Estructura de datos
Montículos binomiales y binarios
Actividad 9

Dagoberto Quevedo

25 de marzo de 2020

Resumen

En esta actividad se describen las propiedades y funcionalidades de los montículos binomiales y binarios, realizando finalmente un diseño experimental que compara el rendimiento de estas estructuras en una operación de inserción y búsqueda del elemento mínimo.

1. Montículos binarios

Se define como un árbol binario con las siguientes consideraciones:

- Todos los niveles del árbol están completos (quizá excepto el último) y si el último nivel del árbol no está completo, los vértice de ese nivel se asignan de izquierda a derecha.
- La clave almacenada en cada vértice es mayor o igual que ($\geq$) o menor o igual que ($\leq$) las claves en los elementos secundarios del vértice.
- Los monticulos binarios donde la clave principal es mayor o igual que ($\geq$) las claves secundarias se llaman **montículo máximo**; aquellos donde es menor o igual a ($\leq$) se llaman **montículo mínimo**

2. Montículos binómico

Un montículo binómico se define de una manera recursiva. Un árbol binómico de un vértice es B_0 y el único vértice es la raíz. El árbol B_{k+1} contiene dos copias de B_k tal que la raíz de una copia es la raíz de $B : k+1$ y la raíz de la otra copia es un hijo directo de esta raíz Algunas consecuencias

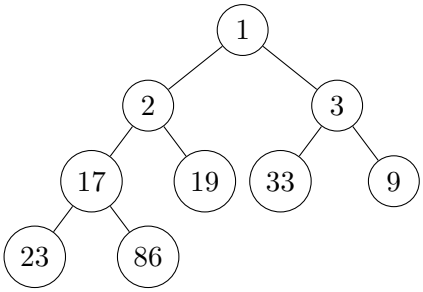

Figura 1: Ejemplo de un montículo mínimo binario

de esta definición son que: a) El árbol B_k contiene 2_k vértices, b) La altura de B_k es k. La raíz de B_k tiene k hijos directos, $B_{k-1}, B_{k-2}, \ldots, B_1, B_0$.

Las claves son almacenadas en un árbol binómico según el orden de montículo: la clave del vértice padre es menor que la clave de su hijo. Cada vértice contiene una clave. Un montículo binómico es un conjunto de árboles binómicos tal que no hay duplicados de los B_k. [3]

3. Operaciones con montículos

La siguiente tabla resume el orden de operaciones requeridos para las operación usuales en montículos.

Operación	Binario	Binómico
Inserción	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$
Minímo	$\mathcal{O}(1)$	$\mathcal{O}(\log n)$
Unión	$\mathcal{O}(n)$	$\mathcal{O}(\log n)$
Eliminación	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$

4. Implementación computacional

Se realiza la implementación computacional en Python de las estructuras de un montículo binómico y binario. Para el montículo binario se hace uso de la librería `binary_heap`, para la implementación del montículo binómico se hace uso del código disponible en [1].

4.1. Condiciones de experimentación

Se mide el rendimiento en las operaciones de inserción y búsqueda del elemento mínimo en cada estructura. La evaluación se realiza con una serie de instancias, en este caso cada instancia se define como un vector con elementos enteros de tamaño 2^n, donde $n \in \{1, \ldots, 20\}$, la ejecución se repite $k = 10$ veces, con este vector se realiza la inserción iterativa y al finalizar la búsqueda del elemento mínimo.

4.2. Resultados

El gráfico 2 muestra los resultados del diseño experimental en el tiempo de computo expresado en escala logarítmica, el eje vertical expresa el valor de n que define el tamaño de la instancia 2^n.

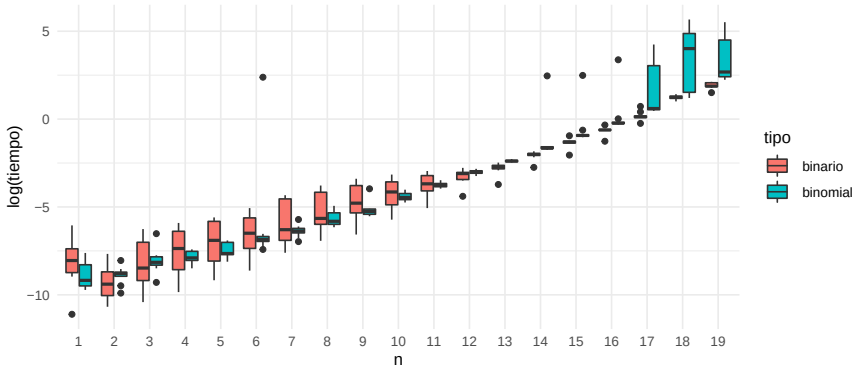

Figura 2: Resultados del diseño experimental con $k = 10$ ejecuciones para cada tipo de estructura

4.3. Conclusiones

El montículo binario muestra un rendimiento superior para las operaciones asignadas, siendo consistente con tiempo en orden de operaciones requeridos para cada operación descrito en la tabla **??**. Cabe señalar que el gráfico no muestra aun que el experimento alcanzo zona asintótica por lo que es requerido ampliar el tamaño del experimento de tal manera de tener resultados concluyentes.

Referencias

[1] *Binomial heap*, Project Nayuki, `https://www.nayuki.io/page/binomial-heap`

[2] Donald Knuth, *Sorting and searching*, The Art of Computer Programming, Addison-Wesley Professional, 1998.

[3] Elisa Schaeffer, *Modelos computacionales*, Complejidad computacional de problemas y el análisis y diseño de algoritmos, notas de curso, 2020.