¿Qué es un algoritmo?

Es una secuencia de pasos finitos que explican como resolver un tipo de problema de manera inequívoca, sin ambigüedades

¿Qué es un programa?

Un algoritmo codificado en un lenguaje de programación que es entendido y ejecutado por una computadora

¿Qué es un proceso?

Es un programa en ejecución. Posee un estado que depende del valor que van tomando sus variables

¿Qué es la abstracción?

Es el proceso donde se realiza una observación de una entidad de la realidad, y se relevan los aspectos más importantes de la misma, y se llevan estos datos al sistema para almacenarse en forma de estructuras de datos

¿Qué es la recursividad?

Es una técnica que permite definir una función en términos de sí misma. En otras palabras: una función es recursiva cuando se invoca a sí misma. Por lo tanto, está formado por funciones que se llaman a sí mismo

Que es un arreglo

Estructuras de datos que almacenan elementos del mismo tipo en bloques contiguos de memoria. El tamaño del array debe ser conocido en tiempo de compilación o dinamico

Como ventaja tiene Acceso rápido a los elementos mediante índices.

Los arrays son la implementación más básica de almacenamiento de datos y a menudo se utilizan como base para otras estructuras de datos más complejas.

¿Qué es una estructura de datos?

Entidad que almacena la información necesaria por el programa para utilizar en un proceso.
Tipos de estructuras de datos:

- **Simples:** permiten guardar un único valor, del mismo tipo especificado.
- **Compuestos:** permiten guardar más de un valor, pudiendo ser un arreglo

¿Qué es un puntero? Asignación y desreferenciacion

Un puntero es una variable que almacena el valor de la dirección de memoria de otra variable. Es una variable que “apunta” a otra variable

Se utilizan para el manejo de memoria dinámica. Podemos asignarle a un puntero la dirección de otra variable y también podemos desreferenciarlo, proceso donde devolvemos el valor de la variable a la cual está apuntando

Defina Estructuras de datos lineales.

Pilas

Son estructuras lineales de entradas ordenadas, donde se añaden y extraen los elementos por el mismo extremo, de manera que el ultimo elemento en ingresar es el primero que sale (LIFO). Son de acceso destructivo, para acceder a algún elemento debo sacarlo de la pila. Se pueden implementar utilizando arreglos o listas enlazadas

Poseen 2 operaciones, empilar (ingresar elemento en la pila9 y desempilar (sacar elemento). Puede tener dimensión estática o dinámica (funcionando como una LSE). Si la dimensión es fija puede estar llena o vacía. Si es dinámica su tamaño va creciendo

Colas

Estructura lineal de entradas ordenadas, donde se añaden los elementos por un extremo y se eliminan por el otro, de manera que el primer elemento que ingresa es también el primero que sale (FIFO). Son de acceso destructivo, para acceder a algún elemento debo sacarlo de la cola.

Posee 2 operaciones, encolar y desencolar. Puede tener dimensión fija o dinámica (funcionando como una LSE). Si la dimensión es fija la cola se puede llenar. También puede estar vacía. Si es dinámica, su tamaño va creciendo.

Listas

Estructura lineal donde los elementos se almacenan de forma secuencial. Cada elemento o nodo de la lista tiene 2 partes, la información y un enlace que apunta al siguiente nodo. Por lo tanto, el orden de una lista este dado por sus enlaces. El ultimo nodo de la lista tiene un enlace a null, que indica el final de la lista

El acceso no es destructivo. Se puede consultar los valores de cada nodo sin necesitar sacarlos de la lista, ahora este acceso es de manera secuencial a diferencia de un arreglo, por lo que debemos recorrer cada nodo hasta llegar al que queremos consultar.

Posee 3 operaciones básicas; agregar nodo, quitar nodo y consultar nodo.

Tenemos varios tipos de listas

- **LSE**: enlace a siguiente nodo
- **LDE**, cada nodo tiene enlace al siguiente y al anterior nodo, excepto el primero que su predecesor es NULL, y el ultimo
- **LCSE**, cada nodo apunta al siguiente, y el ultimo nodo apunta al primero de manera que se puede recorrer toda la lista de manera circular hacia adelante
- **LCDE**, cada nodo tiene enlace al siguiente y al anterior nodo. El ultimo también apunta al primero y viceversa. Por lo que se recorre de manera circular en ambos sentidos.

Defina Estructuras estáticas y dinámicas.

Son dos tipos específicos de estructuras de datos lineales que se diferencian en la forma en que gestionan la memoria:

Las estructuras estáticas son aquellas donde se debe definir el tamaño en memoria que tendrán previo a la ejecución del programa, durante la compilación o declaración. El mismo se mantiene estático, no puede crecer ni disminuir por lo que estas estructuras se pueden llenar.

Las variables dinámicas son aquellas donde su tamaño va variando en tiempo de ejecución.

¿Puede un árbol tener un nodo que es hoja y raíz al mismo tiempo?

No, un árbol no puede tener nodos hoja (sin hijos) y que sean raíz (nodo principal) a la vez del mismo árbol.

En cambio, si consideramos los subárboles, un árbol puede tener un nodo hoja, y a su vez este nodo hoja será raíz del subárbol que contiene al mismo como único nodo.

¿Puede un árbol NO tener hijos?

Si, un árbol puede no tener hijos. Sería un árbol con un único nodo que hace de raíz

¿Puede un nodo tener dos hijos?

Si, un nodo puede tener n hijos en una estructura de tipo árbol. En arboles binarios, como máximo 2 hijos

Defina Sub arboles

Un árbol se divide en subárboles. Un subárbol es cualquier estructura conectada por debajo de la raíz. Cada nodo de un árbol es la raíz de un subárbol que se define por el nodo y todos o algunos de los descendientes del mismo.

Diferencia árbol B y B+

Tanto los arboles B como los arboles B+ son arboles enarios que pueden almacenar varias claves en cada nodo. Estas claves están ordenadas al igual que un árbol binario. Todas las hojas están al mismo nivel.

Poseen un grado p, y cada nodo tiene como máximo (p/2)-1 claves, y tiene como mínimo p/2 punteros a hijos

La diferencia entre estos es que el árbol B+, todos los punteros a registros se almacenan en las hojas: al tener toda la información en las hojas, permite buscar datos de manera más eficiente.

Esto se hace cuando se divide una hoja durante la inserción de un elemento, este deja una copia del padre en el nodo izquierdo. Los nodos intermedios y la raíz solamente almacenan claves o índices para ordenar la búsqueda. Cada nodo hoja tiene punteros al nodo hoja a su derecha

Base de comparación	árbol B	árbol B+
Punteros	Todos los Nodes internos y secundarios tienen punteros de datos.	Solo los Nodes de hoja tienen punteros de datos
Búsqueda	Dado que no todas las claves están disponibles en la hoja, la búsqueda suele llevar más tiempo.	Todas las claves están en los Nodes hoja, por lo que la búsqueda es más rápida y precisa.
Teclas redundantes	No se mantiene ningún duplicado de claves en el árbol.	Se mantienen duplicados de claves y todos los Nodes están presentes en la hoja.
Inserción	La inserción lleva más tiempo y, a veces, no es predecible.	La inserción es más fácil y los resultados son siempre los mismos.
Supresión	La eliminación del Node interno es muy compleja y el árbol debe sufrir muchas transformaciones.	La eliminación de cualquier Node es fácil porque todos los Nodes se encuentran en la hoja.
Nodes de hoja	Los Nodes hoja no se almacenan como lista enlazada estructural.	Los Nodes hoja se almacenan como lista enlazada estructural.
Acceso	El acceso secuencial a los Nodes no es posible	El acceso secuencial es posible al igual que la lista enlazada

Barridos

Es un algoritmo que devuelve la secuencia de todos los nodos contenidos en un árbol. Tenemos 4 tipos, los primeros 3 solo disponibles en arboles binarios:

- Pre orden (RID): primero leemos la raíz, luego el hijo izquierdo y luego el hijo derecho
- Post Orden (IDR)
- In Orden (IRD)
- Por niveles

Nombre las diferencias y ventajas entre lista circular y cola circular. Utilice gráficos para su explicación

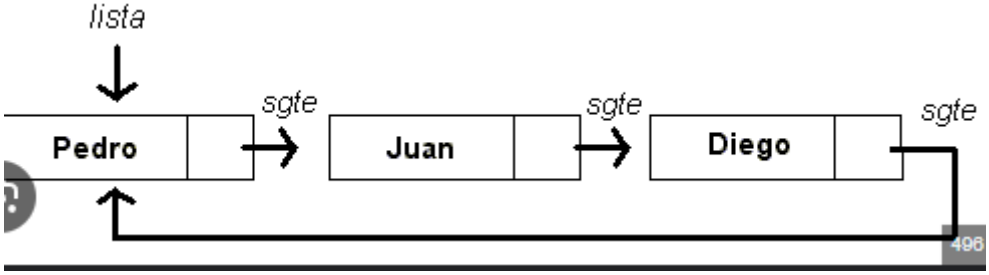
La principal diferencia entre una lista circular y una cola circular esta en el acceso. La cola es de acceso destructivo por lo que para acceder a un elemento debo sacarlo de la cola, además que solo puede sacar el que esta al frente de la cola (el primero en ser ingresado).

En la lista puedo recorrerla secuencialmente y acceder a los elementos que yo necesite. Puede recorrer la misma en forma circular hacia adelante.

La otra diferencia esta en su tamaño. La lista circular tiene tamaño variable, puede ir creciendo a lo largo del programa. La cola circular no, por lo que se puede llenar

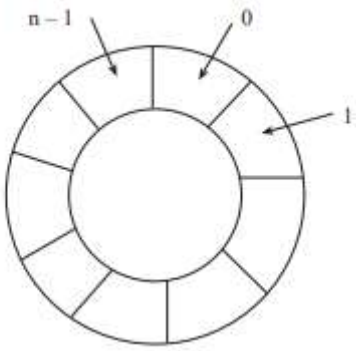
Las ventajas de una **lista circular** son:

- Flexibilidad en el acceso. No necesito remover los elementos para verlos
- Tamaño variable
- Se adapta a diferentes necesidades de manipulación de datos



Ventajas de una **Cola circular**:

- Mayor utilidad en aplicaciones donde el orden de llegada es importante
- Útil cuando se quiere trabajar con tamaños máximos de estructura
- Eficiente para situaciones donde se necesita un uso continuo y cíclico de los elementos



Diferencia cola circular(estática) y cola (lse)

La diferencia entre cola y cola circular es que la cola tiene tamaño indefinido. Puede ir aumentando su tamaño durante la ejecución del programa. Se implemente mediante una LSE. En cambio, la cola circular tiene frente y final. Puede llenarse y a partir de entonces cada nuevo ingreso se hará eliminando el primer elemento ingresado. Se implementa con arreglos estáticos.

Defina que es un árbol AVL y que no es. Utilice gráficos para su representación

Un árbol AVL es un árbol binario de búsqueda al que se le añade un factor de equilibrio a cada nodo. La diferencia de alturas entre los subárboles izquierdo y derecho de cualquier nodo es como máximo uno, lo que garantiza que la altura total del árbol nunca supere lógicamente el número de nodos en el árbol

Este árbol es Auto balanceado: Después de realizar inserciones o eliminaciones, el árbol se reorganiza automáticamente para mantener su propiedad de balance.

Que no es un árbol AVL

No se considera árbol AVL a un árbol binario de búsqueda en que la diferencia de altura entre los subárboles izquierdo y derecho es mayor a 1, ya que no cumplen la propiedad de balance de árbol AVL

Nombre cuál le parece las ventajas de usar un ABB y cuáles son sus desventajas

Un Árbol Binario de Búsqueda (ABB) es una estructura de datos que proporciona un acceso eficiente, inserción y eliminación de datos ordenados

Ventajas

- Inserción y eliminación ordenada: Los datos se mantienen ordenados en la inserción y en la eliminación, además de ser más eficientes
- Búsqueda eficiente: Facilita la búsqueda de elementos ya que se realiza en tiempo logarítmico promedio. El máximo número de comparaciones que necesitaríamos para saber si un elemento se encuentra en un árbol binario de búsqueda estaría entre $[\log_2(N+1)]$ y N, siendo N el número de nodos.
- Flexibilidad de uso: pueden adaptarse a cualquier tipo de datos que admitan orden por valores
- Estructura simple, son mas simples de implementar y comprender que otros tipos de arboles

Desventajas

- Sensibilidad a la distribución de los datos: si los datos se insertan en orden ascendente o descendente, esto provoca que el árbol quede desequilibrado, afectando a la eficiencia
- Complejidad de equilibrado: el costo de equilibrar un árbol puede ser alto en cuanto a las rotaciones
- Mayor espacio de almacenamiento: los ABB pueden ocupar mas espacios que otro tipo de estructuras como las listas

¿Por qué un árbol binario es más eficiente que una lista enlazada?

1. En un árbol binario bien diseñado el tiempo de búsqueda de un elemento es proporcional al logaritmo del número de elementos. Por el contrario, una lista enlazada requiere un tiempo de búsqueda lineal, ya que necesita recorrer cada elemento secuencialmente hasta encontrar el elemento deseado.

2. Al insertar o eliminar un elemento en un árbol binario, el árbol se puede reestructurar para mantener su equilibrio, por ejemplo, usando rotaciones. Esto asegura que el árbol queda equilibrado y mantiene sus posibilidades de recorrerlo eficientemente.

   En cambio, en una lista vinculada, las operaciones de inserción y eliminación requieren estar actualizando los punteros de los nodos adyacentes, lo que puede ser más lento para listas grandes o cuando se desconoce la posición del elemento.

3. Con un árbol de búsqueda binaria (BST), los elementos se pueden insertar de una manera que mantenga el orden de clasificación, lo que permite realizar búsquedas y consultas de rango eficientes.

   En la lista vinculada, para mantener ese mismo orden requieres recorrer la lista para encontrar la posición justa para la inserción, lo que genera una ineficiencia en tiempos de proceso.

¿Qué es un árbol?

En una estructura de datos NO lineales y dinámicas., cada elemento puede tener más de un antecesor y/o sucesor (árboles y grafos).

Un árbol se puede definir como una estructura jerárquica aplicada sobre una colección de elementos conocidos como nodos, donde cada uno tiene un padre salvo el nodo raíz que no tiene padre. No existen elementos “aislados”.

Estas ED son estructuras recursivas, ya que cada nodo interno forma un árbol, y un nodo de ese subárbol forma otro, y así sucesivamente.

Definición formal:

Un árbol es una colección de elementos llamados “nodos”, uno de los cuales es la “raíz”. Existe una relación de parentesco por la cual cada nodo tiene uno y solo un “padre”, salvo la raíz que no tiene.

Propiedades de arboles

Equilibrado: Cada nodo h-2 (nodo nieto) tiene el máximo de hijos que puede (grado)

Perfectamente equilibrado / lleno: un árbol está lleno cuando cada nodo excepto las hojas tienen todos los hijos que puede tener y todas las hojas están al mismo nivel

Balanceado: En cada nodo la diferencia entre la altura del subárbol izquierda y derecha es <=1.

Perfectamente balanceado: Si en cada nodo la altura de sus subárbol izquierda y derecha son iguales

Completo un árbol esta completo cuando cada nodo excepto las hojas tienen la máxima cantidad de hijos o no tiene ninguno

---

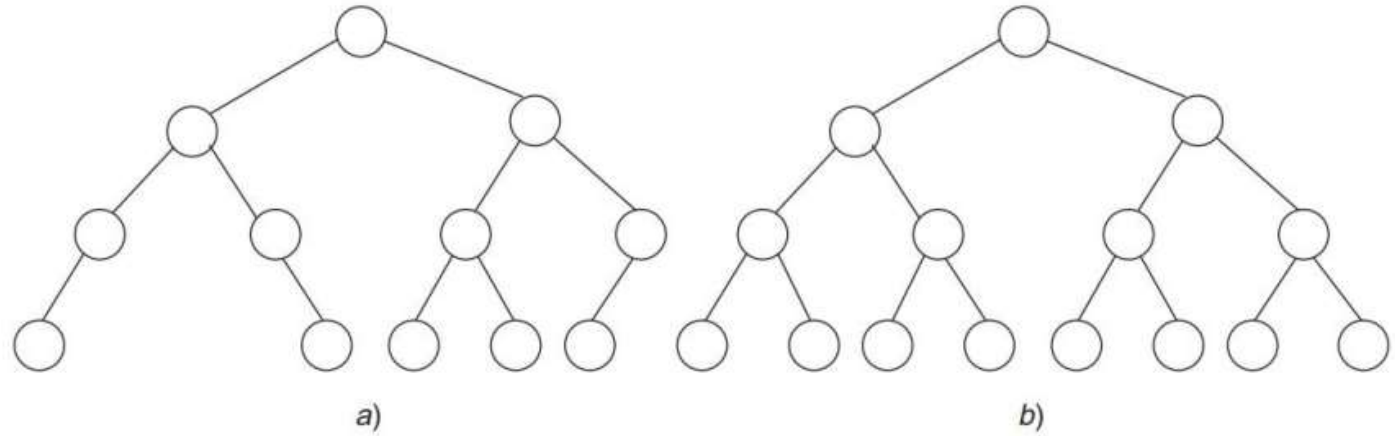
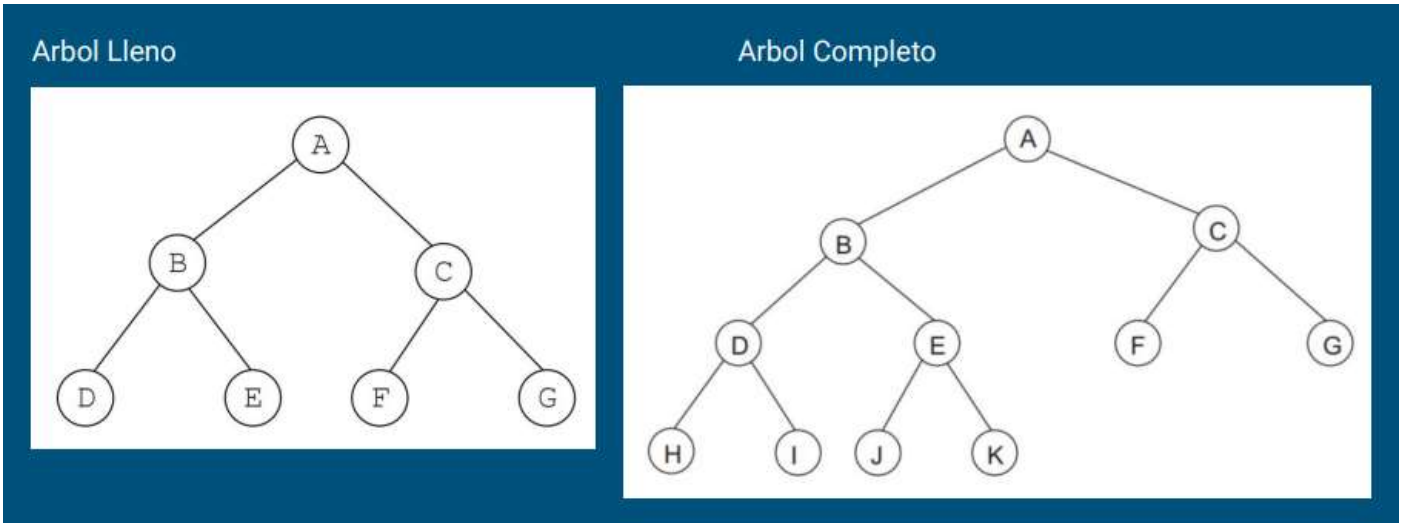


Figura 16.7. *a*) Un árbol equilibrado; *b*) Un árbol perfectamente equilibrado.

Árbol lleno vs completo



- En cualquier nivel n, un árbol de grado G puede contener de 1 a G^n nodos.
- En total, un Árbol de grado G y altura A (distinta de 0) puede contener un mínimo de 1 nodo y un máximo de G^A - 1 nodos.

Clasificación de arboles

Completos: un árbol esta completo cuando cada nodo excepto las hojas tienen la máxima cantidad de hijos o no tiene ninguno

Degenerados: cada nodo incluyendo la raíz tiene solo 1 hijo, excepto las hojas. Se asemeja a una LSE

Binario: cada nodo tiene como máximo 2 hijos

Binaria búsqueda cada nodo tiene como máximo dos hijos. Además, los valores de cada nodo del subárbol izquierdo son menores que la raíz, y los valores de cada nodo del subárbol derecho son mayores que la raíz

AVL: árbol binario de búsqueda que tiene la propiedad de auto balancearse cuando la diferencia de altura entre sus subárbol izquierdo y derecho es mayor a 1.

B: árbol enario de orden p, donde cada nodo puede almacenar como máximo p/2-1 claves, y tener p/2 hijos. Se mantiene ordenado al igual que los árboles binarios

B+: árbol B con la particularidad de que toda la información se almacena solamente en las hojas, teniendo en sus nodos intermedios y raíz claves o índices que favorecen a la búsqueda de datos.

Implementación computacional

Árbol Principal izquierdo: los punteros van de los hijos a los padres. Por lo que se tiene 1 solo puntero por nodo

Árbol principal derecho: los punteros van de los padres a los hijos. Se requiere tantos punteros como hijos se tenga

Diferencia entre altura y profundidad del árbol o nodo. Grado

La **altura** de un árbol se entiende como la altura de la raíz. Esto seria la longitud del camino desde la hoja de mayor profundidad (las del último nivel), hasta la raíz. Es decir, se cuenta de abajo hacia arriba

La **profundidad** de un nodo es la longitud del camino desde la raíz hasta el mismo. Se cuenta de arriba hacia abajo

El **grado** de un árbol es el máximo de hijos que puede tener cada nodo

Definición grafos

Se puede hacer una definición por comprensión y extensión:

- **Comprensión:** es un conjunto P de puntos y un conjunto R de relaciones, con P igual al conjunto X de elementos, tal que X es un nodo y R igual al conjunto de pares (X,Y) tal que tanto X como Y pertenecen al conjunto P de puntos, y X se relaciona con Y
- **Extensión:** es necesario tener un grafo hecho, un grafo determinado, por ejemplo P={A,B,C,D} y R={(A,B), (C,D), (D,B)}, es decir, se expresan las relaciones
- Funciones de asignación a nodo son las características o propiedades del nodo, según el tipo del nodo. Solo debo conocer el nodo
- Funciones de asignación de arco son las características o propiedades del arco, según el tipo de arco. Debo conocer los nodos que se unen

Diferencias hay entre la “asignación estática de memoria” y la “asignación dinámica de memoria”. Proponga ejemplos donde declare variables de uno y otro tipo. Mencione por lo menos una ventaja y una desventaja de cada una de las técnicas analizadas.

Asignación estática: es la que se reserva en tiempo de compilación. Se declara explícitamente el tamaño que tendrá el arreglo y el programa asigna la memoria correspondiente. Como ventaja, es más fácil de implementar, el programa se encarga se liberarla una vez termina su uso, y evita posibles errores por parte del programador. Como desventaja, su tamaño no puede crecer durante la ejecución, lo que muchas veces nos limita su uso o nos obliga a sobredimensionar un arreglo, lo que suele desperdiciar memoria.

Ej: Int n = 5; Pila p;

Asignación dinámica: es la que se reserva en tiempo de ejecución. No se sabe inicialmente el tamaño del arreglo, por lo que se va asignando memoria a mediad que el programa la necesita. En c++ se implementa mediante punteros. Como ventaja, su tamaño puede variar durante la ejecución, dándonos mayor flexibilidad de uso. Como desventaja, es más difícil de implementar, se requiere conocimientos técnicos mas avanzados para manipularla eficientemente, y puede afectar al rendimiento

Ej:

int *n; nodo_pila *p;

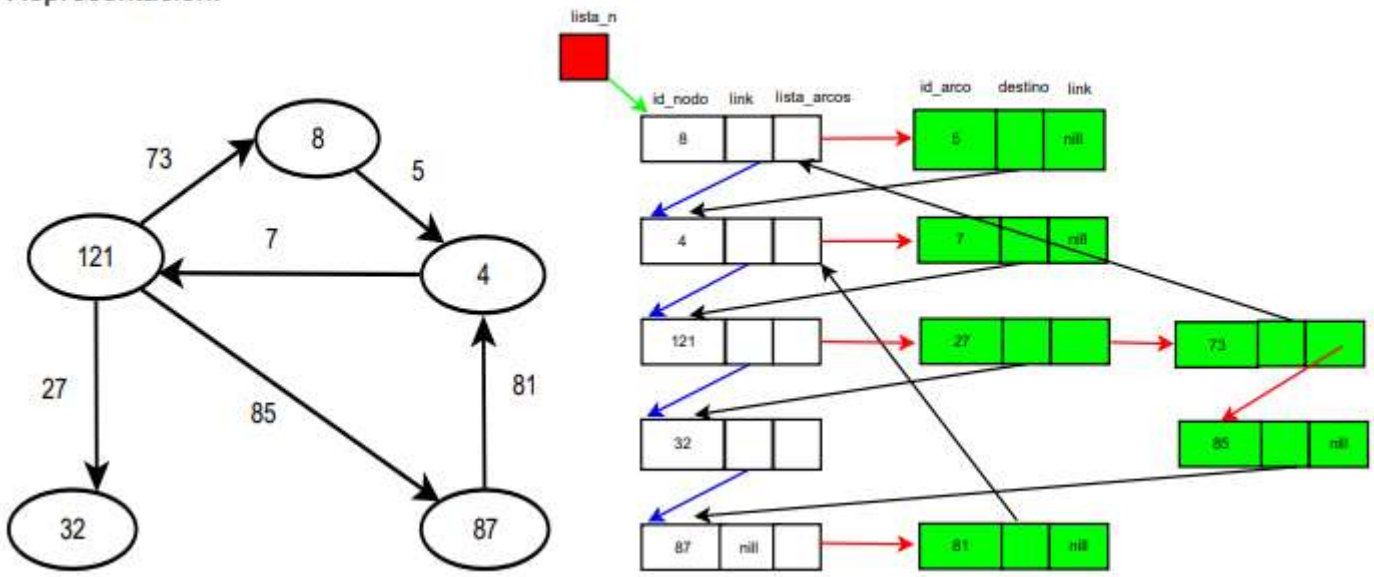
n = new int; p = new nodo_pila;

Lista de adyacencia

Es una estructura multienlazada formada por una "tabla directorio" donde cada nodo representa un elemento (vértice) del grafo, del que emerge una lista enlazada con todos sus vértices adyacentes. Las listas de adyacencia no son ordenadas.

- No necesito conocer la cantidad de nodos con anterioridad
- Optimiza el uso del almacenamiento
- Es un poco más complejo las consultas y el mantenimiento del grafo.

Representación:



Definición de algoritmo, programa y proceso

Algoritmo: secuencia de pasos finitos generales que detallan la solución a un tipo de problema de manera inequívoca sin ambigüedades

Programa: es un algoritmo codificado en un lenguaje de programación que la máquina puede leer e interpretar.

Proceso: es un programa que se encuentra en ejecución. Tiene distintos estados dependiendo del valor de sus variables

Abstracción: proceso en el que se hace una observación sobre una entidad de la realidad con la que se trabaja, tomando nota de sus aspectos mas relevantes para almacenarlos luego en una estructura de datos, a ser utilizados luego por un programa.

Definición de Árbol B y las ventajas y desventajas del árbol B+

Árbol B: árbol de búsqueda completamente equilibrado y eficiente. Son enarios, y cada nodo puede tener varias claves

- Poseen un orden p, que indica el máximo de hijos de cada nodo (punteros)
- Cada nodo tiene como mínimo p/2 hijos

- Cada nodo tiene como **mínimo** (p/2)-1 claves
- Cada nodo tiene como **máximo** p-1 claves
- Todas las hojas están al mismo nivel

Árbol b+

- Permite repetición de nodos un máximo de 2 veces, una en nodo hijo y una en padre
- Puntero a registro solo está en los nodos hojas.

Ventajas árbol B

- Ocupa menos memoria

Ventajas árbol B+

- B**úsqueda** mas eficiente al tener todos los datos en las hojas, sobretodo en árboles de muchos niveles
- Permite **lectura secuencial** al estar sus hojas conectadas por enlaces
- La información se encuentra ordenada
- La **operación** de eliminación en árboles-B+ es más simple que en árboles-B. Esto ocurre porque las claves a eliminar siempre se encuentran en las páginas hojas.
- Para la misma cantidad de nodos, suelen ser **menos profundos** (menos niveles) que los arboles B

Desventajas

- Puede ocupar mas memoria debido a las claves que se repiten

Definición formal de paso, camino, conjunto minimal y left de un nodo

Paso $\rho(x,z)$ es la secuencia $\langle y_0, y_1, \ldots, y_n \rangle$ $n \geq 0$ /

1. $x = y_0 ; z = y_n$ 2. $y_{i-1} \neq y_i$ 3. $(y_{i-1}, y_i) \in R$ $1 \leq i \leq n$

$|\rho(x,z)|$ = nº de arcos entre x y z

Camino: $C(x,z)$ es la secuencia $\langle y_0, y_1, \ldots, y_n \rangle$ $n \geq 0$ /

1. $x = y_0 ; z = y_n$ 2. $y_{i-1} \neq y_i$ 3. $(y_{i-1}, y_i) \in R \vee (y_i , y_{i-1}) \in R$ $1 \leq i \leq n$

$|C(x,z)|$ = nº de conexiones entre x y z

Ciclo: $|\rho(x,x)| \geq 2$ Circuito: $|C(x,x)| \geq 2$ Loop: $|\rho(x,x)| = 0$

$L(x) = \{y / y \in P; (y,x) \in R\}$ $R(x) = \{z / z \in P; (x,z) \in R\}$

$\overline{L(x)} = \{y / y \in P; \exists \rho(y,x)\}$ $\overline{R(x)} = \{z / z \in P; \exists \rho(x,z)\}$

$|L(x)|$ = cantidad de arcos que llegan a x $|R(x)|$ = cantidad de arcos que salen de x

Minimal = $\{ x / x \in P , |L(x)| = 0 \}$ Maximal = $\{ z / z \in P , |R(z)| = 0 \}$

Mínimo = x es mín si $|L(x)| = 0 \wedge$ x es único. Máximo = z es máx si $|R(z)| = 0 \wedge$ z es único.

Grafo Básico: 1. Libre de loops. 2. $\forall x,y \in P$, si $\exists |\rho(x,y)| \geq 2 \Rightarrow (x,y) \notin R$

Grafos Isomorfos: dos grafos $G_1 = (P_1,R_1)$ $G_2 = (P_2,R_2)$ son isomorfos $G_1 \cong G_2$ si $\exists \varphi: P_1 \rightarrow P_2$ /

$\forall x,y \in P_1$: $(x,y) \in R_1 \Leftrightarrow (\varphi(x), \varphi(y)) \in R_2 \wedge \varphi(x) , \varphi(y) \in P_2$

Subgrafo: dado G=(P,R) G' = (P',R') será subgrafo de G si :

Problemas

1. **una impresora que tenía un máximo de 70 páginas para imprimir y se imprimían en orden los documentos, una vez impresos ya se eliminaban de la cola de impresión, a qué estructura de datos se parecía el funcionamiento de esa impresora era la pregunta**

La estructura de datos se asemeja a una cola circular. Ya que los elementos van ingresando y se van imprimiendo según el orden de ingreso de cada uno., el primero en ingresar es el primero que sale impreso. El acceso es destructivo, a medida que se imprimen se eliminan de la estructura.

Es una cola circular y no una cola, ya que esta tiene un tamaño máximo permitido. Si la impresora no tuviera un tamaño máximo implementaríamos una cola.

2. **se quería llevar un grafo a una estructura estática, (una matriz de adyacencia) y tenías que decir cómo sería el alta y la baja de una conexión**

se crea una matriz de Anxn, siendo n la cantidad de nodos. Cada nodo aij puede tomar el valor de 0 si no hay arcos que unan estos nodos, o 1 si hay un arco que los une

Para añadir una nueva conexión entre i y j, basta con situarse sobre el elemento a_{ij} (i columna, j fila) y colocar un 1. Para dar de baja una conexión, colocamos un 0

Las columnas son los **destinos**. Para obtener el conjunto Left de un nodo, leemos la columna del mismo de arriba hacia abajo.

Las filas son lo **orígenes**. Para obtener el conjunto Right de un nodo, leemos la fila del mismo de izquierda a derecha.