Assignment 7.1: Car Sales (3 points)

A car shop has cars $A = [a_1, a_2, \ldots, a_n]$ (one of each) where a_i is the price of the car i. Customers $B = [b_1, b_2, \ldots, b_m]$ arrive to the shop. b_i is the price that the customer i can afford. What is the maximum amount of sales that can be made?

For example the shop has cars $A = [20, 10, 15, 26]$ and there are customers $B = [11, 25, 15, 9]$ it is possible to make 3 sales.

- The first customer (11) gets car that cost 10, The second customer (25) gets the car that cost 20, and the third customer (15) gets the car that cost 15.

Create a function `sales(A: list, B: list)` in Python which returns the number of possible sales.

Targets:

- The function `sales` generates the correct solution (**2 points**).
- The function also performs at least in $\Theta(n \log n)$ time (**1 point**).

Limits:

- $1 \leq n, m \leq 10^4$
- $1 \leq a_i, b_i \leq 10^4$

A code template with an example program:

```
# sales.py


def sales(cars, customers):
    # TODO


if __name__ == "__main__":
    print(sales([20, 10, 15], [11, 25, 15]))
    print(sales([13, 7, 2, 3, 12, 4, 19], [3, 25, 16, 14]))
    print(sales([24, 6, 20, 21, 12, 5], [25, 1, 24, 15]))
    print(sales([14, 9, 10, 15, 18, 20], [24, 17, 9, 22, 12, 4]))
```

```
$ python sales.py
3
4
3
5
```

Submit your solution in CodeGrade as `sales.py`.

Assignment 7.2: Subsets (3 points)

A given set that has numbers from 1 to N in increasing order $(1, 2, 3, 4, \ldots, N)$, create a function `subsets(N: int)` in Python which produces a list of all possible subsets.

For example when $N = 3$ the subsets are $[1]$, $[2]$, $[1, 2]$, $[3]$, $[1, 3]$, $[2, 3]$ and $[1, 2, 3]$.

Note: The function must return the list of subsets in specific order, The four first generated lists should be:

- `subsets(1) -> [[1]]`
- `subsets(2) -> [[1], [2], [1, 2]]`
- `subsets(3) -> [[1], [2], [1, 2], [3], [1, 3], [2, 3], [1, 2, 3]]`
- `subsets(4) -> [[1], [2], [1, 2], [3], [1, 3], [2, 3], [1, 2, 3], [4], [1, 4], [2, 4], [1, 2, 4], [3, 4], [1, 3, 4], [2, 3, 4], [1, 2, 3, 4]]`

Target: The function `subsets` generates the correct solution (**3 points**).

Limits: $1 \leq N \leq 20$

A code template with an example program:

```
# subsets.py


def subsets(n: int) -> list:
    # TODO


if __name__ == "__main__":
    print(subsets(1))
    print(subsets(2))
    print(subsets(3))
    print(subsets(4))

    S = subsets(10)
    print(S[95])
    print(S[254])
    print(S[826])
```

Output:

```
[[1], [2], [1, 2]]
[[1], [2], [1, 2], [3], [1, 3], [2, 3], [1, 2, 3]]
[[1], [2], [1, 2], [3], [1, 3], [2, 3], [1, 2, 3], [4], [1, 4], [2, 4], [1, 2, 4], [3, 4], [1, 3, 4], [2, 3, 4], [1, 2,
3, 4]]
[6, 7]
[1, 2, 3, 4, 5, 6, 7, 8]
[1, 2, 4, 5, 6, 9, 10]
```

Submit your solution in CodeGrade as `subsets.py`.

Assignment 7.3: The *m* Queens Problem (3 points)

Familiarize yourself with The n Queens Problem from the background material. Putting chess terms aside, the problem can be described as:

How many ways n dots can be placed on a $n \times n$ grid so that no dot shares same row, column or diagonal?

Now instead of n dots, can we solve the problem with any number of dots while the grid remains the same? Create a function `queen(n, m)` in Python, which solves our new problem:

How many ways m dots can be placed on a $n \times n$ grid so that no dot shares same row, column or diagonal?

Target: The function `queen` generates the correct solution (**3 points**).

Limits: $0 \leq n, m \leq 15$

A code template with an example program:

```
# queen.py


def queen(n, m)
    # TODO


if __name__ == "__main__":
    print(queen(4, 4))
    print(queen(4, 2))
    print(queen(6, 4))
    print(queen(7, 2))
    print(queen(8, 8))
```

Output:

```
$ python queen.py
2
44
982
700
92
```

Submit your solution in CodeGrade as `queen.py`.

Last modified: Wednesday, 31 July 2024, 5:30 PM