Design Browser History

You have a **browser** of one tab where you start on the `homepage` and you can visit another `url`, get back in the history number of `steps` or move forward in the history number of `steps`.

Implement the `BrowserHistory` class:

- `BrowserHistory(string homepage)` Initializes the object with the `homepage` of the browser.
- `void visit(string url)` Visits `url` from the current page. It clears up all the forward history.
- `string back(int steps)` Move `steps` back in history. If you can only return `x` steps in the history and `steps > x`, you will return only `x` steps. Return the current `url` after moving back in history **at most** `steps`.
- `string forward(int steps)` Move `steps` forward in history. If you can only forward `x` steps in the history and `steps > x`, you will forward only `x` steps. Return the current `url` after forwarding in history **at most** `steps`.

解题思路1 两个Queue倒来倒去

```
class BrowserHistory:

    def __init__(self, homepage: str):

        self.stack = [homepage]
        self.forwardStack = []

    def visit(self, url: str) -> None:
        self.forwardStack = []
        self.stack.append (url)


    def back(self, steps: int) -> str:
        while (len (self.stack) > 1 and steps > 0):
            top = self.stack.pop()
            self.forwardStack.append (top)
            steps -= 1

        return self.stack[-1]
```

```
    def forward(self, steps: int) -> str:
        while (self.forwardStack and steps > 0):
            top = self.forwardStack.pop()
            self.stack.append (top)
            steps -= 1
        return self.stack[-1]
```

解题思路2，用一个ptr，很容易出错，比较考验coding能力，可以在加问使用

```
class BrowserHistory:

    def __init__(self, homepage: str):

        self.stack = [homepage]
        self.pointer = 0

    def visit(self, url: str) -> None:

        self.pointer += 1
        # 1. Clear forward content
        while (self.pointer < len(self.stack)):
            self.stack.pop()

        # now self.pointer is the same size as the stack
        self.stack.append (url)

    def back(self, steps: int) -> str:

        self.pointer = max(0, self.pointer-steps)

        return self.stack[self.pointer]


    def forward(self, steps: int) -> str:

        self.pointer = min (len(self.stack)-1, self.pointer+steps)
        return self.stack[self.pointer]
```

解题思路3 两个ptr, top和current，解决了很多stack pop的问题，缺点是extra memory的allocation

```
class BrowserHistory:
```

```
    def __init__(self, homepage: str):
        self.stack = [""] * 5005
        self.p = self.t = 0
        self.stack[0] = homepage

    def visit(self, url: str) -> None:
        self.p += 1
        self.stack[self.p] = url
        self.t = self.p

    def back(self, steps: int) -> str:
        self.p = max(0, self.p-steps)
        return self.stack[self.p]

    def forward(self, steps: int) -> str:
        self.p = min(self.t, self.p + steps)
        return self.stack[self.p]
```