Searching and Sorting Questions

Prelim 2021

1. Name your Jupyter Notebook as

   `Task1_<your name>_<centre number>_<index number>.ipynb`

   For each of the sub-tasks, add a comment statement, at the beginning of the code using the hash symbol '#', to indicate the sub-task the program code belongs to, for example:

```
In[1]:  # Task 1.1
        Program Code
```

```
In[2]:  # Task 1.2
        Program Code
```

```
In[3]:  # Task 1.3
        Program Code
```

```
        Output:
```

Task 1.1

The file `INTEGERS.txt` stores 100 integers. Write a program to read the integers, arrange them in ascending order using quick sort, and write the sorted integers to a file called

`SORTED_<your name>_<centre number>_<index number>.txt` [15]

Task 1.2

Write a function `BinarySearch(list_of_integers, target)` that

- takes a list of ascending integers, `list_of_integers` and an integer `target`
- performs a binary search
- prints out if `target` is found in `list_of_integers`
- returns the number of comparisons during the binary search [8]

Task 1.3

Write a program to read the list of integers from

`SORTED_<your name>_<centre number>_<index number>.txt`

obtained in Task 1.1. Generate 50 random integers between 1 and 200 (inclusive) and perform a binary search for each of these random integers in this sorted list. Output the average number of comparisons of these 50 binary searches. [2]

Save your Jupiter Notebook for Task 1.

Prelim 2022

2. A school library club polled its members and created a list of 18 recommended classics. The list is saved in a file named `booklist.csv` stating the book title, author and year of publication.

   For each of the sub-tasks, add a comment statement at the beginning of the code using the hash symbol '#' to indicate the sub-task the program code belongs to, for example:

```
In [1]:
    # Task 2.1
    Program code
    Output:
```

Task 2.1

Write a function `read_csv(filename)` which

- reads a csv file, `filename`
- stores the records in an array with 3 columns `title`, `author` and `year`
- `title`, `author` and `year` all stored as strings
- returns the array

Use the following code to test your function.

```
books_array = read_csv("booklist.csv")
print(len(books_array))
print(books_array)
```

[4]

Task 2.2

As the data is not suitably ordered, a sort algorithm is required.

Write a function to implement a bubble sort algorithm `bubble(array, sort_key)` which sorts `array` by the `sort_key` provided.

- The `sort_key` should be one of the values `"title"`, `"author"` or `"year"`
- The function will return the sorted `array` in ascending order
- The function will give a return of `-1` if an invalid `sort_key` is provided

Use the following code to test your function.

```
print(bubble(books_array, "title"))
print(bubble(books_array, "ISBN"))
```

[6]

Task 2.3

When the same algorithm was used on the entire library catalogue, it was deemed to be too slow.

Write a function to implement a merge sort algorithm `merge(array, sort_key)` which sorts `array` by the `sort_key` provided.

- The `sort_key` should be one of the values `"title"`, `"author"` or `"year"`
- The function will return the sorted `array` in ascending order
- The function will give a return of `-1` if an invalid `sort_key` is provided

Use the following code to test your function.

```
print(merge(books_array, "author"))
print(merge(books_array, "year"))
```

[7]

Task 2.4

Write a function `reverse(array)` which reverses the order of the array, without the use of any built-in functions.

Use the following code to test your function.

```
print(reverse([1,3,5,2,4]))
print(reverse([1,9,6,4]))
```

[3]

Task 2.5

The library bought some new books and they are recorded in the file `newbooks.csv`. Using the function(s) you have written, write a procedure which

- uses bubble sort to sort the new books starting with the most recent publication, and
- saves the sorted new books array in a csv

Save your csv file as

`YEAR_<your name>_<center number>_<index number>.csv` [5]

Save your Jupyter notebook for Task 2.

Prelim 2023

2. Name your Jupyter Notebook as:

`TASK2_<your name>_<centre number>_<index number>.ipynb`

This task is to perform sorting algorithms on `Person` objects held in a 1-dimensional array.

For each of the sub-tasks, add a comment statement at the beginning of the code, using the hash symbol '#' to indicate the sub-task the program code belongs to, for example:

In [1]:
```
# Task 2.1
Program code
```

`Output`:

Task 2.1

The class `Person` contains two properties:

- `name` - stored as a string
- `age` - stored as an integer

Write program code to declare the class `Person` and its constructor and `print()` method to output the `name` and `age` of a `Person` object.

[2]

Task 2.2

Write a function `task2_2(filename)` that:

- takes a string `filename` which represents the name of a text file
- reads in the contents of the text file
- returns the content as a list of `Person` objects.

Call the function `task2_2` with the file `PERSON.txt` and print `Person` objects using the following statements:

```
list_of_person = task2_2('PERSON.txt')
for person in list_of_person:
    person.print()
```

[4]

Task 2.3

One method of sorting is the insertion sort.

Write a function `task2_3(list_of_person, key, order)` that:

- accepts three parameters:
  - `list_of_person` contains a list of `Person` objects
  - `key` should be one of the values:
    - `name` – list to be sorted by name
    - `age` – list to be sorted by age
  - `order` should be one of the values:
    - `asc` – list to be sorted by `key` in ascending order
    - `desc` – list to be sorted by `key` in descending order
- sorts `list_of_person` by `key` in `order` using **insertion sort**.

Call the function `task2_3` with the contents of the file `PERSON.txt` and print the sorted `Person` objects using the following statements:

```
list_of_person = task2_2('PERSON.txt' )
task2_3(list_of_person, 'name', 'asc')
for person in list_of_person:
    person.print()
```

[8]

Task 2.4

Another method of sorting is the quick sort.

Write a function `task2_4(list_of_person, key, order)` that:

- accepts three parameters:
  - `list_of_person` contains a list of `Person` objects
  - `key` should be one of the values:
    - `name` – list to be sorted by name
    - `age` – list to be sorted by age
  - `order` should be one of the values :
    - `asc` – list to be sorted by `key` in ascending order
    - `desc` – list to be sorted by `key` in descending order
- sorts `list_of_person` by `key` in `order` using **quick sort**.

Call the function `task2_4` with the contents of the file `PERSON.txt` and print the sorted `Person` objects using the following statements:

```
list_of_person = task2_2('PERSON.txt')
task2_4(list_of_person, 'age', 'desc')
for person in list_of_person:
    person.print()
```

[8]

Task 2.5

Write a function `task2_5(list_of_person, method, key, order)` that:

- accepts four parameters:
  - `list_of_person` contains a list of `Person` objects
  - `method` should be one of the values:
    - `insertion sort` – sort the list using insertion sort
    - `quick sort` – sort the list using quick sort
  - `key` should be one of the values:
    - `name` – list to be sorted by name
    - `age` – list to be sorted by age
  - `order` should be one of the values:
    - `asc` – list to be sorted by `key` in ascending order
    - `desc` – list to be sorted by `key` in descending order
- sorts `list_of_person` by `key` in `order` using `method.`

Call the function `task2_5` with the contents of the file `PERSON.txt` and print the sorted `Person` objects using the following statements:

```
list_of_person = task2_2('PERSON.txt')
task2_5(list_of_person, 'quick sort', 'name', 'desc')
for person in list_of_person:
    person.print()
```

[2]

Save your Jupyter Notebook for Task 2.

Solution

Prelim 2021

Task 1.1

```
def split(L, low, high):
    mid = (low + high) // 2
    pivot = L[mid]
    L[low], L[mid] = L[mid], L[low]
    left = low + 1
    right = high

    while left <= right:
        while left <= right and L[left] <= pivot:
            left += 1

        while L[right] > pivot:
            right -= 1

        if left < right:
            L[left], L[right] = L[right], L[left]

    pos = right
    L[low] = L[right]
    L[pos] = pivot


    return pos


def quicksort(L, low, high):
    if low < high:
        pivot = split(L, low, high)
        quicksort(L, low, pivot - 1)
        quicksort(L, pivot + 1, high)

L = []
with open('INTEGERS.txt', 'r') as f:
    for line in f:
        L.append(int(line.strip()))
quicksort(L, 0, len(L) - 1)

filename = 'SORTED.txt'

with open(filename, 'w') as g:
    for item in L:
```

```
        g.write(str(item) + '\n')
```

Task 1.2

```
def BinarySearch(L, target):

    count = 0
    low = 0
    high = len(L) - 1
    found = False

    while not found and low <= high:
        mid = (low + high) // 2
        count += 1
        if L[mid] == target:
            found = True
        elif L[mid] < target:
            low = mid + 1
        else:
            high = mid - 1


    if found:
        print(target, 'is found')
    else:
        print(target, "is not found!")

    return count
```

Task 1.3

```
import random
L = []
with open('SORTED.txt', 'r') as f:
    for line in f:
        L.append(int(line.strip()))

avg = 0

for i in range(50):
    target = random.randint(1, 200)
    avg += BinarySearch(L, target)

print("Average comparison:", avg / 50)
```

Prelim 2022

Task 2 Soln:

Overall comments:

students are advised NOT to add additional printout lines in the output. if print() was added or debug print-statements were made, please comment them out and run again before submission.

While opening csv is within syllabus, writing csv with the csv package is confusing and students are encouraged to treat csv files as normal text files.

```
# Task 2.1
def read_csv(filename):
    books_file = open(filename, "r")
    book_str = books_file.read()
    book_list = book_str.split("\n")
    array = []
    for book in book_list:
        title, author, year = book.split(",")
        array.append([title, author, year])
    books_file.close()
    return array

books_array = read_csv("booklist.csv")
print(len(books_array))
```

```
import csv
# Task 2.1 (alternative by csv package)
def read_csv(filename):
    books_file = open(filename, "r", encoding="utf-8")
    book_list = csv.reader(books_file, delimiter=",")
    array = []
    for book in book_list:
```

```
        title, author, year = book
        array.append([title, author, year])
    books_file.close()
    return array


books_array = read_csv("booklist.csv")
print(len(books_array))
print(books_array)
```

```
18
[['White Fang', 'Jack London', '1906'], ['The Wind in the
Willows', 'Kenneth Grahame', '1908'], ['Moby Dick', 'Herman
Melville', '1851'], ['Jane Eyre', 'Charlotte Bronte', '1847'],
['The Picture of Dorian Gray', 'Oscar Wilde', '1890'], ['The
Three Musketeers', 'Alexandre Dumas', '1844'], ['Persuasion',
'Jane Austen', '1817'], ['Dream of the Red Chamber', 'Cao
Xueqin', '1791'], ['Little Women', 'Louisa May Alcott',
'1868'], ['The Phantom of the Opera', 'Gaston Leroux', '1909'],
['Water Margin', 'Shi Naian', '1450'], ['A Christmas Carol',
'Charles Dickens', '1843'], ['One Hundred Years of Solitude',
'Gabriel Garcia Marquez', '1967'], ['Nineteen Eighty-Four',
'George Orwell', '1949'], ['Journey to the West', 'Wu Chengen',
'1592'], ['Romance of the Three Kingdoms', 'Luo Guanzhong',
'1522'], ['Fahrenheit 451', 'Ray Bradbury', '1953'], ['War and
Peace', 'Leo Tolstoy', '1867']]
```

Marker's Comments:

Task 2.1

- Different students approached this part differently with most storing the 2D-array as a list of list, dictionaries, 3-tuples, etc…

```
# Task 2.2
def bubble(array, sort_key):
    sort_dict = {"title": 0, "author": 1, "year": 2}
    if sort_key not in sort_dict:
```

```
        return -1
    s_index = sort_dict[sort_key]
    n = len(array)
    for i in range(n-1):
        for j in range(n-1):
            if array[j][s_index] > array[j+1][s_index]:
                array[j], array[j+1] = array[j+1], array[j]
    return array


print(bubble(books_array,"title"))
print(bubble(books_array,"ISBN"))
```

```
[['A Christmas Carol', 'Charles Dickens', '1843'], ['Dream of the Red
Chamber', 'Cao Xueqin', '1791'], ['Fahrenheit 451', 'Ray Bradbury',
'1953'], ['Jane Eyre', 'Charlotte Bronte', '1847'], ['Journey to the
West', 'Wu Chengen', '1592'], ['Little Women', 'Louisa May Alcott',
'1868'], ['Moby Dick', 'Herman Melville', '1851'], ['Nineteen Eighty-
Four', 'George Orwell', '1949'], ['One Hundred Years of Solitude',
'Gabriel Garcia Marquez', '1967'], ['Persuasion', 'Jane Austen',
'1817'], ['Romance of the Three Kingdoms', 'Luo Guanzhong', '1522'],
['The Phantom of the Opera', 'Gaston Leroux', '1909'], ['The Picture of
Dorian Gray', 'Oscar Wilde', '1890'], ['The Three Musketeers',
'Alexandre Dumas', '1844'], ['The Wind in the Willows', 'Kenneth
Grahame', '1908'], ['War and Peace', 'Leo Tolstoy', '1867'], ['Water
Margin', 'Shi Naian', '1450'], ['White Fang', 'Jack London', '1906']]

-1
```

Marker's Comments:

Task 2.2

- common mistake: while most knew that the correct index was to be compared, many did not swap the entire book and only swapped the author.

```python
# Task 2.3
def merge(array, sort_key):
    sort_dict = {"title": 0, "author": 1, "year": 2}
    if sort_key not in sort_dict:
        return -1
    s_index = sort_dict[sort_key]

    if len(array)<2:
        return array

    mid = len(array) // 2
    left = merge(array[:mid],sort_key)
    right = merge(array[mid:],sort_key)

    merged = []
    while len(left) and len(right):
        if left[0][s_index] <= right[0][s_index]:
            merged = merged + [left.pop(0)]
        else:
            merged = merged + [right.pop(0)]

    merged = merged + left + right

    for i in range(len(array)):
        array[i] = merged[i]

    return array

print(merge(books_array,"author"))
print(merge(books_array,"year"))
```

```
[['The Three Musketeers', 'Alexandre Dumas', '1844'], ['Dream of the Red Chamber', 'Cao Xueqin', '1791'], ['A Christmas Carol', 'Charles Dickens', '1843'], ['Jane Eyre', 'Charlotte Bronte', '1847'], ['One Hundred Years of Solitude', 'Gabriel Garcia Marquez', '1967'], ['The Phantom of the Opera', 'Gaston Leroux', '1909'], ['Nineteen Eighty-Four', 'George Orwell', '1949'], ['Moby Dick', 'Herman Melville', '1851'], ['White Fang', 'Jack London', '1906'], ['Persuasion', 'Jane Austen', '1817'], ['The Wind in the Willows', 'Kenneth Grahame', '1908'], ['War and Peace', 'Leo Tolstoy', '1867'], ['Little Women', 'Louisa May Alcott', '1868'], ['Romance of the Three Kingdoms', 'Luo Guanzhong', '1522'], ['The Picture of Dorian Gray', 'Oscar Wilde', '1890'], ['Fahrenheit 451', 'Ray Bradbury', '1953'], ['Water Margin', 'Shi Naian', '1450'], ['Journey to the West', 'Wu Chengen', '1592']]
```

```
[['Water Margin', 'Shi Naian', '1450'], ['Romance of the Three Kingdoms', 'Luo Guanzhong', '1522'], ['Journey to the West', 'Wu Chengen', '1592'], ['Dream of the Red Chamber', 'Cao Xueqin', '1791'], ['Persuasion', 'Jane Austen', '1817'], ['A Christmas Carol', 'Charles Dickens', '1843'], ['The Three Musketeers', 'Alexandre Dumas', '1844'], ['Jane Eyre', 'Charlotte Bronte', '1847'], ['Moby Dick', 'Herman Melville', '1851'], ['War and Peace', 'Leo Tolstoy', '1867'], ['Little Women', 'Louisa May Alcott', '1868'], ['The Picture of Dorian Gray', 'Oscar Wilde', '1890'], ['White Fang', 'Jack London', '1906'], ['The Wind in the Willows', 'Kenneth Grahame', '1908'], ['The Phantom of the Opera', 'Gaston Leroux', '1909'], ['Nineteen Eighty-Four', 'George Orwell', '1949'], ['Fahrenheit 451', 'Ray Bradbury', '1953'], ['One Hundred Years of Solitude', 'Gabriel Garcia Marquez', '1967']]
```

```python
# Task 2.4

def reverse(array):
    length = len(array)
    mid = length // 2
    for i in range(mid):
        array[i], array[length-1-i] = array[length-1-i], array[i]
    return array


print(reverse([1,3,5,2,4]))
print(reverse([1,9,6,4]))
```

```
[4, 2, 5, 3, 1]
```

```
[4, 6, 9, 1]
```

```
# Task 2.5
arr = read_csv("newbooks.csv")
merge(arr,"year")
reverse(arr)

new_csv = open("YEAR_name_ct.csv", "w")

book_str = []
for book in arr:
    book_str.append(",".join(book))
ret_str = "\n".join(book_str)
new_csv.write(ret_str)
new_csv.close()
```

Marker's Comments:

Task 2.5

- no concerns for students who used default file writing (as in solution)
- for students who used the csv package, it is worth noting that `csv.writer()` appends `'\r\n'` to the end of line. Hence, either an extra parameter of `lineterminator='\n'` is needed, or the file must be opened with `quotechar=""`
    - students are advised to check their output file if time allows
- a few students created the file with a different name from the one required. it should be created with the name in the format required

Prelim 2023

```
#Task2_1 –
class Person:
    def __init__(self, name, age):
        self._name=name
        self._age=age

    def getName(self):
        return self._name

    def getAge(self):
        return self._age

    def print(self):
        print (f"{self._name}, {self._age}")
```

```
#Task2_2 [4M]

def task2_2(filename):
    lst=[]
    f = open(filename, 'r')
    for line in f:
        name, age= line.split(',')
        lst.append(Person(name.strip(), int(age.strip())))
    f.close()
    return lst
```

```
#print(task2_2('person.txt'))


list_of_person = task2_2('PERSON.txt')
for person in list_of_person:
    person.print()
```

```
Alice, 18
Bob, 20
Charlie, 17
David, 16
Emily, 19
Austin, 19
Cole, 20
Adam, 16
Benjamin, 16
Chloe, 19
Daniel, 19
Eva, 20
Bailey, 18
Daisy, 18
Amelia, 17
Brian, 19
Catherine, 18
Dylan, 17
Eleanor, 16
Bella, 17
Caleb, 16
Delilah, 20
Ethan, 17
Ella, 18
Arthur, 20
```

```
# Task2_3 insertion sort

def task2_3(list_of_person, key, order):
    n=len(list_of_person)
    for i in range(1, n):
        target = list_of_person[i]
        j = i-1
        if key =="name":
            if order =='asc':
                while j >= 0 and
                     (target.getName() < list_of_person[j].getName()):
                    list_of_person[j+1] = list_of_person[j]
                    j -= 1
            else:
                while j >= 0 and
                        (target.getName() > list_of_person[j].getName()):
                    list_of_person[j+1] = list_of_person[j]
                    j -= 1
        elif key =="age":
            if order=='asc':
                while j >= 0 and
                        (target.getAge() < list_of_person[j].getAge()):
                    list_of_person[j+1] = list_of_person[j]
                    j -= 1
            else:
                while j >= 0 and
                        (target.getAge() > list_of_person[j].getAge()):
                    list_of_person[j+1] = list_of_person[j]
                    j -= 1
```

```
        list_of_person[j+1] = target



list_of_person=task2_2('PERSON.txt')
task2_3(list_of_person, 'name', 'asc')
for person in list_of_person:
    person.print()
```

```
Adam, 16
Alice, 18
Amelia, 17
Arthur, 20
Austin, 19
Bailey, 18
Bella, 17
Benjamin, 16
Bob, 20
Brian, 19
Caleb, 16
Catherine, 18
Charlie, 17
Chloe, 19
Cole, 20
Daisy, 18
Daniel, 19
David, 16
Delilah, 20
Dylan, 17
Eleanor, 16
Ella, 18
Emily, 19
```

```
Ethan, 17
Eva, 20
```

```
# Task2_4
#Quicksort

def quicksort(list_of_person, low, high, key, order):
    # perform a recursive quicksort
    # sorting the range [low,high], inclusive of both ends

    if low < high:   # list has more than one element
        # partition into two sublists, pos is partitioning index
        pos = partition(list_of_person, low, high, key, order)

        # separately sort elements before partition and after partition
        quicksort(list_of_person, low, pos-1, key, order)
        quicksort(list_of_person, pos+1, high, key, order)

    # else list has 0 or 1 element and requires no sorting

def partition(list_of_person, low, high, key, order):
    # partition list into two sublists
    # re-arrange the list so that the pivot is properly partitioned
    i =  low   # boundary index
    pivot = list_of_person[high]
    for j in range(low, high):
        # if current element is smaller than the pivot, move it to the left
        if key == 'name':
            if order == 'asc':
                if list_of_person[j].getName() < pivot.getName():
```

```
                    list_of_person[i], list_of_people[j]
                        = list_of_person[j], list_of_person[i]
                    #swap cur element with the element at boundary index
                    i = i + 1      # increment boundary index
            else:
                if list_of_person[j].getName() > pivot.getName():
                    list_of_person [i], list_of_person[j]
                        = list_of_person[j], list_of_person[i]
                    #swap cur element with the element at boundary index
                    i = i + 1      # increment boundary index

        elif key == 'age':
            if order == 'asc':
                if list_of_person[j].getAge() < pivot.getAge():
                    list_of_person[i], list_of_person[j]
                        = list_of_person[j], list_of_person[i]
                    #swap cur element with the element at boundary index
                    i = i + 1      # increment boundary index
            else:
                if list_of_person[j].getAge() > pivot.getAge():
                    list_of_person[i], list_of_person[j]
                        = list_of_person[j], list_of_person[i]
                    # swap cur element with the element at boundary index
                    i = i + 1      # increment boundary index

    # end of FOR loop; place pivot in the correct position at index i
    list_of_person[i], list_of_person[high]
         = list_of_person[high], list_of_person[i]
    return i       # final position of pivot

def task2_4(list_of_person, key, order):
```

```
    quicksort(list_of_person, 0, len(list_of_person)-1, key, order)



list_of_person=task2_2('PERSON.txt')
task2_4(list_of_person, 'age', 'desc')
for person in list_of_person:
    person.print()
```

```
Arthur, 20
Delilah, 20
Bob, 20
Eva, 20
Cole, 20
Emily, 19
Chloe, 19
Daniel, 19
Austin, 19
Brian, 19
Alice, 18
Ella, 18
Bailey, 18
Daisy, 18
Catherine, 18
Charlie, 17
Amelia, 17
Dylan, 17
Bella, 17
Ethan, 17
Benjamin, 16
Caleb, 16
```

David, 16

Eleanor, 16

Adam, 16

```
#Task 2.5 [2M]


def task2_5(list_of_person, method, key, order):
    if method=='insertion sort':
       task2_3(list_of_person, key, order)
    elif method=='quick sort':
       task2_4(list_of_person, key, order)



list_of_person=task2_2('PERSON.txt')
task2_5(list_of_person,'quick sort','name','desc')
for person in list_of_person:
    person.print()
```

```
Eva, 20
Ethan, 17
Emily, 19
Ella, 18
Eleanor, 16
Dylan, 17
Delilah, 20
David, 16
Daniel, 19
Daisy, 18
Cole, 20
Chloe, 19
Charlie, 17
Catherine, 18
Caleb, 16
Brian, 19
```

```
Bob, 20
Benjamin, 16
Bella, 17
Bailey, 18
Austin, 19
Arthur, 20
Amelia, 17
Alice, 18
Adam, 16
```