Estructura de Datos y Algoritmos

ITBA 2024-Q2

Indices y Arreglos ordenados

Ventajas

- El arreglo ordenado es una Buena estrategia para implementar un índice (búsqueda rápida de la info). Permite algoritmo de "búsqueda binaria".

- Excelente para algunas operaciones que requieren acceso a una componente en particular, ya que se accede por "posición". Ej: getMaximo(), getMínimo()

Indices y Arreglos ordenados

Deventajas

- Los datos tienen que estar contiguos. Para garantizarlo, el insertado/borrado de un elemento require "mover" otras componentes.

- Además, cuando se acaba el espacio pre-alocado, generar más espacio (aún de a "chunks") implica generar **otro espacio contiguo** y llevar todas las componentes. Por eso conviene hacerlo de a "chunks".

Analizando otras alternativas

Tratando de superar el problema de la contiguedad y re-alocación del espacio, podemos pensar en estructuras de datos que permitan que los elementos estén “físicamente aislados y lógicamente conectados”.

Opción: “Lista lineal simplemente encadenada”

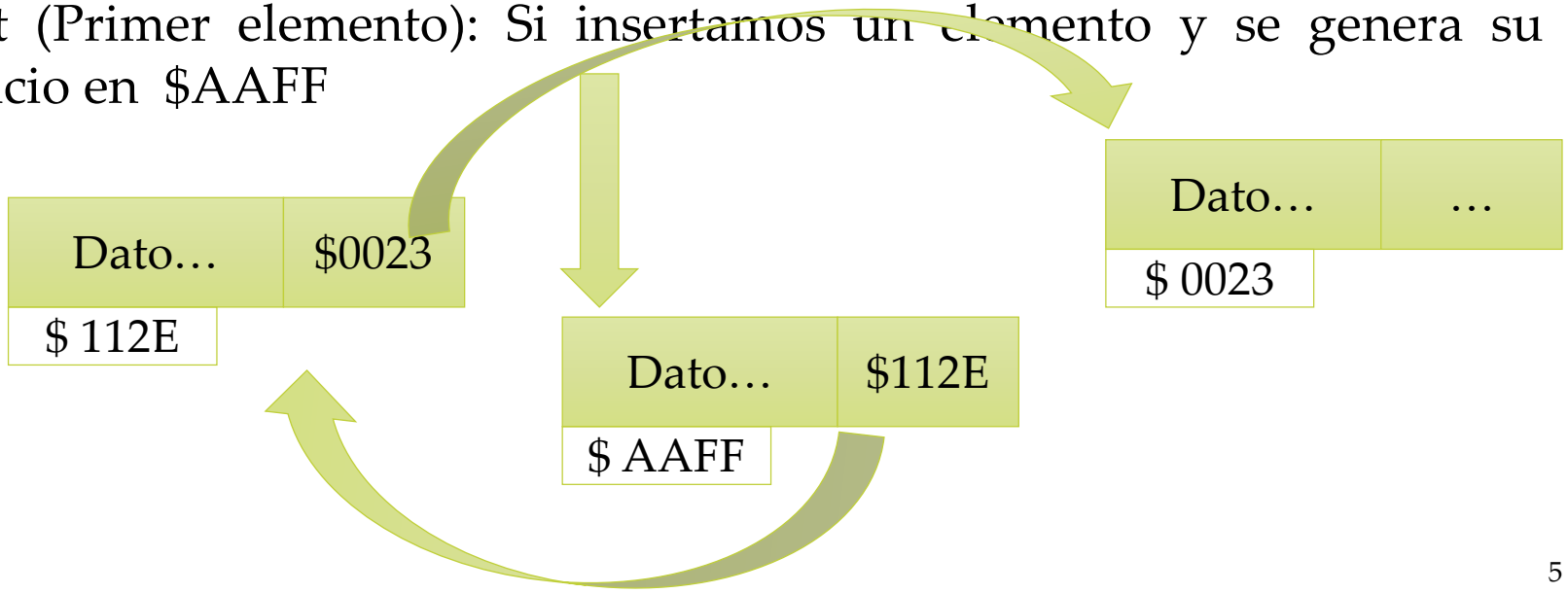
Root (Primer elemento): null

Analizando otras alternativas

Tratando de superar el problema de la contiguedad y re-alocación del espacio, podemos pensar en estructuras de datos que permitan que los elementos estén "físicamente aislados y lógicamente conectados".

Opción: "Lista lineal simplemente encadenada"

Root (Primer elemento): Si insertamos un elemento y se genera su espacio en $AAFF



Definición Lista Lineal Simplemente Encadenada

Es una estructura de datos que está compuesta por 0 o más nodos. Cada nodo (elemento) almacena 2 cosas: su info y la referencia al elemento siguiente.

Variante para Indice

Definición Lista Lineal Simplemente Encadenada **Ordenada**

Es una lista lineal simplemente encadenada que además mantiene los **elementos ordenados** con algún criterio de ordenación.

TP 3C- Ejer 2.1, 2.2

Bajar de Campus **SortedListService** y **SortedLinkedList.java**

La clase permite crear un índice ordenado que no acepta repetidos (los ignora).

Modicando la implementación (no la interface) podríamos usarla para implementar lista compactada (guarda la cantidad de apariciones)

Discutamos el insert

La inserción puede implementarse de diferente maneras:

- **Resuelto totalmente en SortedLinkedList (la provista), en forma iterativa**
- **Resuelto totalmente en SortedLinkedList, en forma recursiva (ejer 2.2)**
- **Delegando a la clase Node la inserción. (ejer 2.3)**

Implementar ahora la versión 2.2

TP 3C- Ejer 2.4

Analogamente, remove() puede implementarse en 3 versiones.

Ahora haganlo en la version iterativa.

Luego en las otras 2 versiones.

Posible solución

```
@Override
public boolean remove(T data) {
  Node prev= null;
  Node current = root;

  while (current!=null && current.data.compareTo(data) < 0) {
   // avanzo
   prev= current;
   current= current.next;
   }

  // lo encontre?
  if (current!=null && current.data.compareTo(data) == 0) {
     // borrando
     if (current == root)
         root= root.next;
     else
        prev.next=current.next;

    return true;
     }

return false;
}
```

TP 3C- Ejer 2.5

Implementar los métodos faltantes (5 min)