Posgrado de Ingeniería de Sistemas

Análisis amortizado del algoritmo unir–encontrar

Alberto Benavides
30 de abril de 2020

1. Análisis amortizado

En el estudio de la complejidad computacional de los algoritmos, el *análisis amortizado* pretende definir el costo promedio de los peores casos a ejecutar de una serie de operaciones. Esto se logra sumando los costos de operación en su peor caso, divididos entre n operaciones que se realicen. Los análisis de costo amortizado se utilizan para demostrar que en promedio, una serie de operaciones representa un costo de operación bajo pese a que tenga que ejecutar operaciones de costo elevado.

Existen tres maneras de realizar un análisis amortizado: El *análisis general*, el *método contable* y el *método potencial*. El primero implica encontrar el peor caso para todas las operaciones y luego dividir la suma de los peores casos entre el número de operaciones. El segundo toma en cuenta el impacto que tienen las operaciones realizadas con las que les siguen. Se llama método contable porque simula el cálculo del impacto de una deuda de las operaciones de mayor costo y lo solventa mediante operaciones de bajo costo infladas, de modo que el excedente que se paga en las operaciones de bajo costo amortiza el de las operaciones más costosas. El último toma en cuenta el costo del conjunto de operaciones al asignar diferencias de costos entre los estados del proceso, de manera que la diferencia entre los costos de los estados inicial y final da por resultado el costo amortizado del conjunto de operaciones.

2. Unión–encontrar

El algoritmo *unión–encontrar* sirve para almacenar estructuras de datos dinámicos al tratarlos como conjuntos por asignarles identificadores únicos a sus elementos. Las operaciones que este algoritmo realiza son las de

- `HacerConjunto`: Crea un conjunto C de un solo elemento x y establece a dicho elemento como su **representante**. Esto se logra en tiempo $\mathcal{O}(1)$ en el peor de los casos.

- `Unir`: A partir de dos elementos x y y, une los dos subconjuntos que los contienen y establece a uno de los representantes como el representante del nuevo conjunto: $x \in C_1, y \in C_2, C_1 U C_2$. El tiempo de esta operación es consiste primero en encontrar x y y en los subconjuntos. Si suponemos que la cardinalidad entre conjuntos podría ser $n/2$ en el peor de los casos, entonces esta búsqueda tiene complejidad $\mathcal{O}(n/2 + n/2) = \mathcal{O}(n)$.

- `Buscar`: Devuelve el representante de un conjunto en tiempo ∞.

Ahora se procederá por análisis general para elaborar el análisis amortizado del algoritmo unir–encontrar. Primero se define una cantidad m de ejecuciones de esta secuencia para n elementos tal que $m = n - 1$. Así

1. primero se hacen n operaciones de `HacerConjunto` con tiempo $\mathcal{O}(n)$; y

2. por último se realiza $n - 1$ veces la operación `Unir` para cada par de conjuntos, lo cual implica $\mathcal{O}(n(n-1)) = \mathcal{O}(n^2)$.

Como este proceso se realiza $m = n - 1$ veces, entonces se puede dividir $\mathcal{O}(n^2)/(n-1)$ para concluir que un análisis amortizado del algoritmo unión–encontrar implica un tiempo de cómputo de aproximadamente $\mathcal{O}(n)$.

Referencias

[1] Cormen, Thomas and Leiserson, Charles and Rivest, Ronald and Stein, Clifford. Lecture 11. Amortized analysis. `https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-046j-design-and-analysis-of-algorithms-spring-2012/lecture-notes/MIT6_046JS12_lec11.pdf`, 2012. [Accedido 25/abril/2020].

[2] Morgan Wilde. Amortized analysis and Union–Find. `https://www.cs.upc.edu/~mjserna/docencia/grauA/T19/Union-Find.pdf`, 2020. [Accedido 28/abril/2020].

[3] Elisa Schaeffer. Complejidad computacional de problemas y el análisis y diseño de algoritmos. [No publicado], 2019.