CHALLENGES

challenge **prisonForPrincesses**

4D 20:24:09

Author anastasiya_8 4000

DESCRIPTION SOLUTIONS 164 COMMENTS 17 README CODEWRITING

Bowser is a villain who often kidnaps princesses.

There are `n` prison cells in the dungeon. Prison cells are numbered from `1` to `n` and arranged in this order along a long corridor at the same distance from each other. Every prison cell has its own capacity $prison_i$.

The entrance to the dungeon is located between the prison cells `entrance` and `entrance + 1`, while if `entrance = 0`, this means that the entrance is in the very beggining before the first prison cell. If `entrance = n`, the entrance is after the last prison cell.

Bowser is going to make raids for kidnaping pricesses. During his j^{th} raid, he plans to kidnap $princesses_j$ princesses. Bringing them into the dungeon, Bowser chooses a prison cell using the following rules:

- If there is a free prison cell with a capacity of $prison_i = princesses_j$, then Bowser chooses this prison cell. Otherwise, he chooses a free prison cell with a capacity of $prison_i > princesses_j$.
- If there are several options, Bowser chooses the one that is located as close as possible to the `entrance`. If there are several prison cells with the same distance to the entrance, he chooses the one closer to the beginning of the corridor. The distance from the `entrance` to the prison cell equals to the number of prison cells between them.
- Bowser does not put `2` groups of princesses to one prison cell, i.e. if the prison cell was filled once it won't be filled anymore.
- If there are no free prison cells with $prison_i \geq princesses_j$, Bowser is upset and let the princesses go.

For each Bowser raid he wants to know the number of the prison cell the princesses will be imprisoned in.

Example

For `prisons = [1, 3, 2, 2]`, `princesses = [1, 1, 3, 2, 1]` and `entrance = 2`, the output should be `prisonForPrincesses(prisons, princesses, entrance) = [1, 2, -1, 3, 4]`.

- `princesses[0] = 1` and `prisons[0] = 1` exists, so Bowser put the first group of princesses to prison number `1`.
- `princesses[1] = 1` and there are no free prisons with the same capacity, so Bowser is looking for prisons with bigger capacity - prisons number `2`, `3` and `4`. The nearest to the entrance are prisons number `2` and `3`, but prison `2` is closer to the beginning of the corridor, so Bowser put the second group of princesses to prison number `2`.
- `princesses[2] = 3` and there are no free prisons with equal or greater capacity, so Bowser let this group of princesses go.
- `princesses[3] = 2` and there are `2` free prisons with the same capacity - prisons `3` and `4`, but prison `3` is closer to the entrance, so Bowser chooses it instead of prison `4`.
- `princesses[4] = 1` and there is one left free prison with greater capacity, so Bowser put princesses there.

Input/Output

- **[execution time limit] 4 seconds (js)**
- **[input] array.integer prisons**

  The capacities of the prison cells.

  Guaranteed constraints:

  $1 \leq prisons.length \leq 10^3$,

  $1 \leq prisons[i] \leq 10^3$.

- **[input] array.integer princesses**

  The number of princesses during the raids.

  Guaranteed constraints:

  $1 \leq princesses.length \leq 10^3$,

  $1 \leq princesses[i] \leq 10^3$.

- **[input] integer entrance**

  The location of the dungeon's entrance.

  Guaranteed constraints:

  $0 \leq entrance \leq prisons.length$.