TP 04

Ana Neri

6 Março 2023

Exercícios práticos sobre complexidade parte 3.

Exercício 1 Uma implementação possível de uma fila de espera (*Queue*) utiliza duas *stacks* A e B, por exemplo:

```
class Queue:
    def __init__(self):
        self.a = []
        self.b = []

    def push(self, x):
        self.a.append(x)

    def pop(self):
        if not self.b:
            while self.a:
                self.b.append(self.a.pop())
        return self.b.pop()

    def empty(self):
        return not self.a and not self.b
```

- A inserção (`enqueue`) de elementos é sempre realizada na *stack* A;
- para a remoção (`dequeue`), se a *stack* B não estiver vazia, é efetuado um *pop* nessa *stack*; caso contrário, para todos os elementos de A exceto o último, faz-se sucessivamente *pop* e *push* na *stack* B. Faz-se depois *pop* do último, que é devolvido como resultado.

1. Efetue a análise do tempo de execução no melhor e no pior caso das funções `enqueue` e `dequeue`, assumindo que todas as operações das *stacks* são realizadas em tempo constante.
2. Mostre que o custo amortizado de cada uma das operações de *enqueue* ou *dequeue* numa sequência de N operações é $\mathcal{O}(1)$ usando o método do potencial.

Exercício 2 Considere um algoritmo de inserção ordenada numa lista (crescente), com uma particularidade: são apagados os nós iniciais da lista contendo valores inferiores ao que está a ser inserido. Por exemplo, a inserção de 30 na lista $[10, 20, 40, 50]$ resulta na lista $[30, 40, 50]$.

```
def insert_rem(p, x):
    new_node = Node(x)
    if p is None:
        return new_node
    if x <= p.value:
        new_node.next = p
        return new_node
    else:
        aux = p
        p = p.next
        del aux
        p = insert_rem(p, x)
        new_node.next = p
        return new_node
```

- Analise o tempo de execução assimptótico de 'insert_rem', identificando o pior e o melhor caso.
- Em termos amortizados a operação de inserção da questão anterior executa em tempo constante. Efetue a sua análise agregada considerando a sequência de inserções $20, 70, 60, 30, 40, 50, 10, 80$ (partindo de uma lista vazia). Considere que o custo real de cada inserção/remoção efetuada à cabeça da lista é 1, por isso a inserção de 30 na lista $[10, 20, 40, 50]$ tem custo 3. Apresente ainda uma função de potencial apropriada sobre as listas, e calcule a partir dela o custo amortizado constante desta operação 'insert_rem'.