ED

Juan García Candón
2022-2023

Índice

Implementación de TADs....3
Operaciones de los TADs....3
TAD Pila....3
TAD Cola....3
TAD Lista....3
Representación vectorial estática y pseudoestática general....4
Representación dinámica (celdas enlazadas)....4
Representación vectorial circular pseudoestática del TAD Cola....5
Representación mediante una estructura enlazada circular....5
Representación mediante una estructura doblemente enlazada del TAD Lista....5
Cuestiones Generales....6
Preguntas de Desarrollo....6
Preguntas de Verdadero - Falso....7
TAD Pila....8
Preguntas de Desarrollo....8
Preguntas Verdadero - Falso....8
TAD Cola....9
Preguntas de Desarrollo....9
Preguntas Verdadero - Falso....10
TAD Lista....11
Preguntas Desarrollo....11
Preguntas Verdadero - Falso....12

Implementación de TADs

Operaciones de los TADs

TAD Pila

```
Pila(); /*Variará dependiendo de la representación utilizada*/
Pila(const Pila& P);
const Pila& operator=(const Pila& P);
const T& tope() const;
void pop();
void push();
bool vacia()const;
bool llena()const; /*Excepto la representación dinámica*/
~Pila();
```

TAD Cola

```
Cola(); /* Variará dependiendo de la representación utilizada*/
void pop();
void push();
bool llena() const; /*Excepto la representación dinámica*/
bool vacia()const;
const T& frente();
const Cola& operator=(const Cola& c);
Cola(const Cola& c);
~Cola();
```

TAD Lista

```
Lista();
const Lista& operator=(const Lista& L);
Lista(const Lista& L);
void insertar(const T& elto, posicion p);
void eliminar(const T& elto, posicion p);
T& elemento(posicion p);
const T& elemento(posicion p)const;
posicion inicio()const;
posicion final()const;
posicion siguiente()const;
posicion anterior()const;
bool vacia()const;
/*Implementaciones vectoriales*/bool llena()const;
~Lista();
```

Representación vectorial estática y pseudoestática general

Esta representación la comparten:

- Representación vectorial estática del TAD Pila.
- Representación vectorial pseudoestática del TAD Pila.
- Representación pseudoestática del TAD Cola.
- Representación vectorial pseudoestática del TAD Lista

```
T* elementos;   //Vector de elementos
size_t TamMax;  //Tamaño máximo del vector
size_t n_eltos; //número de elementos
```

Representación dinámica (celdas enlazadas)

La estructura de los nodos es la siguiente (misma para todas las representaciones de celdas enlazadas):

```
 struct nodo
{
   T elto;
   nodo* sig;
   nodo(const T& e, nodo* p = nullptr): elto(e), sig(p){}
};
```

Las variaciones de la representación entre los diferentes TADs son las siguientes:

- Para el TAD Pila:

  ```
  nodo* tope_;
  ```

- Para el TAD Cola:

  ```
  nodo* inicio_, fin_;
  ```

- Para el TAD Lista:

  ```
  nodo* L;
  typedef nodo* posicion;
  ```

Representación vectorial circular pseudoestática del TAD Cola

Ventajas:

No se desperdicia memoria durante el uso del TAD Cola mediante esta representación, ya que a diferencia de la representación vectorial aquí movemos los punteros a los elementos en vez de los elementos en sí.

¿Por qué n-1 elementos?

Debido a que la cola es circular, se utiliza un puntero para indicar el inicio de la cola y otro puntero para indicar el final de la cola. Si se permite que ambos punteros se ubiquen en la misma posición, la cola podría estar vacía o llena, lo que dificultaría la distinción entre ambos casos. Por lo tanto, se reserva una posición en el arreglo para distinguir entre cola llena y cola vacía, lo que reduce en uno la cantidad de elementos que se pueden almacenar en la cola.

```
T* elementos; //Vector de elementos
size_t Lmax; //Tamaño del vector
size_t inicio, fin; //Posiciones de los extremos de la cola
```

Representación mediante una estructura enlazada circular

Nota*:* No hace falta almacenar un puntero a la posición del primer elemento ya que este se almacena recursivamente en el “nodo*sig” del último elemento de la cola.

```
 struct nodo
{
   T elto;
   nodo* sig;
   nodo(const T& e, nodo* p = nullptr): elto(e), sig(p){}
};
nodo* fin; //El último nodo de la cola, apunta al primero en vez de
nullptr
```

Representación mediante una estructura doblemente enlazada del TAD Lista

¿ Por qué está doblemente enlazada?

La representación mediante celdas doblemente enlazadas del TAD Lista surgió para superar las limitaciones de la representación mediante celdas simplemente enlazadas. En una lista simplemente enlazada, cada elemento solo tiene un puntero que apunta al siguiente elemento. Esto significa que solo se puede recorrer la lista en una dirección, desde el principio hasta el final. Además, insertar o eliminar elementos en el medio de la lista es ineficiente, ya que se necesitan reorganizar los punteros de los elementos vecinos.

En cambio, en una lista doblemente enlazada, cada elemento tiene dos punteros: uno que apunta al elemento anterior y otro que apunta al siguiente elemento. Esto permite recorrer la lista en ambas direcciones, lo que es útil en muchos casos. Además, insertar o eliminar elementos en el medio de la lista es más eficiente, ya que solo se necesitan reorganizar los punteros de los elementos vecinos, sin tener que recorrer toda la lista desde el principio.

```
 struct nodo
{
   T elto;
   nodo* sig, ant;
   nodo(const T& e, nodo* p = nullptr): elto(e), sig(p){}
};
nodo* L; //El último nodo de la cola, apunta al primero en vez de
nullptr
typedef nodo* posicion;
```

Cuestiones Generales

Preguntas de Desarrollo

1) ¿Qué es un TAD?

2) ¿Qué define el nivel conceptual o de especificación de un TAD?

3) ¿Qué define el nivel de representación o implementación de un TAD?

4) Definición de "*especificación sintáctica*"

5) Definición de "*especificación semántica"*

6) ¿En qué consiste la implementación de un TAD?

7) ¿En qué consiste el principio de independencia de la representación de los TAD?

8) ¿Existe alguna relación entre la ocultación de la información y la independencia de la representación?

9) Ante la posibilidad de que no se verifiquen las precondiciones de una operación de un TAD, ¿Que decisiones de diseño puede implementar una operación?

10) ¿Qué sucede si después de la ejecución de una determinada operación, no se cumplen las precondiciones de la misma?

Preguntas de Verdadero - Falso

11) Es siempre una obligación del diseñador del TAD, al implementar las operaciones del mismo, comprobar que se cumplen las precondiciones, y en caso de no ser así, enviar un mensaje de error y abortar la ejecución de la misma

12) La especificación de un TAD es útil solamente para el usuario, para saber el tipo y las operaciones del mismo al implementar sus programas haciendo uso de él

13) Si no se cumplen las postcondiciones de una operación, implica obligatoriamente que dicha operación está mal implementada.

14) Un usuario de un TAD puede acceder a la estructura del mismo, siempre y cuando respete el principio de la independencia de la representación.

15) Aunque desconozcamos el tamaño máximo del problema, es posible, aunque ineficiente, usar una estructura vectorial pseudoestática.

16) La especificación TAD debe realizarse siempre después de la implementación, para asegurarse de que no se cometen errores.

17) Por el principio de independencia de la representación del TAD, puedo adaptar la especificación del mismo mientras estoy haciendo la implementación

18) La abstracción de datos permite al programador ignorar los detalles de representación interna de los mismos.

19) Por el principio de independencia de representación del TAD, puedo adaptar la especificación del mismo mientras estoy haciendo la implementación.

20) No es conveniente, pero resulta más claro, que en la definición del TAD dada en la especificación se indique la estructura de datos elegida para representar dicho TAD.

21) El uso de la implementación vectorial pseudoestática tiene demasiados inconvenientes y por eso debemos usar siempre una implementación basada en una estructura dinámica.

22) Conociendo el tamaño máximo del problema, es posible, aunque ineficiente, usar una estructura vectorial pseudoestática.

23) Si una operación está bien implementada, deben cumplirse obligatoriamente las postcondiciones al finalizar la misma.

TAD Pila

Preguntas de Desarrollo

1) ¿Qué es una Pila?

2) ¿Cuando utilizamos la representación estática del TAD Pila? Ventajas e inconvenientes.

3) ¿Cuando utilizamos la representación pseudoestática del TAD Pila? Ventajas e inconvenientes.

4) ¿Cuando utilizamos la representación mediante celdas enlazadas del TAD Pila? Ventajas e inconvenientes.

5) ¿Tiene sentido representar una Pila con una estructura doblemente enlazada?

6) ¿Cuándo usaremos la representación circular del TAD Pila?

7) ¿Por qué en las implementaciones mediante celdas enlazadas de pila, no existe un método *“llena()”*?

8) ¿En qué programas usarías el TAD Pila?
9) ¿Cómo se podría eliminar un elemento cualquiera del TAD Pila?

Preguntas Verdadero - Falso

1) Una estructura dinámica ocupará más espacio que una pseudoestática para una pila llena cualquiera.

2) El TAD Pila no incorpora una operación concreta para acceder a un elemento cualquiera de la misma, pero sí podemos acceder a cualquier elemento de la Pila, implementando la operación a partir de las operaciones del TAD.

3) La representación vectorial circular del TAD Pila deja un hueco libre para distinguir cuando está llena o vacía.

4) El TAD Pila no incorpora una operación concreta para acceder a un elemento cualquiera de la misma, pero sí podemos acceder a cualquier elemento de la Pila, siempre y cuando la implementación del TAD sea utilizado para la representación vectorial.

5) Es posible hacer una búsqueda en el TAD Pila, pero es de orden lineal.

6) Una estructura dinámica ocupará más espacio que una pseudoestática para una pila llena cualquiera.

7) En el TAD Pila, puedo guardar un elemento menos debido a que en el nodo cabecera no se guarda nada.

8) Las pilas se usan única y exclusivamente cuando el enunciado nos pide expresamente inventar algo.

9) No tiene sentido implementar una pila mediante una estructura doblemente enlazada porque, aunque el doble puntero es útil, resulta muy costoso en términos de espacio.

10) Si he de escoger entre una estructura simplemente enlazada y doblemente enlazada a la hora de implementar una pila, la decisión dependerá de si alguna operación en concreto se va a realizar con mucha frecuencia o no.

11) El TAD Pila es útil para resolver, entre otros, problemas en los que es necesario procesar los elementos en orden inverso al que se proporcionan.

12) En la representación dinámica del TAD Pila, no es necesario añadir un nodo cabecera, pero si lo ponemos, nos facilita el acceso al elemento que está en el tope.

13) Desde el punto de vista de la eficiencia espacial, siempre es mejor usar una estructura enlazada en una pila que una estructura vectorial.

TAD Cola

Preguntas de Desarrollo

1) ¿Qué es una Cola?

2) ¿Cuándo usaremos una representación estática del TAD Cola?Ventajas e inconvenientes

3) ¿Cuándo usaremos una representación pseudoestática del TAD Cola?Ventajas e inconvenientes

4) ¿Cuándo usaremos una representación Vectorial del TAD Cola?Ventajas e inconvenientes

5) ¿Cuándo usaremos una representación Circular del TAD Cola?Ventajas e inconvenientes

6) ¿Cuándo usaremos una representación Circular mediante celdas enlazadas del TAD Cola?Ventajas e inconvenientes

7) Eres un diseñador del TAD Cola y un usuario te solicita la operación de acceso al n-enésimo elemento de la misma ¿Incluirías está operación en el TAD?

8) ¿Por qué en la implementación vectorial circular de las colas, existe una posición que siempre está vacía en el vector?

9) La representación de una cola mediante estructuras enlazadas, ¿por qué se representa usando dos punteros a cada extremo de la misma?

10) Con la implementación de colas mediante un vector circular de tamaño n ¿cuántos elementos pueden almacenarse?

11) Comente la siguiente afirmación: "*El TAD Cola circular es un TAD representado mediante un vector circular en el que el número de elementos a insertar es como máximo n-1*"

12) ¿En qué programas usarías el TAD Cola?

13) ¿Para que se utiliza el nodo cabecera en el TAD cola en su implementación mediante celdas enlazadas?

14) ¿Es necesario en la representación vectorial circular del TAD Cola el nodo cabecera?

Preguntas Verdadero - Falso

1) En la vida real es prácticamente imposible encontrar situaciones modelables por el TAD Cola.

2) La cola no es una estructura LIFO.

3) En el TAD Cola el frente debe estar siempre a la derecha y el final a la izquierda

4) La implementación vectorial circular del TAD Cola es preferible a la basada en un vector lineal, a pesar de su mayor dificultad por el uso de aritmética modular.

5) En el TAD Cola no es correcto utilizar una representación pseudoestática a menos que se conozca el número mínimo de elementos a almacenar.

6) Implementar el TAD Cola con dos punteros a los extremos es un desaprovechamiento de espacio claro, ya que se puede hacer con sólo uno.
7) En el TAD Cola, puedo guardar un elemento menos debido a que en el nodo cabecera no se guarda nada.

8) En la representación enlazada del TAD Cola Circular es innecesario usar el nodo cabecera.

9) En la representación vectorial circular del TAD Cola, nunca es posible aprovechar todos los elementos del vector ya que no nos permite distinguir entre cola llena y cola vacía.

10) El TAD Cola es útil para resolver, entre otros, problemas en los que es necesario procesar los elementos en orden inverso al que se proporcionan

11) Desde el punto de vista de la eficiencia espacial, siempre es mejor usar una estructura enlazada en una Cola que una estructura vectorial.

12) El TAD Cola debe presentarse obligatoriamente con un vector, o con una estructura doblemente enlazada.

13) No es cierto que en el TAD Cola sea innecesario usar el nodo cabecera.

14) En la representación del TAD Cola mediante vector circular, no hay un límite concreto de elementos que puedan almacenarse, debido al aprovechamiento de espacio.

TAD Lista

Preguntas Desarrollo

1) ¿Que es una Lista?

2) ¿En una representación vectorial de una lista, como representamos la posición "*fin*"?
3) ¿Cuáles son las ventajas e inconvenientes de la implementación vectorial del TAD Lista?.

4) ¿Cuáles son las ventajas e inconvenientes de la representación mediante celdas enlazadas con cabecera del Tad Lista?

5) ¿Cuáles son las ventajas e inconvenientes de la representación mediante celdas doblemente enlazadas con cabecera del TAD Lista?

6) ¿Por qué surge el nodo cabecera en el TAD Lista?

7) ¿Cuándo se utiliza un TAD Lista Circular?

8) ¿Diferencia entre el TAD Lista Circular y TAD Lista?

9) ¿Para que se utiliza una implementación mediante estructura enlazada doble?

10) ¿Cuándo es conveniente utilizar una implementación mediante una estructura doblemente enlazada?

11) En el TAD Lista circular, ¿es posible implementar alguna operación de orden logarítmico?

12) ¿Qué condición tiene que cumplir una lista para que la búsqueda sea logarítmica?¿Y orden cuadrático?

13) ¿Por qué en la especificación del TAD Lista, en la operación anterior, las precondiciones son L = (a1, a2, ..., an) 2 <= p <= n + 1?

14) ¿Por qué en la especificación del TAD LIsta, en la operación siguiente, las precondiciones son L = (a1, a2, .. , an) 1 <= p <= n ?

15) ¿Por qué en la especificación del TAD Lista Circular, en las operaciones anterior y siguiente, las precondiciones son L = (a1, a2, ..., an, a1) 1<=p<=n?

16) ¿Qué pasaría si insertar o eliminar en la primera posición en una implementación de lista mediante estructura enlazada sin cabecera?

17) Representando el TAD Lista mediante un vector ¿es posible evitar el coste O(n), para inserciones y borrados extremos?

18) ¿En qué programas utilizarías el TAD Lista?

Preguntas Verdadero - Falso

1) En el TAD Lista doblemente enlazada la operación anterior() es de coste constante.

2) La operación anterior() del TAD Lista simplemente enlazada es de orden lineal.
3) En el TAD Lista, el coste de la operación anterior() es lineal si utilizamos la representación simplemente enlazada con nodo cabecera

4) En el TAD Lista, el coste de la operación anterior() es constante si utilizamos la representación simplemente enlazada con nodo cabecera

5) En la implementación del TAD Lista con nodo cabecera, la operación anterior() ya es por definición de coste constante, dado que, en este caso, físicamente apuntamos el nodo anterior al que apuntamos lógicamente

6) No se puede implementar la operación buscar() del TAD Lista Ordenada en orden logarítmico, salvo que la representación sea vectorial

7) Es posible realizar búsquedas en una lista en orden logarítmico, pero exigen que los elementos de la lista estén ordenados.

8) No es posible realizar búsquedas en una lista en orden logarítmico, aunque los elementos de la lista están ordenados.

9) La operación fin() del TAD Lista de orden constante en la representación vectorial, igual que en la enlazada con un puntero al último nodo.

10) La operación fin() del TAD Lista es de orden constante en la representación vectorial, igual que en la enlazada, obviamente si tenemos un puntero apuntando allí.

11) La operación fin() del TAD Lista es de orden lineal en la representación vectorial, frente a coste constante en la representación enlazada, obviamente si tenemos un puntero apuntando allí.

12) En el TAD Lista no existe ninguna forma de implementar la operación fin() en coste de O(1) en ninguna representación enlazada.

13) En el TAD Lista no existe ninguna forma de implementar la operación fin() en coste de O(1) en la representación vectorial

14) En el TAD Lista Circular, el coste de la operación fin() está en O(1).

15) En el TAD Lista Circular la operación fin() es de coste O(n).

16) Si los elementos de una lista están ordenados, podemos asegurar un coste lineal en la búsqueda, y el tiempo en caso promedio será mejor que cuando no están ordenados.

17) La primera posición en el TAD Lista Circular es la siguiente a la posición “*fin”*.

18) Si he de escoger entre una estructura simplemente enlazada y doblemente enlazada a la hora de implementar una lista, la decisión dependerá de si alguna operación en concreto se va a realizar con mucha frecuencia o no.