# CGJ008

## Big-O Quiz

|  |  |  |  |
| --- | --- | --- | --- |
| **Function** | **f(n)** |  | **O()** |
| A | 2n+1 | 2n | O(n) |
| B | 2n | 2n | O(n) |
| C |  |  | O |
| D |  |  | O |
| E | 1 | 1 | O(1) |
| F |  |  | O(n) |
| G | 2n+2 | 2n | O(n) |
| H |  |  | O |
| I |  | 2n | O(n) |
| J |  |  | O |
| K |  |  | O() |
| L |  |  | O |
| M |  |  | O |
| N |  |  | O |
| O |  |  | O |
| P |  |  | O |
| Q |  |  | O |
| R |  |  | O |
| S |  |  | O |
| T |  |  | O |
| U |  |  | O |
| V |  | 2n | O(n) |
| W |  | 2n | O(n) |
| X |  |  | O |
| Y |  |  | O |
| Z |  |  |  |

Note: In the answer to functionM f(n), the first part is valid for n=0 and up. The alternative is valid for n=1 and up.

## Union Find

## c)

In this implementation (using an integer array, id[]), it is not possible. We need to set the lesser root of p and q as root of the combined set, to keep track of the oldest account. Which excludes the option to always choose the tallest tree as root.

If we were to solve the problem using node objects in a tree structure, one could perform weighted union, swapping the values of the root nodes if the lesser root is the top of the shorter tree.

### d)

### Scheme

While m < n, for i = 1 to m, union (0, i).

### Analysis

|  |  |  |
| --- | --- | --- |
|  |  |  |

Todo:

* Rewrite reverse polish (make simple stack)
* Junit tests for sortablelinkedlist
* Function T, U, Z
* Tests for Big-O
* Union find part d)
* Change big-O(log2(n)) to log(n)