Universidad Nacional Autónoma de Honduras

Departamento de Matematicas

Estructura de Datos

Implementación de Vector y Lista

Docente:
Msc. Pablo Esau Mejia

Estudiante:
Rosy Nicole Moreno Natarén
20202001698

Febrero 26, 2024

1 Vector

Se le presentan los algoritmos de las operaciones pushFront y pushBack para la estructura de datos vector, los algoritmos de popBack y popFront son similares a los otros casos.

Importante:
numE: numero de elementos en el vector.
Vec= vector de elementos.
Dim= Dimensión del vector.

Algorithm 1 pushFront (key)

```
1: if numE = Dim then
2:     return Esta lleno
3: end if
4: else
5: Vec1 ← key
6: for i = 1 hasta numE
7: Vec1 ← Vec[i-1]
8: numE ← numE+1
9: Vec ← Vec1
10: end for
```

Tiempo de ejecución: O(n)

Algorithm 2 pushBack (key)

```
1: if numE = Dim then
2:     return Esta lleno
3: end if
4: else
5: Vec[numE] ← key  O(1)
6: numE ← numE+1
```

Tiempo de ejecución: O(1)

2 Lista

En programación, las listas son como las listas de compras de la vida real. Puedes añadir elementos (como si añadieras más cosas a tu lista de com-

pras), quitar elementos (como si tacharas elementos de tu lista después de comprarlos), e incluso insertar elementos en una posición específica (como si decidieras que un elemento es más importante y debe estar en la parte superior de tu lista).

Las listas en programación son dinámicas, lo que significa que puedes agregar o eliminar elementos a voluntad, y la lista se ajustará para acomodar estos cambios.

Alternativamente, una lista es una colección de objetos almacenados en memoria (no necesariamente consecutiva) que se encuentran conectados o en enlazados de manera secuencial. Otro forma de definir una lista es mediante el termino "Nodo", un nodo es un objeto que almacena un dato (key) y un apuntador a Nodo

2.1 Lista enlazada sin cola

Algorithm 3 PushFront (key)

```
1: new node ← node.key
2: new node.next ← head
3: head ← new node
```

Tiempo de ejecución: O(1)

Algorithm 4 PopFront ()

```
1: if head = Null then
2:     ERROR: empty list
3: end if
4:
5: dato ← head.key
6: head ← head.next
7: return dato
```

Tiempo de ejecución: O(1)

Algorithm 5 PushBack (key)

```
1: new node ← node.key
2: iterador ← head
3:
4: if head ≠ Null then
5:     while iterador.next ≠ Null do
6:         iterador ← iterador.next
7:     end while
8: end if
9:
10: iterador.next ← new node
```

Tiempo de ejecución: O(n)

Algorithm 6 PopBack ()

```
1: iterador ← head
2: node aux
3:
4: if head = Null then
5:     ERROR: empty list
6: end if
7:
8: while iterador.next ≠ Null do
9:     aux ← iterador
10:    iterador ← iterador.next
11:    dato ← iterador.key
12: end while
13:
14: aux.next← Null
15: return dato
```

Tiempo de ejecución: O(n)

2.2 Lista enlazada con Cola

Algorithm 7 PushFront (key)

```
1: new node ← node
2: node.key ← key
3: node.next ← head
4:
5: if tail = Null then
6:     tail ← head
7: end if
```

Tiempo de ejecución: O(1)

Algorithm 8 PopFront ()

```
1: if head = Null then
2:     ERROR: empty list
3: end if
4:
5: head ← head.next
6: if head = Null then
7:     tail ← Null
8: end if
```

Tiempo de ejecución: O(1)

Algorithm 9 PushBack (key)

```
1: new node ← node
2: node.key ← key
3: node.next ← Null
4:
5: if head ≠ Null then
6:     head ← tail ← node
7: end if
```

Tiempo de ejecución: O(1)

Algorithm 10 PopBack ()

```
1: dato ← head.key
2:
3: if head = Null then
4:     ERROR: empty list
5: end if
6:
7: Nodo iter ← head
8: while iterador.next ≠ Null do
9:     iterador ← iterador.next
10: end while
11:
12: iterador.next ← NULL
13: tail ← iterador
```

Tiempo de ejecución: O(n)

2.3 Lista doblemente enlazada

Una lista doblemente enlazada es una estructura de datos en la que cada elemento de la lista está enlazado tanto a su predecesor como a su sucesor. Cada nodo en la lista contiene dos enlaces, uno apuntando al nodo anterior y otro apuntando al nodo siguiente. Esta estructura permite la navegación en ambas direcciones: hacia adelante y hacia atrás.

Una lista doblemente enlazada se define como una colección de elementos (nodos) donde cada nodo contiene dos campos: uno para almacenar el valor del elemento y otros dos campos para enlazar con el nodo anterior y el nodo siguiente. Además, la lista debe tener referencias a su primer y último nodo para permitir un acceso eficiente a ambos extremos de la lista.

Algorithm 11 PushBack (key)

```
1: node ← new node
2:
3: if head = Null then
4:     head ← node
5:     tail ← node
6:
7: else
8:     tail.next ← node
9:     node.prev ← tail
10:    tail ← node
11: end if
```

Tiempo de ejecución: O(1)

Algorithm 12 PopBack ()

```
1: if head = Null then
2:     ERROR: empty list
3:
4: else
5:     key ← tail.key
6:     tail.prev.next ← NULL
7:     tail ← tail.prev
8:     return key
9: end if
```

Tiempo de ejecución: O(1)

Algorithm 13 PushFront(key)

```
1: node ← new node(key)
2:
3: if head = Null then
4:     head ← node
5:     tail ← node
6:
7: else
8:     tail.next ← node
9:     node.next ← head
10:    head ← node
11: end if
```

Tiempo de ejecución: O(1)

3 Implementación:

`https://github.com/nicole-nataren/Proyecto2024`