Estructuras de datos lineales

Clasificación de grafos

En general clasificamos a un grafo G (P, E, f_p, g_e) como: lineal, árbol, acíclico o irrestricto.

Grafo irrestricto

Por definición decimos que todo grafo es irrestricto, siendo este el conjunto que engloba a todos los grafos estudiados.

Grafo acíclico

Un grafo G es acíclico si no tiene ningún ciclo, es decir:
$\forall x \in G: (x,x) \notin E$.

Árboles

Esta definición será dada más adelante

Grafo lineal

Un grafo es lineal cuando la relación de paso inducida es de orden total y además el grafo original es básico.

Teorema:

Un grafo G (P,E) es lineal si se cumple que:

a) $|A|=1$ donde $A = \{ x/ x \in P \wedge L(x)=\emptyset\}$ (existe sólo un nodo al que no llegan arcos)
b) $\forall y \in P: |L(y)| \leq 1 \wedge |R(y)| \leq 1$ (para todos los nodos, como máximo llega un arco, y como máximo sale un arco)
c) G es conexo

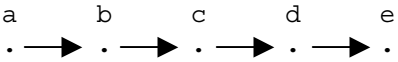
Recordemos que un conjunto está totalmente ordenado si está parcialmente ordenado y además cumple con la dicotomía ($\forall x,y \in P: (x,y) \in E \vee (y,x) \in E$)

Ejemplos:

Sobre el conjunto de los números enteros se puede definir un orden total donde G (P,E) se define como:

P = {x/x es un número entero}
E = {(x,y)/ y=x+1 }

El siguiente grafo cumple con las condiciones de linealidad:



El siguiente grafo no cumple con las condiciones de linealidad, ya que |L(c)|=2 y E no es de orden parcial:

Observaciones:

Cada conjunto definido está incluido dentro del anterior. Por ejemplo, todo grafo lineal es arbol y acíclico. En consecuencia (y como se verá mas adelante) se puede representar un grafo lineal en forma de lista o en forma de vector, formas que sólo sirven si G cumple con las condiciones de linealidad, pero también se pueden usar formas árboreas, matrices de adyacencia / incidencia, u otra representación para grafos irrestrictos, por ejemplo la representación de Pfaltz.
Cada definición brinda más información acerca del grafo, ya que dice que se pueden usar las representaciones para ese tipo, las que serán más eficientes en términos de complejidad espacial y computacional (no tiene sentido usar una matriz donde se puede utilizar simplemente un vector).

Estructuras de datos lineales

Las EDL son las estructuras de datos utilizadas para representar a los grafos lineales. Son probablemente las estructuras de datos más utilizadas.

Clasificación de las Estructuras de Datos Lineales

ESTÁTICAS	DINÁMICAS
Vector	Lista
	Pila
	Cola

Las estructuras lineales dinámicas, no tienen restricciones de tamaño por sí mismas, sino dependiendo de la memoria disponible.
En cambio, las estructuras estáticas por definición tienen restricciones de tamaño.

Conceptos de Overflow y Underflow

Overflow:
Se da en una estructura cuando se quiere dar de alta más delementos de los que la estructura o de los que la memoria de la computadora permite.
Ej.: - En un vector de 10 elementos, se producirá un "OVERFLOW" al querer dar de alta el elemento nro. 11.
- En una lista enlazada, se podrán dar tantas altas como memoria disponible haya.

Underflow:

Se produce al querer realizar una baja en una estructura que está vacía.

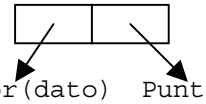
Concepto de puntero

Variable cuyo contenido es un número (entero u otro) representando una locación en memoria donde se encuentra el objeto en sí.

LISTAS - Linkeo simple

Formato de celda:

Linkeo simple



Descriptor(dato) Puntero (sigte)

No tiene restricciones para dar altas o bajas. Pueden ser ORDENADAS o IRRESTRICTAS.

LISTAS - Linkeo doble

Formato de celda:

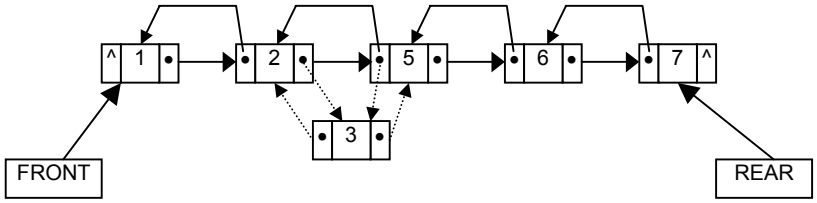
Linkeo doble

ANT: apunta a la celda anterior
POST: apunta a la celda siguiente

FRONT: variable que apunta a la primera celda de la lista
REAR: variable que apunta a la última celda de la lista

No tiene restricciones para dar altas o bajas. Pueden ser ORDENADAS o IRRESTRICTAS.

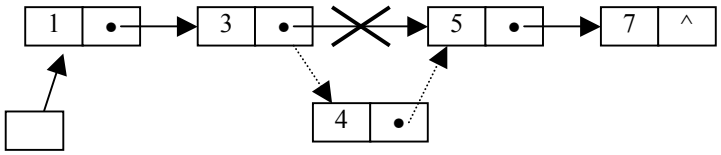
Ejemplos



LISTAS ORDENADAS

Cada elemento debe guardar un orden determinado con el elemento anterior y el posterior.

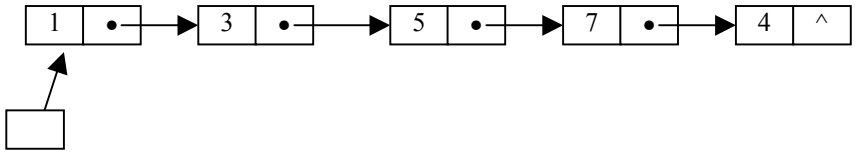
Ejemplo:



LISTAS IRRESTRICTAS

Los elementos no guardan ningún orden.

Ejemplo

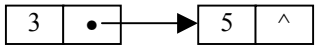


SUBLISTAS

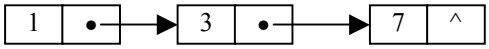
L' es sublista de L si y sólo si L' es un subgrafo conexo de L.

Ejemplo

Dada la lista L:



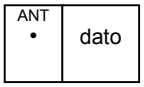
Es sublista de L



No es sublista de L

PILAS

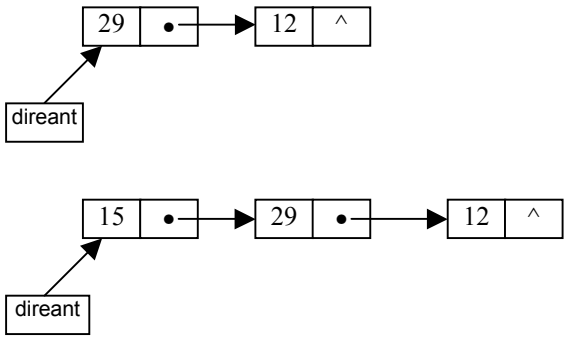
Formato de celda:



ANT: apunta a la celda anterior en la pila

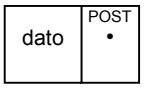
DIREANT: variable que apunta a la última celda de la pila (contiene la dirección de la última celda dada de alta)

Ejemplo



COLAS

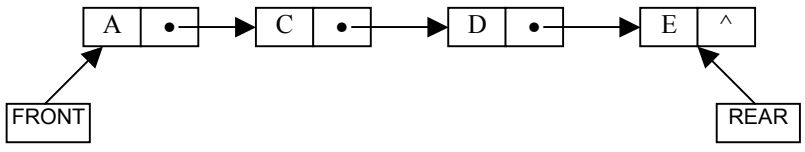
Formato de celda:



POST: apunta a la celda siguiente en la cola

FRONT: variable que apunta a la primera celda de la cola
REAR: variable que apunta a la última celda de la cola

Ejemplo



Altas (por rear)

```
NEW(COLA)
APUNTO := REAR
REAR := COLA
```

```
APUNTO.^DIR := REAR
REAR.^DIR := NIL
```

Bajas (por front)

```
APUNTO := FRONT
FRONT := APUNTO.^DIR
DISPOSE(APUNTO)
```

COLAS DOBLES

Definición

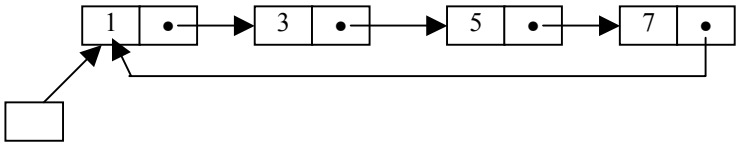
Funciona igual que una lista doble, pero con la restricción de que sólo se pueden dar altas o bajas por los extremos.

Formato de celda:

ANT: apunta a la celda anterior en la cola
POST: apunta a la celda siguiente en la cola

FRONT: variable que apunta a la primera celda de la cola
REAR: variable que apunta a la última celda de la cola

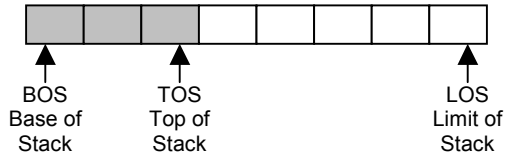
LISTAS CIRCULARES



APLICACIONES EN VECTORES

Implementación de Estructuras Lineales dinámicas, en estáticas.

Pila en un vector

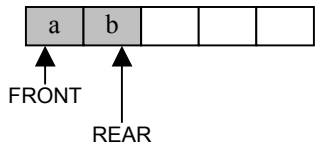


Overflow: TOS > LOS
Underflow: TOS < BOS

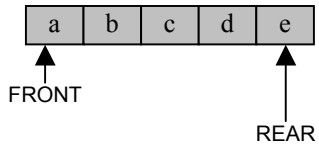
Cola en un vector

Ejemplo:

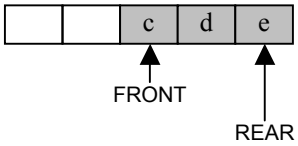
- alta de a
- alta de b



- alta de c
- alta de d
- alta de e



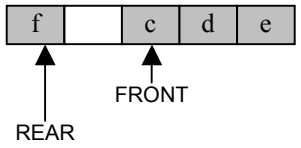
- baja de a
- baja de b



- alta de f

La estructura no está completamente llena, por lo tanto podríamos realizar la nueva alta, del elemento f. Sin embargo, para esto, tenemos que tomar una decisión entre dos alternativas: la primera, es realizar un corrimiento de toda la estructura a medida que se van dando las bajas.

La otra alternativa, es convertir la cola en circular.



REAR = 6
FRONT = 3

Se observa que los apuntadores se invirtieron, lo que se denomina "Wrapped Around". De esta forma, se aprovecha al máximo el espacio disponible en el vector.

n = cantidad máxima de posiciones del vector
i = posición del vector a utilizar en la operación

Alta: $i = mod_n(rear) + 1$
Baja: $i = mod_n(front)$

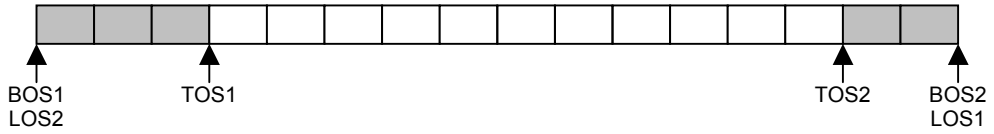
FREE = cantidad de celdas libres

DIF = variable intermedia para calcular el valor de FREE

DIF = REAR - FRONT

Overflow: $mod_n(rear) + 1 = mod_n(front)$
Underflow: $mod_n(front) + 1 > mod_n(rear)$

Dos pilas enfrentadas en un vector



	PILA 1	***PILA 2***
Overflow	TOS1 >= TOS2 ∨ TOS1 >= LOS1	TOS2 <= TOS1 ∨ TOS2 < LOS2
Underflow	TOS1 < BOS1	TOS2 > BOS2