Laboratorio #7

Victor Farfan

October 7, 2018

1 Método de Conteo

1.1 Inciso #1

Sabemos que el costo original de hacer POP() o PUSH() en un Stack es O(1). Asemejemos este O(1) con un $1, por lo que cada PUSH() y POP() nos cuesta $1. Lo que haremos será "cobrar" $2 por cada PUSH(). Ahora cada elemento dentro del arreglo tiene $1 guardado como "crédito", la diferencia entre los $2 que cobramos y el $1 que pagamos, que usaremos cada vez que querramos hacer POP(). Ese $1 de credito no solo lo usaremos al llamar a POP(), sino que también lo podemos usar cuando hagamos un backup del Stack. Como copiar también tiene un costo de O(1)=$1, la operacion Backup la pagamos con el credito que ya teniamos. Como cada elemento nunca tendrá menos de $1 de "credito". El tiempo de ejecución es O(n), siendo n el número de veces que llamamos a PUSH(). No tomamos en cuenta las operaciones de POP() o el Backup porque esas operaciones nos salen "gratis" al cobrar extra en el PUSH().

1.2 Inciso #2

```
1  Stack1 = []
2  Stack2 = []
3
4  def Enqueue(element):
5    Stack1.append(element)
6
7  def Dequeue():
8    if len(Stack2) == 0:
9      if len(Stack1) == 0:
10       return 'Cola vacia'
11     while len(Stack1) > 0:
12       p = Stack1.pop()
13       Stack2.append(p)
14   return Stack2.pop()
```

En este algoritmo tenemos 2 Stacks. Al llamar Enqueue() solo hacemos PUSH() a uno de los 2 Stacks. Esto tiene un costo de O(1). Ahora al llamar Dequeue(), todo el contenido del Stack que estamos usando se traslada al otro Stack usando POP(), para que ahora el orden sea invertido y podamos llamar a POP() para tener el primer elemente al que le hicimos Enqueue(). Como estamos usando costos amortizados, cobramos extra por cada Enqueue(), para que Dequeue() nos salga "gratis". Con esto hacer n operaciones nos cuesta O(n), por lo que cada operacion por separado tiene un costo de O(1).

2 Método potencial

2.1 Inciso #1

Hacer PUSH() al Stack tendrá una función potencial de ϕ(Ti) = (s+1) por lo que ϕ(Ti-1) = s. La diferencia seria:
ϕ(Ti) - ϕ(Ti-1) = (s + 1) - s
Por lo que el costo de la operación seria 1 + 1. Porque el costo original de cada PUSH() es = 1. Hacer Backup() solo sería copiar el arreglo entero por lo que igual tendra un costo constante. POP() tendría un costo de 0, lo cual también es constante. Como todas las operaciones tienen un costo constante, hacer n operaciones tendría un costo de O(n).

2.2 Inciso #2

Llamar Enqueue() tiene la funcion potencial:
ϕ(Ti) = (s + 1). Porque estamos haciendo PUSH() a un Stack.
Enqueue() tiene un costo final de 1 + 1, porque al igual que en el inciso anterior, estamos haciendo PUSH() a un Stack.
Para llamar a Dequeue podriamos tener la función potencial: ϕ(Ti) = (n -1). Donde n es el numero de elementos en la cola
Por lo que ϕ(Ti-1) = n.
Por lo que el potencial de cada operacion seria:
ϕ(Ti) - ϕ(Ti-1) = (n - 1) - n
ϕ = - 1
Tanto ϕ = 2 como ϕ = -1 son numeros constantes por lo que son operaciones O(1).