Tarea 2

Splay Trees y ABB's

Profesores: Benjamín Bustos y Gonzalo Navarro
Auxiliares: Máximo Flores y Sergio Rojas

Fecha de entrega: jueves 7 de noviembre a las 23:59
Atrasos hasta miércoles 13 de noviembre a las 23:59

1. Contexto

Los Splay Trees son árboles binarios de búsqueda que alcanzan un costo amortizado $O(log(n))$ por operación sin necesidad de almacenar información de balanceo. Más aún, una secuencia de accesos a los nodos con distintas probabilidades entrega un costo amortizado de $O(H)$, con H la entropía de esas probabilidades. En esta tarea compararemos los splay trees con árboles de búsqueda clásicos para distintos escenarios de inserción y búsqueda.

2. ABB clásico

Un árbol binario de búsqueda clásico es un árbol binario que cumple con las siguientes condiciones:

- (Su subárbol izquierdo es vacío) $\vee$ (la raíz es mayor a todos los elementos del subárbol izquierdo $\wedge$ este último es un ABB).
- (Su subárbol derecho es vacío) $\vee$ (la raíz es menor a todos los elementos del subárbol derecho $\wedge$ este último es un ABB).

2.1. Búsqueda en ABB clásico

Sea x el elemento buscado, se inicia la búsqueda en la raíz. Si el elemento en la raíz es x, se termina la búsqueda con éxito. En caso contrario, se compara x con el elemento de la raíz r:

- Si $x < r$, se realiza el mismo proceso en el subárbol izquierdo.
- Si $x > r$, se realiza el mismo proceso en el subárbol derecho.

Si eventualmente el subárbol donde deberíamos seguir la búsqueda es vacío, significa que el elemento no existe, terminando el proceso de manera infructuosa.

2.2. Inserción en ABB clásico

Sea x el elemento a insertar, se realiza un proceso homólogo a la búsqueda. Si la raíz es vacía, se crea un árbol de un solo nodo que contiene el elemento a insertar. En caso contrario, se compara x con el elemento de la raíz r:

- Si $x < r$, se realiza el mismo proceso en el subárbol izquierdo.
- Si $x > r$, se realiza el mismo proceso en el subárbol derecho.

3. Splay Tree

Un Splay Tree es un ABB que incluso las operaciones de lectura modifican el árbol. Sus respuestas no cambian, pero se hacen más eficientes gracias a las modificaciones.

3.1. Operaciones

En un splay tree, el nodo al que se acaba de acceder siempre debe quedar en la raíz del árbol. Para esto, si se accedió al nodo x, éste se lleva a la raíz mediante una operación llamada *splay*(x). Esta operación se basa en una secuencia de rotaciones. Si $z(A, B)$ denota el árbol con z el elemento de la raíz, A subárbol izquierdo y B subárbol derecho, para subir el nodo x existen las siguientes rotaciones:

- Zig-zig: $z(y(x(A, B), C), D) \to x(A, y(B, z(C, D)))$
- Zig-zag: $z(y(A, x(B, C)), D) \to x(y(A, B), z(C, D))$
- Zag-zig: $z(A, y(x(B, C), D)) \to x(z(A, B), y(C, D))$
- Zag-zag: $z(A, y(B, z(C, D))) \to x(y(z(A, B), C), D)$
- Zig: $y(x(A, B), C) \to x(A, y(B, C))$ (sólo si y es raíz)
- Zag: $y(A, x(B, C)) \to x(y(A, B), C)$ (sólo si y es raíz)

Notar que las últimas dos rotaciones son simples y solamente se realizan cuando y es raíz, es decir, *splay*(x) consistirá de 0 o más rotaciones dobles consecutivas, y puede (o no) finalizar con una única rotación simple.

3.2. Búsqueda en Splay Tree

Se busca x como en un árbol binario de búsqueda clásico y luego se hace *splay*(x).

3.3. Inserción en Splay Tree

Se inserta x como en un árbol binario de búsqueda clásico y luego se hace *splay*(x).

4. Descripción de la tarea

4.1. Objetivos

Para esta tarea el trabajo será implementar estas dos estructuras de datos, y evaluarlas y analizarlas experimentalmente bajo distintos escenarios. Es decir, los objetivos son:

1. Implementar el ABB clásico junto a sus métodos de búsqueda e inserción.
2. Implementar las 6 rotaciones expuestas y el *splay*(x).
3. Implementar el Splay Tree junto a sus métodos de búsqueda e inserción.
4. Comparar estas dos estructuras en distintos escenarios.

5. Experimentación

Se deben comparar ambas estructuras en los siguientes escenarios:

1. Se insertan N enteros distintos de manera aleatoria, luego se hacen M búsquedas por valores escogidos entre los elementos insertados. Es decir, si se insertaron los elementos $A[0, \ldots, N-1]$, el elemento $A[i]$ se buscará $\frac{M}{N}$ veces. Para esto se puede crear un arreglo $B[0, \ldots, M-1]$ con los elementos a buscar, poniendo $\frac{M}{N}$ copias de $A[i]$ en B. Luego se genera una permutación aleatoria de B, el cual sentenciará el orden de búsqueda.

2. Se realiza la inserción de la misma manera que en 1., pero al hacer las búsquedas, se le asignarán probabilidades sesgadas de búsqueda de elementos. Es decir, si se insertaron los elementos $A[0, \ldots, N-1]$, se deberá definir una función de probabilidad $f : [0, \ldots, N-1] \to [0, 1]$. De esta forma el elemento $A[i]$ se tendrá que buscar $\lfloor M \cdot f(i) \rfloor$ veces. Se crea un arreglo $B[0, \ldots, M'-1]$ con los elementos a buscar, poniendo $\lfloor M \cdot f(i) \rfloor$ copias de $A[i]$ en B. Luego se genera una permutación aleatoria de B, el cual sentenciará el orden de búsqueda. Notar que $M' = \#\text{copias} = \sum_{i=0}^{N-1} \lfloor M \cdot f(i) \rfloor \leq M$, no se preocupe de eso y trabaje simplemente con $|B| = M'$.

3. Antes de insertar los elementos se ordena A. La búsqueda se hace como en 1.

4. Antes de insertar se copia los elementos de A a otro arreglo C, el cual será el arreglo que se ordene para realizar posteriormente las inserciones (es decir, se usa C). La búsqueda se hace como en 2., notar que A debe permanecer desordenado para que f (la misma función definida en 2.) altere aleatoriamente las probabilidades de buscar.

Utilice $N \in 10^6 \cdot \{0.1, 0.2, \ldots, 1.0\}$, y $M = 100 \cdot N$ y mida el tiempo de realizar la totalidad de las búsquedas, con esto se podrá obtener el costo promedio de búsqueda.

Respecto a la función f a utilizar en 2. y 4., esta deberá ser de la forma $f(i) = \frac{C}{(i+1)^2}$, donde deberán determinar la constante C para que $\sum_{i=0}^{N-1} f(i) = \sum_{i=0}^{N-1} \frac{C}{(i+1)^2} = 1$.

6. Entregables

Se deberá entregar el código y un informe donde se explique el experimento en estudio. Con esto se obtendrá una nota de código ($NCod$) y nota de informe ($NInf$). La nota de la tarea será:

$$NT_2 = 0.5 \cdot NCod + 0.5 \cdot NInf$$

6.1. Código

La entrega de código debe ser hecha en C, C++ o Java. Tiene que contener:

- (**0.5 pts**) README: Archivo con las instrucciones para ejecutar el código, debe ser lo suficientemente explicativo para que cualquier persona solo leyendo el README pueda ejecutar la totalidad de su código (incluyendo las librerías no entregadas por nosotros que potencialmente se deban instalar).
- (**0.5 pts**) Implementación del ABB clásico.
- (**1.5 pts**) Implementación del Splay Tree.

- (**1.0 pts**) Creación de los 4 experimentos señalados.
- (**2.0 pts**) Obtención de resultados: La forma en el que se obtienen los resultados es correcta y es suficiente para poder obtener conclusiones.
- (**0.5 pts**) Main: Un archivo o parte del código (función main) que permita ejecutar los experimentos.

6.2. Informe

El informe debe ser claro y conciso. Se recomienda hacerlo en LaTeX o Typst. Debe contener:

- (**0.8 pts**) Introducción: Presentación del tema en estudio, resumir lo que se dirá en el informe y presentar una hipótesis.
- (**0.8 pts**) Desarrollo: Presentación de algoritmos, estructuras de datos y cómo funcionan y por qué. Recordar que los métodos ya son conocidos por el equipo docente, lo que importa son sus propias implementaciones.
- (**2.4 pts**) Resultados: Especificación de los datos que se utilizaron para los experimentos, la cantidad de veces que se realizaron los tests, con qué inputs, que tamaño, etc. Se debe mencionar en que sistema operativo y los tamaños de sus cachés y RAM con los que se ejecutaron los experimentos. Se deben mostrar gráficos/tablas y mencionar solo lo que se puede observar de estos, se deben mostrar los valores y parámetros que se están usando.
- (**1.2 pts**) Análisis: Comentar y concluir sus resultados. Se hacen las inferencias de sus resultados.
- (**0.8 pts**) Conclusión: Recapitulación de lo que se hizo, se concluye lo que se puede decir con respecto a sus resultados. También ven si su hipótesis se cumplió o no y analizan la razón. Por último, se menciona qué se podría mejorar en su desarrollo en una versión futura, qué falta en su documento, qué no se ha resuelto y cómo se podrían extender.

Todo lo mencionado debe estar en sus informes en las secciones en las que se señalan, la falta de algún aspecto o la presencia de algún aspecto en una sección equivocada hará que no se tenga la totalidad del puntaje.

Recordar que por cada día o fin de semana de atraso se descuenta un punto a NT.