 2)

```
n = 6
edges = [(0,1),(1,2),(3,4),(4,5)]
source = 0
destination = 4

g = Graph()
result = g.getresult(n, edges, source, destination)
result
```

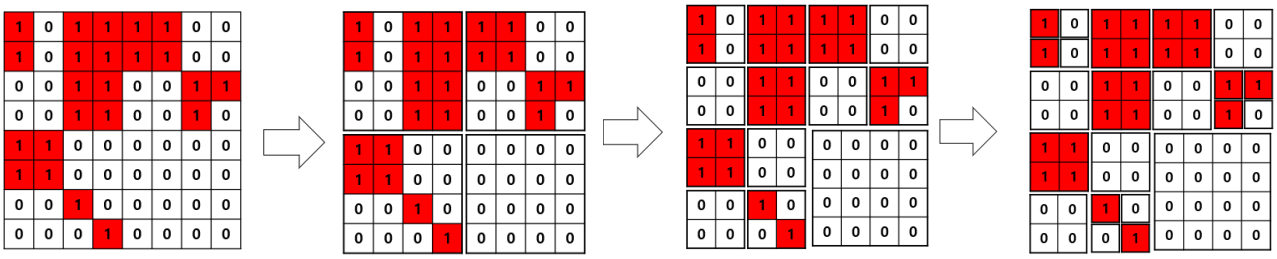
```
False
```

Problem 2. Land Conqueror (land.ipynb)

Wonka and Raymond are playing 'Land Conqueror'. The game has the following rules.

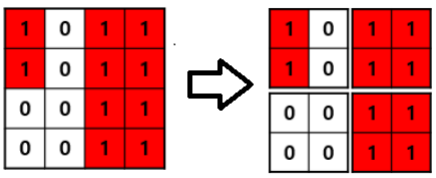
1. The total size of the land is N×N *square*(N=2^k, $1 \leq k \leq 5$, where k is an integer). A *square* is the smallest unit of land.
2. All *squares* are randomly colored with two colors: white(0) and red(1).
3. If a land contains *squares* with different colors, the land must be divided. The land is cut both horizontally and vertically in the middle, creating four disconnected lands each with a size of N/2×N/2.
4. Repeat this until every *square* in a land has the same color. Or until it cannot be cut further.
5. At the end, Wonka takes the white pieces and Raymond takes the red. The person with more pieces wins. (Each land is counted as 1 piece.)

Example) After applying the described process above to the initially painted 8x8 land,



Wonka(white) has ended up with 11 and Raymond(red) has ended up with 11 pieces.

At step2-> step3 for example, the land on the upper left is cut since it contains both colors.



At step4, every separated land only contains a single color, so stop cutting.

Q) Implement a function that takes the length of a side of the land and the color of <u>each square</u> as input and outputs the final count of cutted pieces.

※ 1) To implement a function, you can use a recursive function.

2) One cut makes 4 area

3) In a function *t_conquer(x,y,N)*, x, y represent coordinates of the most upper left square, and we start using function at (x,y) = (0,0)

4) The most upper left square has (0,0) coordinates and the opposite end has (N-1, N-1)

[Test Case]

Test Case 1)

```
N = 8
territory = [[1,0,1,1,1,1,0,0],
             [1,0,1,1,1,1,0,0],
             [0,0,1,1,0,0,1,1],
             [0,0,1,1,0,0,1,0],
             [1,1,0,0,0,0,0,0],
             [1,1,0,0,0,0,0,0],
             [0,0,1,0,0,0,0,0],
             [0,0,0,1,0,0,0,0]]

l = Land()
color = l.t_conquer(0,0,N,territory)
color
```

```
[11, 11]
```

Test Case 2)

```
N = 4
territory = [[0,0,1,1],
             [1,1,1,1],
             [1,0,0,0],
             [0,0,1,1]]

l = Land()
color = l.t_conquer(0,0,N,territory)
color
```

```
[7, 6]
```

Test Case 3)

```
N = 16
territory = [[1,1,1,1,0,0,0,0,1,1,0,0,1,1,1,1],
             [1,1,1,1,0,0,1,1,1,1,0,1,1,1,1,1],
             [1,1,1,1,1,1,0,0,0,0,1,1,1,1,1,1],
             [1,1,1,1,1,1,0,0,0,0,1,1,1,1,1,1],
             [1,1,1,1,0,0,1,1,1,1,0,0,0,0,0,0],
             [0,1,1,1,1,0,1,1,1,1,0,0,0,0,0,0],
             [1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0],
             [1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0],
             [1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0],
             [1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0],
             [1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0],
             [1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0],
             [1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0],
             [1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0],
             [1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0],
             [1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0]]

l = Land()
color = l.t_conquer(0,0,N,territory)
color
```

```
[20, 20]
```

Problem 3. Rotating Array (rotate_array.c)

In this problem, your task is to rotate the given array for N times. The meaning of rotation is "sending the last component of the array to the first, and pushing the rest backward one by one".

Input Structure

- **The numbers listed in the command line are used to create an array and specify the rotation count. For example, if the input is 1 2 3 4 5 2, it means the array consists of [1, 2, 3, 4, 5] and it should be rotated 2 times.**

What You Need to Do:

- **You need to implement the code that can rotate the input array n times. The variables given below will be needed in the implementation.**

  int size – the length of array

  int rotate – the number of rotation

  int arr[ ] – the input array

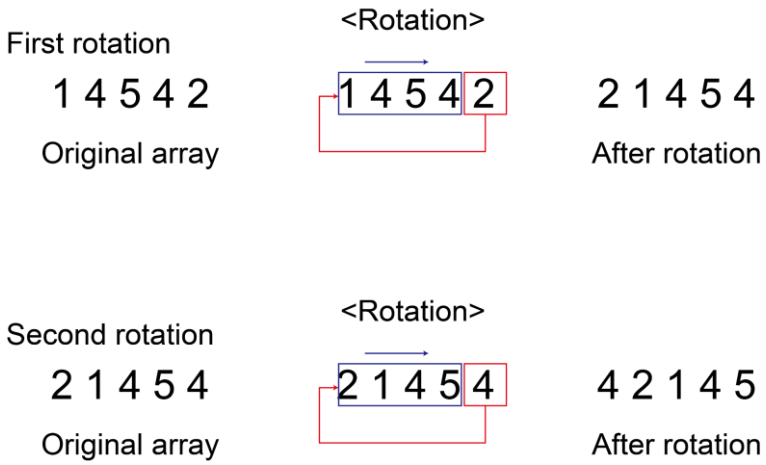
  int result[ ] – the output array you have to fill it.

  Do not modify the part that initializes the variables given above

- **For example, if you insert array 1 4 2 3 5 and rotation 2, which means you run your program with argument 1 4 2 3 5 2, then your program should print 3 5 1 4 2. The below image will make the instruction clear.**

Input: Array: 1 4 5 4 2 Rotation: 2

Output: 4 2 1 4 5



Test Case (The following examples assume your executable file name is rotate_array):

Sample 1)
./rotate_array 4 2 8 8 3
2 8 8 4

Sample 2)
./rotate_array 6 3 1 6
6 3 1

Problem 4. Dividing a Linked List (divide_linked_list.c)

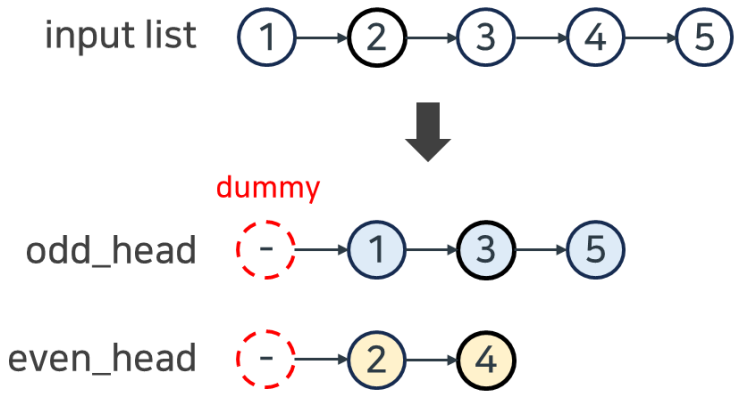
In this problem, you are tasked with dividing a given sorted linked list into two sorted lists: one for even values and one for odd values. The program will receive its data through command-line arguments, which will include the numerical values that make up the linked list. Your main objective is to write a function, divideList, that divides the given linked list into two lists.

Input Structure:

- **The numbers sorted and listed in the command-line are linked in order to form a linked list. For example, if the input is 1 2 3 4 5, they are connected as 1→2→3→4→5.**

What You Need to Do:

- **The code to implement the numbers listed in the command-line as a sorted linked list is already written. (Do not modify it!) You only need to write the divideList function, which takes the completed linked list as an argument.**
- **Within the divideList function, LinkedNode *odd_head and LinkedNode *even_head are pre-declared, pointing to dummy nodes. Nodes with odd values from the given linked list should be linked to 'odd_head', and nodes with even values should be linked to 'even_head'.**
- **In the code you write, 'odd_head' and 'even_head' must never change, and must still point to these dummy nodes at the end of the code. Therefore, create new pointers (for example, 'odd_curr', 'even_curr') to use, instead of directly moving 'odd_head' or 'even_head'.**



Test Case (The following examples assume your executable file name is divide_linked_list):

Sample 1)
./divide_linked_list 2 3 4
odd 3 even 2 4

Sample 2)
./divide_linked_list -7 -5 -3
odd -7 -5 -3 even