Laboratorio 7

José Alejandro Guzmán Zamora

7 de octubre de 2018

- Método de Conteo
  1. Supongamos que realizamos una serie de operaciones sobre un stack que nunca excede un tamaño K. Cuando realizamos K operaciones, el stack se copia como backup. Demuestre que el costo de las operaciones sería O(n) utilizando costos amortizados.

  Para mantener la implementación del backup correctamente, cada vez que se realizan K operaciones, sin importar de qué tipo sean, es necesario hacer una copia exacta del stack que se está utilizando. Una manera sencilla de realizar esta copia (va a ser bastante útil al momento de hacer el análisis amortizado) es, al final las K operaciones en el stack original, se realizan las mismas operaciones sobre el stack de backup.
  Es importante tomar en cuenta que este problema está condicionado por el valor de K, representante tanto del tamaño máximo del stack como de la cantidad de operaciones que se están realizando. Por ejemplo, si el stack está vacío y k = 5, el análisis debe aplicar para acciones como hacer push 5 veces o hacer push 3 veces y pop 2 veces, entre otras. Para empezar el análisis amortizado, debemos de considerar los costos actuales de las operaciones utilizadas:

Push	1
Pop	1
Multipop	min(s,k)
CopyStack	n - proporcional a la cantidad de elementos en el original

  El costo del multipop está basado en la cantidad de valores k que se desean sacar, limitado por el tamaño s del stack. El costo amortizado de cada operación debe cubir la operación correspondiente en el stack de backup, por lo tanto se pueden utilizar los siguientes costos amortizados:

Push	2
Pop	2
Multipop	2(min(s,k))
CopyStack	0

Tomando como ejemplo un stack vacío y un valor de k = 10, se puede observar lo siguiente (las x's representan espacio libre):

1 x x x x x x x x x	costo = 1, crédito = 1
1 2 x x x x x x x x	costo = 1, crédito = 2
1 2 3 x x x x x x x	costo = 1, crédito = 3
1 2 3 4 x x x x x x	costo = 1, crédito = 4
1 2 3 4 5 x x x x x	costo = 1, crédito = 5
1 2 3 4 5 6 x x x x	costo = 1, crédito = 6
1 2 3 4 5 6 7 x x x	costo = 1, crédito = 7
1 2 3 4 5 6 7 8 x x	costo = 1, crédito = 8
1 2 3 4 5 6 7 8 9 x	costo = 1, crédito = 9
1 2 3 4 5 6 7 8 9 10	costo = 1, crédito = 10

Después de hacer 10 operaciones seguidas de Push, se acumula un crédito total de 10, este se usa posteriormente para hacer Push de los 10 valores correspondientes al stack de back up. Tomando los detalles mencionados en consideración, el costo amortizado total de una serie de operaciones sería de 2*n, lo cual pertenece a $O(n)$.

2. Demuestren cómo implementar un queue con 2 stacks de tal manera que cada operación de ENQUEUE y DEQUEUE tenga un costo de O(1).

Para implementar una cola basada en dos stacks hay que clasificar a un stack como el de ENQUEUE y al otro stack como el de DEQUEUE. La manera en que esto funciona es que cada vez que se requiere encolar un valor, simplemente se le hace Push al stack clasificado como ENQUEUE. En cuanto a la operación de decolar un valor, surgen dos situaciones diferentes. Una situación ocurre cuando el stack clasificado como DEQUEUE no está vacío, en ese caso simplemente se le hace pop y se obtiene el valor necesario. En el caso de que el stack de DEQUEUE esté vacío, se hace pop de todos los valores del stack de ENQUEUE y se trasladan (por medio de otro push) al stack de DEQUEUE de manera inversa. Es importante saber que el último valor al que se le hace pop del primer stack no es necesario trasladarlo al otro stack porque ese es el valor que necesita retornar la operación principal.

Operación: ENQUEUE(1)
ENQUEUE 1 x x x x x
DEQUEUE x x x x x x

Operación: ENQUEUE(2)
ENQUEUE 1 2 x x x x
DEQUEUE x x x x x x

Operación: ENQUEUE(3)

ENQUEUE 1 2 3 x x x
DEQUEUE x x x x x x

Operación: ENQUEUE(4)
ENQUEUE 1 2 3 4 x x
DEQUEUE x x x x x x

Operación: DEQUEUE()
ENQUEUE x x x x x x
DEQUEUE 4 3 2 x x x

Si se utiliza el método de conteo, un costo amortizado adecuado para la operación PUSH (en este caso es el equivalente de la operación central ENQUEUE) es de 3. La otra operación se quedaría con un costo amortizado de 0. La razón de ponerle un costo amortizado de 3 a la operación de ENQUEUE es que como se está utilizando un stack, en el peor de los casos al momento de hacer un DEQUEUE es necesario hacerle un pop al valor del primer stack y un push correspondiente al segundo stack. Estos costos amortizados resultan en un costo total de 3*n, función perteneciente a $O(n)$, como se realizan n operaciones el costo individual es de $O(n)/n = O(1)$.

- Método Potencial
1. Utilicen el método potencial para demostrar que el tiempo de ejecución del problema 1.1 es O(n).
Como primer punto en el análisis potencial de las operaciones sobre el stack con back up, se debe de elegir una función potencial. En este caso se elije la siguiente función:

$$\phi(D_i) = 2s$$

En la función potencial s es el tamaño del stack original. Para demostrar que esta función aplica se utilizará un stack con un valor k = 5.//

D_0 x x x x x Potencial: 0 , Costo: 0
Push(1) 1 x x x x Potencial: 2 , Costo: 3
Push(2) 1 2 x x x Potencial: 4 , Costo: 3
Push(3) 1 2 3 x x Potencial: 6 , Costo: 3
Push(4) 1 2 3 4 x Potencial: 8 , Costo: 3
Push(5) 1 2 3 4 5 Potencial: 10 , Costo: 8 (con backup).

En el caso de una combinación de operaciones:

D_0 x x x x x Potencial: 0 , Costo: 0
Push(1) 1 x x x x Potencial: 2 , Costo: 3
Push(2) 1 2 x x x Potencial: 4 , Costo: 3
Push(3) 1 2 3 x x Potencial: 6 , Costo: 3
Pop() 1 2 x x x Potencial: 4 , Costo: -1
Pop() 1 x x x x Potencial: 2 , Costo: 0 (con backup).

En ambos casos, el costo promedio (4*n para el primero y 2*n para el segundo) pertenecen a $O(n)$.

2. Determine el tiempo de ejecución de hacer K operaciones en el queue del problema 2 utilizando el método potencial.
En este caso se tiene una cola implementada a partir de dos stacks, la función potencial para el análisis se puede representar de la siguiente manera:

$$\phi(D_i) = s_1 + s_2$$

Cada uno de los elementos de la función representa la cantidad total de elementos en cada stack.

Inicio
s_1 x x x x x x Potencial: 0 , Costo: 0
s_2 x x x x x x

Operación: ENQUEUE(1)
s_1 1 x x x x x Potencial: 1 , Costo: 2
s_2 x x x x x x

Operación: ENQUEUE(2)
s_1 1 2 x x x x Potencial: 2 , Costo: 2
s_2 x x x x x x

Operación: ENQUEUE(3)
s_1 1 2 3 x x x Potencial: 3 , Costo: 2
s_2 x x x x x x

Operación: ENQUEUE(4)
s_1 1 2 3 4 x x Potencial: 4 , Costo: 2
s_2 x x x x x x

Operación: DEQUEUE()
s_1 x x x x x x Potencial: 3 , Costo: 8
s_2 4 3 2 x x x

En ese caso específico el costo promedio es aproximadamente de 3 por operación, es decir alrededor de 3*k. El valor del costo en la operación de DEQUEUE puede causar confusión, pues es proporcional a una cantidad mayor que los valores dentro del stack. Sin embargo, para que se pueda realizar operaciones de DEQUEUE es necesario primero realizar ENQUEUE, y estas operaciones siempre van a tener un costo de 2 individualmente. En conclusión el costo de

hacer K operaciones va a ir acompañado de una constante, cualquier función de ese tipo pertenece a $O(K)$.