Métodos Algorítmicos en Resolución de Problemas

Grado en Ingeniería Informática

Hoja de ejercicios 1 — Curso 2018-2019

Ejercicios de análisis amortizado

Ejercicio 1 Si la pila vista en clase incluyese una operación *multiapilar* que apilara k elementos en la pila, ¿se seguiría teniendo un coste amortizado constante para las operaciones?

Ejercicio 2 Demostrar que, si en el ejemplo del contador binario se tuviera una operación *decrementar*, n operaciones podrían tener un coste en $\Theta(nk)$.

Ejercicio 3 Calcular el coste total de una secuencia de n llamadas a las operaciones *apilar, desapilar* y *multidesapilar* si la pila inicial tiene s_0 elementos y la pila final tiene s_n elementos.

Ejercicio 4 Supongamos que el contador binario comienza con b bits a 1 en lugar de con todos a 0. Demostrar que el coste de una secuencia de n llamadas a *contar* sigue estando en $O(n)$, siempre que $n \geq b$.

Ejercicio 5 Se realiza una secuencia de n operaciones sobre una estructura de datos. La operación i-ésima tiene un coste igual a i si i es una potencia de 2 y, en caso contrario, igual a 1. Utilizar el método de agregación para determinar el coste amortizado de las operaciones.

Ejercicio 6 Sea una estructura de datos sobre una lista inicialmente vacía, que dispone de las dos operaciones siguientes: `añadir-número`, que añade un número al comienzo de la lista; `reducir-lista`, que recorre la lista, calcula la suma de todos los números que lee y crea una nueva lista con dicha suma como único elemento. Demostrar que el coste amortizado de las operaciones es constante utilizando el método del potencial.

Ejercicio 7 Supongamos que quisiéramos no solo incrementar el contador binario sino también resetearlo a cero (poner todos sus bits a 0). Mostrar cómo implementarlo como un vector de bits de modo que cualquier secuencia de n operaciones `incrementar` y `resetear` tenga coste en $O(n)$ al ejecutarse sobre un contador inicialmente igual a 0.

Ejercicio 8 Supongamos que queremos realizar `buscar` e `insertar` sobre un conjunto de n elementos. Sea $k = \lceil \lg(n+1) \rceil$ y $\langle n_{k-1}, \ldots, n_0 \rangle$ la representación binaria de n. Tenemos k vectores ordenados, $A_0, A_1, \ldots, A_{k-1}$, donde la longitud de A_i es 2^i. Cada vector está o bien lleno o bien vacío, dependiendo de si n_i es 1 o 0, respectivamente. El número total de elementos almacenados en los k vectores es por tanto $\sum_{i=0}^{k-1} n_i 2^i = n$. Aunque cada vector individual está ordenado, no existe ninguna relación entre los elementos de vectores distintos.

1. Describir cómo implementar `buscar`. Analizar su tiempo de ejecución en el peor caso.
2. Describir cómo insertar un nuevo elemento en esta estructura y calcular su coste amortizado.

Ejercicio 9 Dado un vector $b[0..n-1]$ de enteros mayores que 1, llamado *de bases*, un vector $v[0..n-1]$ con $0 \leq v[i] < b[i]$, para todo i, representará el valor de un contador expresado *en bases* b, siendo $v[0]$ la cifra menos significativa. El valor del contador viene dado por el sumatorio $\sum_{i \in 0..n-1}(v[i] * \prod_{j \in 0..i-1} b[j])$. Implementa la operación *incr*, que incrementa el contador en una unidad, admitiéndose que cuando el contador tenga su valor máximo, $\forall i \in 0..n-1 \,.\, v[i] = b[i] - 1$, al incrementarlo pase a valer 0, o sea $\forall i \in 0..n-1 \,.\, v[i] = 0$. Demuestra con cualquiera de los métodos vistos en clase, que el coste amortizado de la operación *incr* está en $O(1)$.

Ejercicio 10 Queremos implementar una estructura de cola cuyos elementos son números, con las dos operaciones típicas de *añadir* en un extremo y *eliminar* en el otro; además, nos interesa una tercera operación, *máx*, que devuelve el valor máximo en la cola.

Demostrar que no se puede implementar esta estructura de forma que las tres operaciones tengan coste constante en el caso peor.

Diseñar una implementación en la que las tres operaciones tengan coste amortizado constante.