6259. Design Memory Allocator

My Submissions (/contest/weekly-contest-323/problems/design-memory-allocator/submissions/) | Back to Contest (/contest/weekly-contest-323/)

User Accepted:	0
User Tried:	0
Total Accepted:	0
Total Submissions:	0
Difficulty:	Medium

You are given an integer `n` representing the size of a **0-indexed** memory array. All memory units are initially free.

You have a memory allocator with the following functionalities:

1. **Allocate** a block of `size` consecutive free memory units and assign it the id `mID`.
2. **Free** all memory units with the given id `mID`.

Note that:

- Multiple blocks can be allocated to the same `mID`.
- You should free all the memory units with `mID`, even if they were allocated in different blocks.

Implement the `Allocator` class:

- `Allocator(int n)` Initializes an `Allocator` object with a memory array of size `n`.
- `int allocate(int size, int mID)` Find the **leftmost** block of `size` **consecutive** free memory units and allocate it with the id `mID`. Return the block's first index. If such a block does not exist, return `-1`.
- `int free(int mID)` Free all memory units with the id `mID`. Return the number of memory units you have freed.

Example 1:

```
Input
["Allocator", "allocate", "allocate", "allocate", "free", "allocate", "allocate", "allocate", "free", "allocate", "free"]
[[10], [1, 1], [1, 2], [1, 3], [2], [3, 4], [1, 1], [1, 1], [1], [10, 2], [7]]
Output
[null, 0, 1, 2, 1, 3, 1, 6, 3, -1, 0]

Explanation
Allocator loc = new Allocator(10); // Initialize a memory array of size 10. All memory units are initially free.
loc.allocate(1, 1); // The leftmost block's first index is 0. The memory array becomes [1,_,_,_,_,_,_,_,_,_]. We return 0.
loc.allocate(1, 2); // The leftmost block's first index is 1. The memory array becomes [1,2,_,_,_,_,_,_,_,_]. We return 1.
loc.allocate(1, 3); // The leftmost block's first index is 2. The memory array becomes [1,2,3,_,_,_,_,_,_,_]. We return 2.
loc.free(2); // Free all memory units with mID 2. The memory array becomes [1,_, 3,_,_,_,_,_,_,_]. We return 1 since there is
loc.allocate(3, 4); // The leftmost block's first index is 3. The memory array becomes [1,_,3,4,4,4,_,_,_,_]. We return 3.
loc.allocate(1, 1); // The leftmost block's first index is 1. The memory array becomes [1,1,3,4,4,4,_,_,_,_]. We return 1.
loc.allocate(1, 1); // The leftmost block's first index is 6. The memory array becomes [1,1,3,4,4,4,1,_,_,_]. We return 6.
loc.free(1); // Free all memory units with mID 1. The memory array becomes [_,_,3,4,4,4,_,_,_,_]. We return 3 since there are
loc.allocate(10, 2); // We can not find any free block with 10 consecutive free memory units, so we return -1.
loc.free(7); // Free all memory units with mID 7. The memory array remains the same since there is no memory unit with mID 7.
```

Constraints:

- `1 <= n, size, mID <= 1000`
- At most `1000` calls will be made to `allocate` and `free`.

JavaScript

```
function Allocator(n) {
    let a = Array(n).fill(-1), used = new Set();
    return { allocate, free }
    function allocate(size, id) {
        let cnt = 0;
        for (let i = 0; i < n; i++) {
            a[i] == -1 ? cnt++ : cnt = 0; // range count
            if (cnt == size) { // previous range valid
                for (let j = 0; j < cnt; j++) a[i-j] = id;
                return i - size + 1;
            }
        }
        return -1;
    }
    function free(id) {
        let res = 0;
        for (let i = 0; i < n; i++) {
            if (a[i] == id) {
                a[i] = -1;
                res++;
            }
        }
        return res;
    }
}
```

☐ **Custom Testcase** Use Example Testcases

Run Submit

Submission Result: Accepted (/submissions/detail/857934922/) More Details > (/submissions/detail/857934922/)

Share your acceptance!