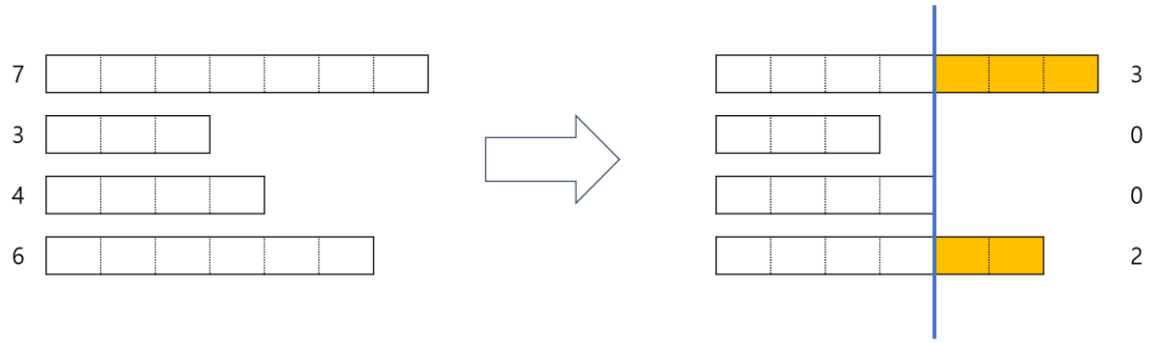
Assignment 02

[2023-2] 데이터사이언스를 위한 컴퓨팅의 기초 (M3239.005600 001)

Due: 2023 년 ***10*** 월 ***20*** 일 금요일 ***23*** 시 ***59*** 분

1. 양초 자르기 [25pts]

민지는 두께가 일정한 막대 양초를 여러 개 가지고 있다. 민지는 이를 녹여 새로운 모양의 양초를 만들려고 하는데, 고정된 길이를 잘라내는 절단기를 사용하여 잘린 양초의 자투리 부분을 사용할 예정이다. 즉, 양초가 절단기에 설정된 길이보다 길면 그 길이를 제외한 나머지 양을 사용하고, 짧으면 양초가 잘리지 않아 그 양초는 사용하지 않는다.



예를 들어, 길이가 각각 7, 3, 4, 6인 4개의 막대 양초가 있다고 하자. 이를 길이가 4로 설정된 절단기로 잘라내면 3, 0, 0, 2 만큼의 자투리가 발생하여 3 + 0 + 0 + 2 = 5 만큼의 양을 새로운 양초를 만들기 위해 녹이는 것이다.

절단기로 자를 양초들의 길이(candleList)와 새로운 양초를 만들기 위해 필요한 양초의 길이(n)가 주어졌을 때, 절단기에 설정할 길이를 구하는 함수를 작성하여라.

- 절단기는 한 번만 사용한다.
- 기존 막대 양초의 길이, 새로운 양초를 만들기 위해 필요한 양초의 길이, 절단기에 설정하는 길이는 모두 자연수이다.
- 절단기에 설정할 길이를 구할 수 없는 경우, 함수는 -1을 return한다.

※ Hint. 주어진 범위 내에서의 자연수를 search한다.

Implementation:

Test Case 1

```
[2]  #Check whether your function works well
     candleList = [20, 15, 10, 7]
     print(findLength(candleList, 5))
```

```
15
```

Test Case 2

```
[3]  #Check whether your function works well
     candleList = [7, 16, 32, 5, 20]
     print(findLength(candleList, 26))
```

```
14
```

Test Case 3

```
[4]  #Check whether your function works well
     candleList = [25, 17, 56]
     print(findLength(candleList, 36))
```

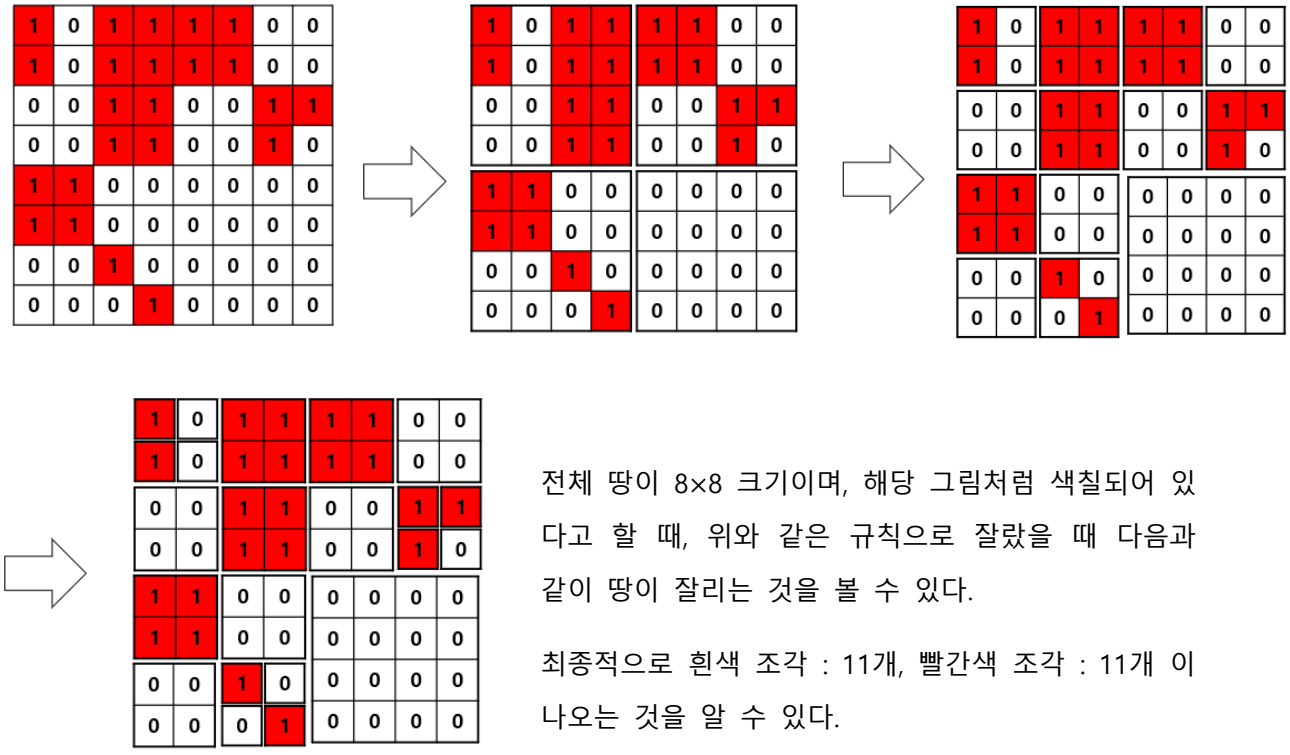
```
-1
```

2. 땅따먹기 [35pts]

철수와 영희가 땅따먹기 게임을 한다. 이 게임은 철수와 영희가 독자적으로 개발한 게임이라, 다음과 같은 규칙을 따른다.

1. 전체 땅의 크기는 N×N(N=2^k, $1 \le k \le 5$, k는 정수) 이다.
2. 전체 땅은 두 가지 색으로 무작위로 색칠 되어있다. (흰색, 빨간색이라 하자.)
3. 땅의 전체 영역이 같은 색으로 칠해져 있지 않다면, 땅을 잘라야 한다. 땅을 자를 때에는 가로와 세로 중간 부분을 잘라, 크기가 N/2×N/2 인 4개의 땅으로 만들어야 한다.
4. 이 과정을 반복하여 자른 땅의 영역의 색이 한 가지 색으로 통일될 때까지, 혹은 하나의 정사각형 땅이 되어 더 이상 자를 수 없을 때까지 반복한다.
5. 철수와 영희는 각자 흰색과 빨간색 조각을 가지고 간다. 이 조각의 수가 많은 사람이 승리한다.

그림으로 설명하면 다음과 같다.



전체 땅이 8×8 크기이며, 해당 그림처럼 색칠되어 있다고 할 때, 위와 같은 규칙으로 잘랐을 때 다음과 같이 땅이 잘리는 것을 볼 수 있다.

최종적으로 흰색 조각 : 11개, 빨간색 조각 : 11개 이 나오는 것을 알 수 있다.

입력으로 땅의 한 변의 길이와 각 정사각형 땅의 색이 주어질 때, 최종적으로 흰색 조각과 빨간색 조각이 몇 조각씩 나오는지 구하는 프로그램을 작성하시오.

※ Hint. 땅을 한 번 자를 때마다 4개의 작은 땅이 생긴다는 점에 주목하자.

함수 t_conquer(x,y,N)에서 x,y는 땅의 왼쪽 위 좌표를 의미한다.

처음 주어지는 땅의 왼쪽 위 좌표는 (0,0) 이며, 오른쪽 아래 좌표는 (N,N)이다.

Implementation:

Test Case1

[3]:
```
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]]

white, red = 0, 0
t_conquer(0,0,N)
print("흰색 : ", white)
print("빨간색 : ", red)
```

```
흰색 :  11
빨간색 :  11
```

Test Case2

[5]:
```
N = 4
territory = [[0,0,1,1],
             [1,1,1,1],
             [1,0,0,0],
             [0,0,1,1]]

white, red = 0, 0
t_conquer(0,0,N)
print("흰색 : ", white)
print("빨간색 : ", red)
```

```
흰색 :  7
빨간색 :  6
```

Test Case3

[6]:
```
N = 16
territory = [[1,1,1,1,0,0,0,0,1,1,0,0,1,1,1,1],
             [1,1,1,1,0,0,1,1,1,1,1,0,1,1,1,1],
             [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],
             [1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0],
             [1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0],
             [1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0],
             [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],
             [1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0]]

white, red = 0, 0
t_conquer(0,0,N)
print("흰색 : ", white)
print("빨간색 : ", red)
```

```
흰색 :  20
빨간색 :  20
```

3. Library [40pts]

아래 지시문에 따라 각 도서관의 책을 정렬하고 조건에 맞는 책을 찾는 코드를 완성하시오.

Instruction

1. Main Library, Engineering Library, Data Science Library 세 곳의 도서관에 들어올 책과 해당 책의 위치를 Library.txt 파일을 넣어두었다. 해당 파일의 행은 (책 이름, 작가 이름, 출판 연도, 책이 있는 도서관)으로 구성되어 있으며, 이 파일을 Input하여 진행한다. 이 파일을 임의로 수정하지 않는다.
2. Book, Library, LibraryManager 총 3개의 Class가 있으며, Book은 책에 대한 Class를, Library는 도서관에 책을 넣고, 정렬하며, 검색할 수 있는 Class를, LibraryManager는 모든 Library에서 원하는 책을 찾기 쉽게 만든 Class이다.
3. 파이썬 내장 함수인 sorted(), sort()s 함수를 이용하지 않는다.
4. Q1~Q4 모두 10pts이나, Q4를 틀려 파일을 불러오지 못하는 경우, Q1~Q3 모두 0점이 될 수 있으니 주의 바랍니다.

Quiz

Q3-1. 정렬 알고리즘을 활용하여 도서관 내 책들을 책 제목 순으로 정렬하는 sort_books_title 함수를 완성하시오. [10pts]

Q3-2. 도서관 내 책들 중 책 이름 혹은 작가 이름의 일부가 들어가면 검색 결과로 나오는 search_book 함수를 완성하시오. [10pts]

※ 대문자, 소문자 관계없이 해당 단어가 책 이름, 작가 이름에 포함되어 있는 책을 모두 가져와야 한다.

Q3-3. 전체 도서관에서 책 이름의 일부가 들어가면 검색 결과로 나오는 find_library_of_book 함수를 완성하시오. [10pts]

※ 대문자, 소문자 관계없이 해당 단어가 책 이름에 포함되어 있는 책을 모두 가져와야 한다.

Q3-4. Library.txt 파일을 가져와 books 이중 리스트에 넣는 함수를 완성하시오. [10pts]

Implementation:

Test Case 1

```
selected_library_name = "Data Science Library"
selected_library = manager.select_library(selected_library_name)

if selected_library:
    query = "Data"
    found_books = selected_library.search_book(query)

    if found_books:
        print(f"{selected_library_name}에서 검색 결과 ({query} 관련 도서):")
        for book in found_books:
            print(f"도서 제목: {book.title}")
            print(f"저자: {book.author}")
            print(f"출판연도: {book.year}")
            print('\n')
    else:
        print(f"{selected_library_name}에서 검색 결과가 없습니다.")
else:
    print(f"{selected_library_name}을 찾을 수 없습니다.")
```

```
Data Science Library에서 검색 결과 (Data 관련 도서):
도서 제목: Innovative systematic data-warehouse
저자: Kyle Knight
출판연도: 1933


도서 제목: Organized uniform database
저자: Anthony Moody
출판연도: 1981
```

Test Case2

```
selected_library_name = "Engineering Library"
selected_library = manager.select_library(selected_library_name)

if selected_library:
    query = "king"
    found_books = selected_library.search_book(query)

    if found_books:
        print(f"{selected_library_name}에서 검색 결과 ({query} 관련 도서):")
        for book in found_books:
            print(f"도서 제목: {book.title}")
            print(f"저자: {book.author}")
            print(f"출판연도: {book.year}")
            print('\n')
    else:
        print(f"{selected_library_name}에서 검색 결과가 없습니다.")
else:
    print(f"{selected_library_name}을 찾을 수 없습니다.")
```

```
Engineering Library에서 검색 결과 (king 관련 도서):
도서 제목: Enhanced fresh-thinking portal
저자: Tracy Smith
출판연도: 1977


도서 제목: Exclusive clear-thinking superstructure
저자: Suzanne Nelson
출판연도: 1984


도서 제목: Reduced cohesive alliance
저자: Darren King
출판연도: 1986
```

Test Case3

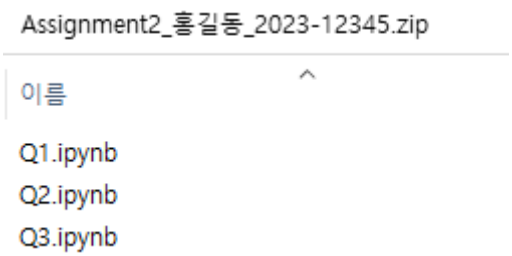
```
book_title_to_find = "Open"
matching_books = manager.find_library_of_book(book_title_to_find)

if matching_books:
    print(f"검색 결과 ({book_title_to_find} 관련 도서):")
    for book_title, library in matching_books.items():
        print(f"'{book_title}' 책은 '{library.name}' 도서관에 있습니다.")
else:
    print(f"'{book_title_to_find}' 관련 도서를 찾을 수 없습니다.")
```

```
검색 결과 (Open 관련 도서):
'Open-architected transitional model' 책은 'Main Library' 도서관에 있습니다.
'Distributed even-keeled open architecture' 책은 'Engineering Library' 도서관에 있습니다.
'Open-source bifurcated intranet' 책은 'Engineering Library' 도서관에 있습니다.
'Open-source client-driven methodology' 책은 'Engineering Library' 도서관에 있습니다.
'Secured responsive open system' 책은 'Engineering Library' 도서관에 있습니다.
'Configurable 6thgeneration open architecture' 책은 'Main Library' 도서관에 있습니다.
'Open-source exuding open architecture' 책은 'Engineering Library' 도서관에 있습니다.
'Adaptive human-resource open architecture' 책은 'Main Library' 도서관에 있습니다.
'Networked dynamic open system' 책은 'Main Library' 도서관에 있습니다.
'Open-source needs-based monitoring' 책은 'Data Science Library' 도서관에 있습니다.
'Open-source responsive workforce' 책은 'Engineering Library' 도서관에 있습니다.
'Open-source demand-driven open system' 책은 'Main Library' 도서관에 있습니다.
'Optimized motivating open system' 책은 'Main Library' 도서관에 있습니다.
'Adaptive attitude-oriented open system' 책은 'Main Library' 도서관에 있습니다.
```

Cautions

1. 코드 채점 시에는, 예시로 주어진 test case 외에 추가적인 test case 들을 활용하여 채점할 예정으로 edge case 들에 대한 검증을 꼼꼼히 하고 제출하기 바랍니다.

2. 작성한 ipynb 파일을 Assignment2_학생이름_학번.zip 로 압축하여 제출하기 바랍니다. (아래 예시 참고) 파일명을 바꾸시면 안됩니다.



3. 제출 양식 미 준수 시 채점 간 오류가 발생할 수 있습니다. 제출 양식 미 준수로 인한 오류 교정은 없습니다. 수강생 분들은 제출 양식을 반드시 확인 바랍니다.