Gina Cody Scool of Computer Science and Software Engineering
Concordia University
COMP 352: Data Structure and Algorithms Programming Part

Student: Duc Nguyen

PSEUDO-CODE

Global Variables:

- Stack: a Stack with purpose of keeping track all the passed index in the algorithm. (Empty initialy).
- board: the given array of integer.

Algorithm 1 Take input of an integer index, and an array of integer. Return a boolean value indicates if the game is solvable

```
procedure CHECKWINNABLE(INDEX, BOARD)
    if $index + 1 = board.length$ then                              ▷ Terminate case
        return $true$
    if $\neg(Stack.search(index) = -1)$ then
        return $false$                        ▷ Possible redundancy/unsolvable
    if $(index < 0)||(index \geq board.length)$ then         ▷ Out of bound case
        return $false$
    Stack.push(index)              ▷ Valid index, update Stack with new index
    return $checkWinnable(index - board[index], board)$
    $||checkWinnable(index + board[index], board)$
```

A.

1. Time complexity

Since the algorithm uses binary recursion, each half of a call will make **(n-1)** another recursive calls. There are 2 half of each call, thus it would be **2n - 2**. However, since the call also calls itself firstly, it becomes **O(2n-1)**. In addition, the algorithm has to check if the current index was already passed before by *search()* in Stack, results in **n** of time complexity. As a result, the time complexity is of **O(3n-1) = O(n)**.

2. Space complexity

First of all, the space complexity of the algorithm is of $1 + \log_2 n$ since that is the number of maximum active calls at once. However, in each recursive call, the algorithm also has to keep a Stack to store all the passed index; thus, it takes up to **n** space. Therefore, the algorithm space complexity is of $O(1 + \log_2 n + n)$ = **O(n)**.

B.

The type of recursion was being used is **Binary Recursion**. It has no significant change in term of Time complexity, but it has a great impact on reducing space complexity for the algorithm. However, this improvement is covered by the lost of memory for keeping the Stack in this situation.

C.

The test logs for the program was generated randomly in order to guarantee the algorithm's efficiency in every possible scenario. 20 test cases are generated, in addition to a few personal test cases (personal configuration).

D.

To detect an unsolvable array configuration, it depends mostly on the initial pointer (trigger). Every move after the first trigger should not repeat any of the passed index since it will lead to nowhere. In reality, the player actually can move back to his/her old position and then goes on to another position. However, this is redundancy. In order to win the game in a fastest way, the player should **only** move pointer to new position. In the other hand, when designing the algorithm, the computer can detect if an array is winnable or not by checking the next possible moves. If the pointer cannot move to anywhere else other than old position or invalid index (out of bound), the configuration is unsolvable.

$$\underbrace{\underbrace{\boxed{5}}_{\text{index}}\quad \boxed{8}\quad \boxed{2}\quad \boxed{3}\quad \boxed{1}\quad \boxed{5}\quad \boxed{0}}_{\text{Given Configuration}}$$

$$\underbrace{\boxed{0}}_{\text{Stack}}$$

▷ Index was at 0, push(0) to the stack

$$\boxed{5}\quad \boxed{8}\quad \boxed{2}\quad \boxed{3}\quad \boxed{1}\quad \underbrace{\boxed{5}}_{\text{index}}\quad \boxed{0}$$

▷ Move index to index 5

$$\underbrace{\boxed{0}\ \boxed{\longleftarrow 5}}_{\text{Stack}}$$

▷ Index was at 5, push(5) to the stack

At this point, the index (pointer) cannot move 5 steps to the right because it will be out of bound $\longrightarrow$ it moves left

$$\underbrace{\boxed{5}}_{\text{index}}\quad \boxed{8}\quad \boxed{2}\quad \boxed{3}\quad \boxed{1}\quad \boxed{5}\quad \boxed{0}$$

▷ Move index back to 0

$$\underbrace{\boxed{0}\quad \boxed{5}}_{\text{Current stack}}$$

▷ Current index is 0; but there was index 0 inside the Stack

Return False, indicates redundancy/unsolvable configuration