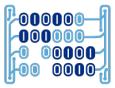

MACC
Matemáticas Aplicadas y
Ciencias de la Computación

UNIVERSIDAD DEL ROSARIO

MATEMÁTICAS APLICADAS Y CIENCIAS DE LA COMPUTACIÓN

van Emde Boas Trees

Entrega final

Autores:
Samuel Pérez
Nicolás Duque

Supervisor:
Prof. Julián Rincón

Entrega final, proyecto para el curso ”Algoritmos y Estucturas de Datos”.
Noviembre 2018

0.1 van Emde Boas Trees

Entrega final
Repositorio: `https://github.com/BogoCoder/vanemdeboas`

Resumen: En anteriores estructuras de datos tales como `priority_queues`, `red_black trees`, o `Fibonacci heaps`, nos dimos cuenta que al menos una de sus operaciones importantes requería un tiempo de $O(\log n)$.

Este proyecto consistió en la comprensión e implementación de los van Emde Boas Trees, una estructura de datos avanzada la cual nos permitirá tener un desempeño óptimo en las operaciones típicas de árboles, logrando un tiempo de $O(\log \log u)$, en la cual u representará el tamaño de nuestro $universo\ de\ datos$. Nuestra estructura se restringirá para enteros desde 1 hasta $u-1$, en los cuales no se permite duplicados. Para esto, se trabajó fuertemente en los conceptos preliminares e incluso, se desarrolló un prototipo de la estructura, para que fuera base en la implementación objetivo.

Funcionalidad: A continuación, presentamos los diversos métodos desarrollados en la clase `vEB` (van Emde Boas Tree), un breve resumen de sus funcionalidades, y sus respectivos análisis en el $worst - case$.

- `INSERT(x)`: Inserta el elemento x a la estructura. $O(\log \log u)$
- `REMOVE(x)`: Remueve el elemento x de la estructura. $O(\log \log u)$
- `SUCC(x)`: Encuentra y retorna el sucesor del elemento x. $O(\log \log u)$
- `PRED(x)`: Encuentra y retorna el predecesor del elemento x. $O(\log \log u)$
- `MIN()`: Encuentra y retorna el mínimo de la estructura. $O(1)$
- `MAX()`: Encuentra y retorna el máximo de la estructura. $O(1)$
- `EMPTY()`: Retorna `TRUE` si la estructura está vacía (tiene 0 elementos) o `FALSE` en el caso contrario. $O(1)$
- `CLEAR()`: Limpia la estructura, es decir, elimina todos los elementos contenidos en esta. $O(\log u)$
- `SIZE()`: Retorna la cantidad de elementos de la estructura. $O(1)$

Descripción: Para la construcción de una estructura `vEB`, la cual, como se inferiría, es una estructura propiamente recursiva, necesitamos de los siguientes atributos condicionados:

- Asumimos $u = 2^k$ para un entero $k \geq 1$.

- Definimos `minv` como el mínimo elemento de la estructura.
- Definimos `maxv` como el máximo elemento de la estructura.

A menos de que `u == 2`, definimos lo siguiente:

- `summary` apunta a una estructura `vEB(`$2^{\lfloor \log u/2 \rfloor}$`)`
- Arreglo `cluster` apunta a `(`$2^{\lceil \log u/2 \rceil}$`)vEB(`$2^{\lfloor \log u/2 \rfloor}$`)` trees.

Dada esta estructura con los atributos mencionados, se nos permite agilizar los procesos típicos de una manera sumamente eficiente, proporcionándonos las herramientas para implementar los métodos mencionados arriba en tiempos tales como $O(\log \log u)$.

Conclusiones:

- Se logró, claramente, nuestro objetivo inicial, que era llegar a un tiempo de $O(\log \log u)$ en las funciones usuales.
- Se logró la implementación de la estructura `vEB`, en la cual agregamos un atributo extra `count` que almacena la cantidad de elementos en la estructura, y nos permitió el desarrollo de métodos como `SIZE` o `CLEAR`.
- Se logró la implementación de la estructura `proto_vEB`, la cual, logramos mejorar, al agregar un atributo `count` que almacena la cantidad de elementos en la estructura, esto nos permitió agilizar métodos tales como `INSERT`, y desarrollar métodos como `CLEAR`.
- Un objetivo a futuro es modificar la estructura para que soporte $keys$ duplicadas, o incluso, tipos de datos que sean escalares.
- Tomar el proceso de aprendizaje tomándose el detenimiento de entender primero los conceptos base resulta mucho más provechoso a la hora de abrirse a una nueva estructura tal como es van Emde Boas Trees.

Herramientas:

- Cormen, Thomas H., Leiserson, Charles E., Rivest, Ronald L. and Stein, Clifford. (2009). Introduction to Algorithms. Tercera edición. MIT Press.
- C++.