--------------------

Programming Approach

--------------------

1. Greedy

2. Iterative

------------

Explaination

------------

Data structure

--------------

- license => {license\_no, rate\_of\_growth}

- license\_no : It refers to the number associated to a license

- rate\_of\_growth : the rate by which the price of a license is increasing every month

======================================================================================================================

-------------------------------

Function 1 : find\_max(l\_arr, n)

-------------------------------

\*\*\*\*\*\*\*\*\*\*

Pseudocode

\*\*\*\*\*\*\*\*\*\*

find\_max(l\_arr, n):

1. max = 0

2. for i = 1 to n

3. if l\_arr[i].rate\_of\_growth > l\_arr[max].rate\_of\_growth

4. max = i

5. swap(l\_arr[0].rate\_of\_growth, l\_arr[max].rate\_of\_growth)

6. swap(l\_arr[0].license\_no, l\_arr[max].license\_no)

7. return l\_arr[0].license\_no

Description

-----------

find\_max(l\_arr, n) returns the license\_no of software having maximum rate\_of\_growth value in l\_arr.

Input variables

---------------

- l\_arr : array of objects of license type

- n : number of objects in array l\_arr

Return value

------------

license number with maximum rate\_of\_growth

Side effects

------------

Swaps the maximum object(based on rate\_of\_growth) of array l\_arr with the first object

======================================================================================================================

----------------------------------

Function 2 : print\_order(l\_arr, n)

----------------------------------

\*\*\*\*\*\*\*\*\*\*

Pseudocode

\*\*\*\*\*\*\*\*\*\*

print\_order(l\_arr, n):

1. for i = 1 to n

2. print find\_max(l\_arr[i..n], n-i)

Description

-----------

print\_order(l\_arr, n) function finds and prints the order of buying license to minimize cost.

Input variables

---------------

- l\_arr : array of objects of license type

- n : number of objects in array l\_arr

Return value

------------

None

Side effects

------------

Sorts l\_arr in decreasing order of rate\_of\_growth and prints this sorted array.

======================================================================================================================

Approach

--------

To minimize the cost, license with most expensive rate of growth must be bought first.

The solution is to find maximum, then second maximum, then third maximum and so on.

Since we are interested in immediate best choice, it is a greedy algorithm.

It can be achieved by sorting the array which will take at best O(nlogn) [merge sort, heap sort].

We were given the flexibility to find solution in polynomial of n time.

So, here is a more ituitive solution.

The approach is to find maximum of entire array, print the license number and swap it with the beginning of array.

Now, to find second maximum

find the maximum of l\_arr[2..n], since the first maximum is stored at location l\_arr[1].

Similarly find subsequent maximum and print it.

----------

Complexity

----------

Total no. of software license to buy = n

The find\_max() function works in linear time,

line 1, 5-7 take constant time.

loop line 2-4 of function1 iterates n times.

print\_order() calls the function find\_max() n times, function2 line 1-2.

Therefore, total time complexity is O(n^2).

Time complexity: O(n^2)