Sorting

Bubble, Selection, Insertion sort already covered

Assumption till now

- TC: O(N log N)
- SC: O(1)

```
Figure out the TC and SC for your langauge's sorting implementation
JAVA:
C++
Python
Javascript
C#
```

In [ ]:
```
1
```

comparison based sort

1. Bubble sort
2. selection Sort
3. Insertion Sort
4. Merge sort: Merge 2 sorted arrays **[Divide and Conquer]**
5. Quick sort **[Divide and Conquer]**
6. Heap sort: Heap DS

Non-comparison

- Counting sort
- Radix sort
- Bucket sort

Counting Sort

TC: max(O(N), O(range))
SC: O(range)

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

[10,15]

[min, max]
- create a frequency array of size max + 1
- count frequency of each value in the range
- overwrite the data in original array using frequency map

Frequency array
[0 1 3 1 2 1]
 0 1 2 3 4 5

Sorted result
[1 2 2 2 3 4 4 5]

```
1 - Stable vs Unstable
2 - Inplace vs not inplace
```

In [ ]:
```
1
```

Merge sort

- merging 2 sorted arrays
- assuming an array of size 1 is always sorted

```
1 4 5 2 3 5 7 6
0 1 2 3 4 5 6 7

Is an array of size 1 sorted ?
[1]

[1] [4] [5] [2] [3] [5] [7] [6]
[1,4]    [2,5]   [3,5]  [6,7]
 [1,2,4,5]         [3,5,6,7]
       [1 2 3 4 5 5 6 7]
```

```
def merge_sort(arr):

    if (len(arr) <= 1):
        return arr;

    a1 = merge_sort(arr[0:len(arr)//2]) # recurse for first half of a
rray
    a2 = merge_sort(arr[len(arr)//2: len(arr)]) # recurse for second
half of array

    a3 = merge_sorted_arrays(a1, a2)
    return a3

def merge_sorted_arrays(a1, a2):

    i = 0
    j = 0
    a3 = []
    while i < len(a1) and j < len(a2):
        if a1[i] < a2[j]:
            a3.append(a1[i])
            i+=1
        else:
            a3.append(a2[i])
            j+=1

    while i < len(a1):
        .......
    return a3
```

```
[1 4 5 2 3 5 7 6] -> [1 2 3 4 5 5 6 7]
```

```
N ops
      [1 4 5 2]->[1, 2, 4, 5]                                    [3 5 7
6]->[3 5 6 7]                  N ops
         [1 4]->[1,4]           [5,2]->[2,5]                    [3,5]->
[3,5]         [7,6]->[6,7]     N ops
           [1]->[1] [4]->[4]    [5]->[5]  [2]->[2]              [3]->
[3]  [5]->5   [7]->[7]  [6]->[6] N ops

No. of levels = recursion depth = log n
N + N + N +.... (log N times)

TC: O(n log n)
SC: O(n)
```

In [ ]:

```
1
```

External Sort : uses merge sort algorithm

```
RAM: 4G
Data: 32G file.txt

file1.txt (4GB), file2.txt, file3.txt..... file8.txt
f1                f2

  file12     file34 .....  file78
      file1234   file5678
         file12345678

- open -> file pointer
- read from file : read data in chunks and move pointer further.


    file1 [10 20 40 50 80]  file2 [20 30 60]
           ^                        ^
    file3 10 20 20 30


out.txt write(1)  write(2)  write(4) [4 5 6]

[1 2 3 4 5 6]
File O(1)


RAM: random access ? Yes
Disk HDD: random access ? No
Disk SSD: random access ? Yes
```

In [ ]:

```python
def merge_files(name1, name2):
    f1 = open(name1, 'r')
    f2 = open(name2, 'r')

    f3 = open('temp.txt', 'w')

    while True:
        d1 = f1.readline()
        d2 = f2.readline()

        if d1 == '' || d2 == '':
            break

        if d1 > d2:
            f3.write(d2)
        else:
            f3.write(d1)

    if d1 == '':
        while True:
            d = f2.readline()
            if d == '':
                break
            f3.write(d)

    if d2 == '':
        while True:
            d = f1.readline()
            if d == '':
                break
            f3.write(d)

```

In [ ]:

```python

```

quick sort

Properties: inplace, not-stable

- TC
  - AVG = BEST = O(N log N)
  - Worst = O(N^2)
- SC
  - AVG = O(log N)
  - WORST = O(N)

```
void qsort(array, start, end) {
    if end-start <=1
        return

    p = partition(array, start, end)

    qsort(array, start, p)
    qsort(array, p, end)
}


int parition(array, start, end) {
    // returns the point of partition

    pivot = end-1
    i =  start-1
    j = start
    while(j<pivot) {
        if (array[j] < array[pivot]) {
            i+=1
            swap(array[i], array[j])
        }
        j++
```

In [ ]:
```
- numbers
- strings
```

In [ ]:
```
```

Language Specifics: Comparator Sorting

C++

```cpp
// Type your code here, or load an example.
# include <vector>
# include <algorithm>
# include <string>

using namespace std;
class Person {

    public:
        int age;
        string name;

        Person(string name, int age) {
            this->age = age;
            this->name = name;
        }

        bool operator<(Person other) const {
            if (this->age < other.age) {
                return true;
            }
            return false;
        }

        bool operator>(Person other) const {
            if (this->age > other.age) {
                return true;
            }
            return false;
        }
};

int main() {
    std::vector<Person> persons;
    persons.push_back(Person("A", 2));
    persons.push_back(Person("C", 3));
    persons.push_back(Person("B", 1));
    persons.push_back(Person("A", 4));
```

````
**Java**
```Java
import java.util.*;

class Person implements Comparable<Person> {
    public String name;
    public int age;

    Person(String name, int age) {
        this.name = name;
        this.age = age;
````
````

```
    }

    @Override
    public int compareTo(Person p) {
        if (this.age < p.age) {
            return -1;
        } else if (this.age == p.age) {
            return 0;
        }

        return 1;
    }

}

class CompByName  implements Comparator<Person> {

    public int compare(Person a, Person b)
    {
        return a.name.compareTo(b.name);
    }
}

class CompByNameThenAge  implements Comparator<Person> {

    public int compare(Person a, Person b)
    {
        int val = a.name.compareTo(b.name);
        if (val == 0) {
           if (a.age > b.age) {
                return -1;
            } else if (a.age == b.age) {
                return 0;
            }

            return 1;
        }

        return val;
    }
}


class CompByAge  implements Comparator<Person> {

    public int compare(Person a, Person b)
    {
        if (a.age < b.age) {
            return -1;
        } else if (a.age == b.age) {
            return 0;
        }

        return 1;
    }
}

```

```
public class Main {
    public static void main(String[] args) {

        ArrayList<Person> list = new ArrayList<>();

        list.add(new Person("A", 2));
        list.add(new Person("C", 3));
        list.add(new Person("B", 1));
        list.add(new Person("A", 4));

        // Collections.sort(list);
        Collections.sort(list, new CompByNameThenAge());

        for (int i=0; i < list.size(); i++) {
            System.out.println(list.get(i).name + " " +
 list.get(i).age);
        }

    }

}
```

In [ ]:
```
lessthan(a1,a2):
    return a1 < a2

lessthan(1,2) -> true
lessthan(2,2) -> false

x = 3
y = 2
lessthan(x,y) -> false so is x==y or is x>y
    lessthan(y,x) -> true -> x > y
    lessthan(y,x) -> false -> equal
```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```

Question
https://leetcode.com/problems/largest-number/ (https://leetcode.com/problems/largest-number/)

TC: O(n log n)
SC: O(1)

```
// Type your code here, or load an example.
# include <vector>
# include <algorithm>
# include <string>

using namespace std;
class Person {

    public:
        int age;
        string name;

        Person(string name, int age) {
            this->age = age;
            this->name = name;
        }

        bool operator<(Person other) const {
            if (this->age < other.age) {
                return true;
            }
            return false;
        }

        bool operator>(Person other) const {
            if (this->age > other.age) {
                return true;
            }
            return false;
        }
};

int main() {
    std::vector<Person> persons;
    persons.push_back(Person("A", 2));
    persons.push_back(Person("C", 3));
    persons.push_back(Person("B", 1));
    persons.push_back(Person("A", 4));

    //a<b   -> a.func(b)

    // std::sort(persons.begin(), persons.end());
    // std::sort(persons.begin(), persons.end(), std::greater<Person>
());
    std::sort(persons.begin(), persons.end(), [](Person a, Person b)
{
        return a.age > b.age;
    });
```

```
    // for(auto p: persons) {
    //     std::cout << p.name << " " << p.age << endl;
    // }


    std::vector<int> nums = {3,30,34,5,9};
    // std::sort(nums.begin(), nums.end(), [](int n1, int n2) {
    //     auto s1 = to_string(n1);
    //     auto s2 = to_string(n2);
    //     return s1 > s2;
    // });

    std::sort(nums.begin(), nums.end(), [](int n1, int n2) {
        auto s1 = to_string(n1);
        auto s2 = to_string(n2);
        return s1+s2 > s2+s1;
    });

    sort(a1, a2, less())
#     3 5 9 30 34 // assume bubble sort

#     5 3 9  30 34 swap
#     5 9 3  30 34 swap
#     5 9 3  30 34 no swap 330 > 303 -> true
#     5 9 3  34 30 swap

#     9 5 3 34 30 swap
#     9 5 3 34 30 no swap
#     9 5 34 3 30 swap
#         .....

    for(auto num: nums) {
```

Question

https://leetcode.com/problems/h-index/ (https://leetcode.com/problems/h-index/)

In [ ]:

```
1
```