Intro a las Estructuras de Datos

Miguel Raggi

Redes
Escuela Nacional de Estudios Superiores
UNAM

31 de enero de 2018

Índice:

Índice:

Abstracto vs Concreto

- Hay dos puntos de vista en las estructuras de datos:

Abstracto vs Concreto

- Hay dos puntos de vista en las estructuras de datos:
- El abstracto: Es una descripción de qué quieres que haga tu estructura de datos, cómo quieres que se comporte, etc.

Abstracto vs Concreto

- Hay dos puntos de vista en las estructuras de datos:
- El abstracto: Es una descripción de qué quieres que haga tu estructura de datos, cómo quieres que se comporte, etc.
- El concreto: Es una descripción de cómo estará metido en la memoria de una computadora todo.

Abstracto vs Concreto

- Hay dos puntos de vista en las estructuras de datos:
- El abstracto: Es una descripción de qué quieres que haga tu estructura de datos, cómo quieres que se comporte, etc.
- El concreto: Es una descripción de cómo estará metido en la memoria de una computadora todo.
- Por ejemplo, es común considerar una cola de prioridad implementado como una heap.

Abstracto vs Concreto

- Hay dos puntos de vista en las estructuras de datos:
- El abstracto: Es una descripción de qué quieres que haga tu estructura de datos, cómo quieres que se comporte, etc.
- El concreto: Es una descripción de cómo estará metido en la memoria de una computadora todo.
- Por ejemplo, es común considerar una cola de prioridad implementado como una heap.
- La cola de prioridad es una descripción de cómo quieres que se comporte tu estructura.

Abstracto vs Concreto

- Hay dos puntos de vista en las estructuras de datos:
- El abstracto: Es una descripción de qué quieres que haga tu estructura de datos, cómo quieres que se comporte, etc.
- El concreto: Es una descripción de cómo estará metido en la memoria de una computadora todo.
- Por ejemplo, es común considerar una cola de prioridad implementado como una heap.
- La cola de prioridad es una descripción de cómo quieres que se comporte tu estructura.
- La heap es una descripción de cómo será implementada dicha estructura.

Abstracto vs Concreto

- Hay dos puntos de vista en las estructuras de datos:
- El abstracto: Es una descripción de qué quieres que haga tu estructura de datos, cómo quieres que se comporte, etc.
- El concreto: Es una descripción de cómo estará metido en la memoria de una computadora todo.
- Por ejemplo, es común considerar una cola de prioridad implementado como una heap.
- La cola de prioridad es una descripción de cómo quieres que se comporte tu estructura.
- La heap es una descripción de cómo será implementada dicha estructura.
- Sin embargo, hay otras maneras de implementar una cola de prioridad, como fibonacci heap, o arreglos, o etc.

Abstracto vs Concreto

- Hay dos puntos de vista en las estructuras de datos:
- El abstracto: Es una descripción de qué quieres que haga tu estructura de datos, cómo quieres que se comporte, etc.
- El concreto: Es una descripción de cómo estará metido en la memoria de una computadora todo.
- Por ejemplo, es común considerar una cola de prioridad implementado como una heap.
- La cola de prioridad es una descripción de cómo quieres que se comporte tu estructura.
- La heap es una descripción de cómo será implementada dicha estructura.
- Sin embargo, hay otras maneras de implementar una cola de prioridad, como fibonacci heap, o arreglos, o etc.
- Algunas son mejores y otras peores, dependiendo del problema particular.

Operaciones

- ¿Qué tipo de cosas podríamos querer hacer con nuestra estructura de datos?

Operaciones

- ¿Qué tipo de cosas podríamos querer hacer con nuestra estructura de datos?
  - Ver los elementos

Operaciones

- ¿Qué tipo de cosas podríamos querer hacer con nuestra estructura de datos?
  - Ver los elementos
  - Insertar un elemento en alguna posición

Operaciones

- ¿Qué tipo de cosas podríamos querer hacer con nuestra estructura de datos?
  - Ver los elementos
  - Insertar un elemento en alguna posición
  - Borrar un elemento de alguna posición

Operaciones

- ¿Qué tipo de cosas podríamos querer hacer con nuestra estructura de datos?
  - Ver los elementos
  - Insertar un elemento en alguna posición
  - Borrar un elemento de alguna posición
  - Encontrar un elemento dado

Operaciones

- ¿Qué tipo de cosas podríamos querer hacer con nuestra estructura de datos?
  - Ver los elementos
  - Insertar un elemento en alguna posición
  - Borrar un elemento de alguna posición
  - Encontrar un elemento dado
  - Ordenar los elementos

Operaciones

- ¿Qué tipo de cosas podríamos querer hacer con nuestra estructura de datos?
  - Ver los elementos
  - Insertar un elemento en alguna posición
  - Borrar un elemento de alguna posición
  - Encontrar un elemento dado
  - Ordenar los elementos
- Cada estructura de datos tiene ventajas y desventajas para hacer las operaciones anteriores.

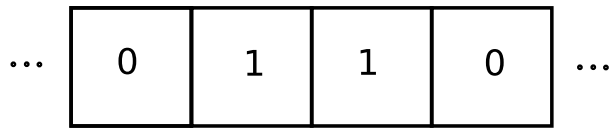
Operaciones

- ¿Qué tipo de cosas podríamos querer hacer con nuestra estructura de datos?
  - Ver los elementos
  - Insertar un elemento en alguna posición
  - Borrar un elemento de alguna posición
  - Encontrar un elemento dado
  - Ordenar los elementos
- Cada estructura de datos tiene ventajas y desventajas para hacer las operaciones anteriores.
- Quizás algunas estructuras son más rápidas para hacer algo y otras más rápidas para hacer otra cosa.

Índice:

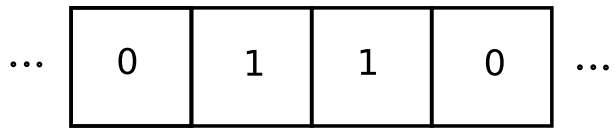
Memoria

- La memoria en la computadora se puede ver como una muy larga banda de 0's y 1's



Memoria

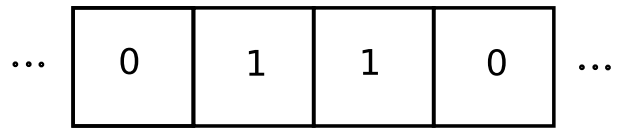
- La memoria en la computadora se puede ver como una muy larga banda de 0's y 1's



- Podemos pensar que las cajitas tienen cosas mas grandes, como números o bytes o lo que sea.

Memoria

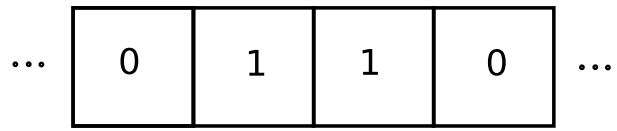
- La memoria en la computadora se puede ver como una muy larga banda de 0's y 1's



- Podemos pensar que las cajitas tienen cosas mas grandes, como números o bytes o lo que sea.
- ¿Cómo guardamos una lista de cosas en la computadora?

Memoria

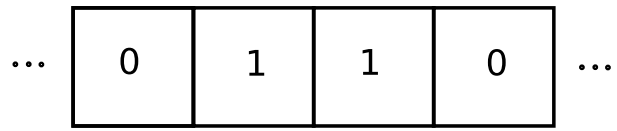
- La memoria en la computadora se puede ver como una muy larga banda de 0's y 1's



- Podemos pensar que las cajitas tienen cosas mas grandes, como números o bytes o lo que sea.
- ¿Cómo guardamos una lista de cosas en la computadora?
- Hay dos tipos de estructuras de datos concretas "básicas", aunque se pueden combinar para hacer muchas, muchas más:

Memoria

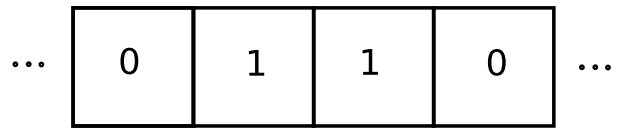
- La memoria en la computadora se puede ver como una muy larga banda de 0's y 1's



- Podemos pensar que las cajitas tienen cosas mas grandes, como números o bytes o lo que sea.
- ¿Cómo guardamos una lista de cosas en la computadora?
- Hay dos tipos de estructuras de datos concretas "básicas", aunque se pueden combinar para hacer muchas, muchas más:
- Podemos utilizar mezclas de arreglos y apuntadores.

Memoria

- La memoria en la computadora se puede ver como una muy larga banda de 0's y 1's



- Podemos pensar que las cajitas tienen cosas mas grandes, como números o bytes o lo que sea.
- ¿Cómo guardamos una lista de cosas en la computadora?
- Hay dos tipos de estructuras de datos concretas "básicas", aunque se pueden combinar para hacer muchas, muchas más:
- Podemos utilizar mezclas de arreglos y apuntadores.
- Podría dar un curso entero de estructuras de datos, pero ahora solo daré una pequeñísima introducción.

Arreglos

- La cosa más básica que podemos hacer es guardar los datos en memoria contigua, es decir, una cosa tras otra. Cuando termina una, podemos poner la siguiente y luego la siguiente y etc.

Arreglos

- La cosa más básica que podemos hacer es guardar los datos en memoria contigua, es decir, una cosa tras otra. Cuando termina una, podemos poner la siguiente y luego la siguiente y etc.
- Hacer esto se le llama poner los objetos en un arreglo.

Arreglos

- La cosa más básica que podemos hacer es guardar los datos en memoria contigua, es decir, una cosa tras otra. Cuando termina una, podemos poner la siguiente y luego la siguiente y etc.
- Hacer esto se le llama poner los objetos en un arreglo.
- El problema que tiene esto es que si tienes una lista larga, insertar algo o borrar algo de la mitad significa que hay que mover todo lo que viene después!

Arreglos

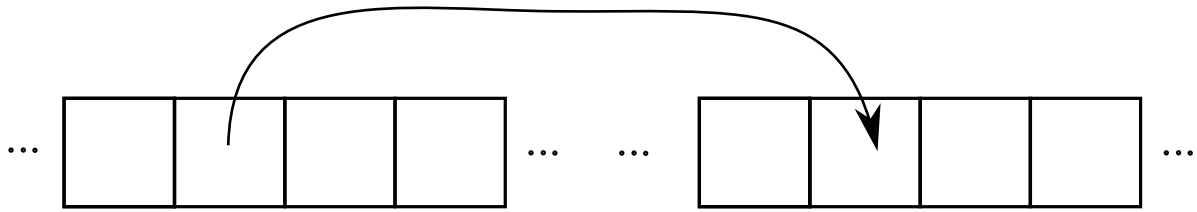
- La cosa más básica que podemos hacer es guardar los datos en memoria contigua, es decir, una cosa tras otra. Cuando termina una, podemos poner la siguiente y luego la siguiente y etc.
- Hacer esto se le llama poner los objetos en un arreglo.
- El problema que tiene esto es que si tienes una lista larga, insertar algo o borrar algo de la mitad significa que hay que mover todo lo que viene después!
- La ventaja principal de esto es que si todos los objetos que guardamos son del mismo tamaño (por ejemplo, si todos fueran enteros), si queremos ver el elemento número 1000 es muy fácil, sólo saltamos 1000 "espacios" hacia la derecha y vemos donde está.

Arreglos

- La cosa más básica que podemos hacer es guardar los datos en memoria contigua, es decir, una cosa tras otra. Cuando termina una, podemos poner la siguiente y luego la siguiente y etc.
- Hacer esto se le llama poner los objetos en un arreglo.
- El problema que tiene esto es que si tienes una lista larga, insertar algo o borrar algo de la mitad significa que hay que mover todo lo que viene después!
- La ventaja principal de esto es que si todos los objetos que guardamos son del mismo tamaño (por ejemplo, si todos fueran enteros), si queremos ver el elemento número 1000 es muy fácil, sólo saltamos 1000 "espacios" hacia la derecha y vemos donde está.
- Si no fueran del mismo tamaño, (por ejemplo que hubiera matrices y enteros mezclados), podemos poner en cada uno de los cuadraditos un apuntador a algún lugar de la memoria!

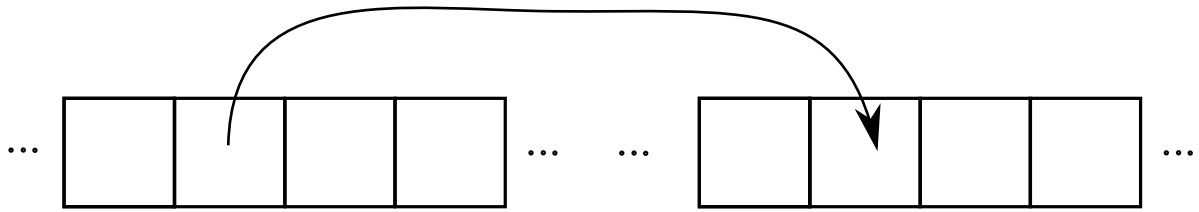
Apuntadores

- En un cuadrito, podemos poner una "flecha" que va a otro lugar diferente de la memoria. Eso se llama un apuntador.



Apuntadores

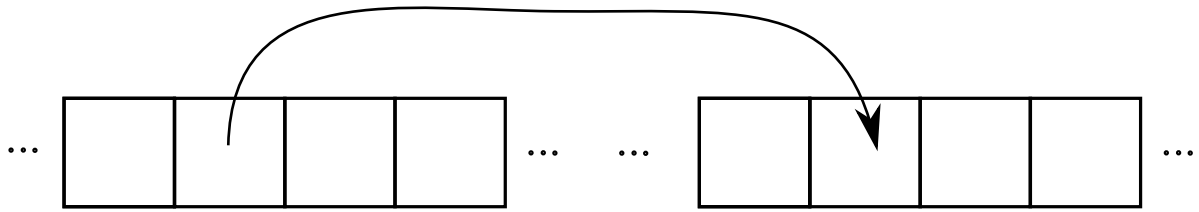
- En un cuadrito, podemos poner una "flecha" que va a otro lugar diferente de la memoria. Eso se llama un apuntador.



- Saltar de un lugar de la memoria a otro que conozcamos se hace en un solo paso. No hay que pasar por todos los del medio.

Apuntadores

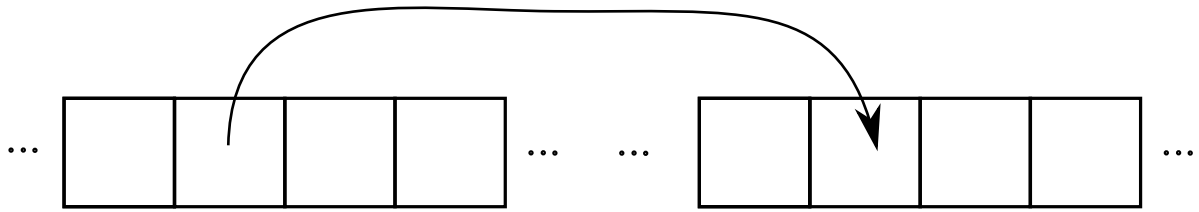
- En un cuadrito, podemos poner una "flecha" que va a otro lugar diferente de la memoria. Eso se llama un apuntador.



- Saltar de un lugar de la memoria a otro que conozcamos se hace en un solo paso. No hay que pasar por todos los del medio.
- Para guardar un conjunto de cosas en la computadora, podríamos hacer que cada cosa apunte a la siguiente (y quizás a la anterior) de la lista.

Apuntadores

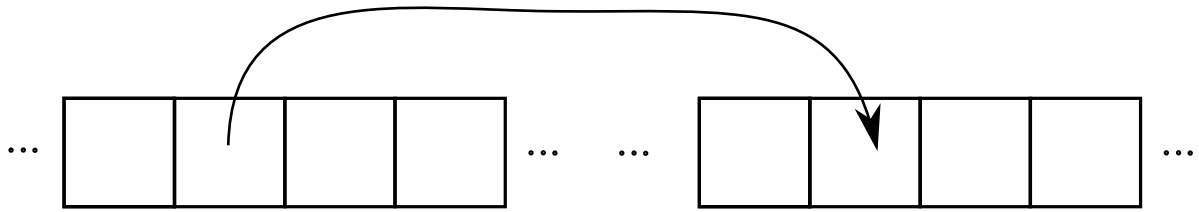
- En un cuadrito, podemos poner una "flecha" que va a otro lugar diferente de la memoria. Eso se llama un apuntador.



- Saltar de un lugar de la memoria a otro que conozcamos se hace en un solo paso. No hay que pasar por todos los del medio.
- Para guardar un conjunto de cosas en la computadora, podríamos hacer que cada cosa apunte a la siguiente (y quizás a la anterior) de la lista.
- El problema principal es que si queremos ver al elemento 1000, tenemos que pasar por todos los del medio.

Apuntadores

- En un cuadrito, podemos poner una "flecha" que va a otro lugar diferente de la memoria. Eso se llama un apuntador.



- Saltar de un lugar de la memoria a otro que conozcamos se hace en un solo paso. No hay que pasar por todos los del medio.
- Para guardar un conjunto de cosas en la computadora, podríamos hacer que cada cosa apunte a la siguiente (y quizás a la anterior) de la lista.
- El problema principal es que si queremos ver al elemento 1000, tenemos que pasar por todos los del medio.
- Para insertar elementos en un lugar ya dado es muy fácil, sólo hay que cambiar dos cosas.

Índice:

Estructuras de datos abstractas

- Vamos a ver algunas estructuras de datos usuales.

Estructuras de datos abstractas

- Vamos a ver algunas estructuras de datos usuales.
- Cuando hablamos de estructuras de datos “abstractas” estamos pensando, no en cómo guardar la información, sino en como queremos poder utilizar la información.

Estructuras de datos abstractas

- Vamos a ver algunas estructuras de datos usuales.
- Cuando hablamos de estructuras de datos "abstractas" estamos pensando, no en cómo guardar la información, sino en como queremos poder utilizar la información.
- Hay muchas otras, claro, y a lo largo del curso estaremos viendo algunas.

Pila

Definición

Una pila es una estructura de datos que permite:

Pila

Definición

Una pila es una estructura de datos que permite:

- *Insertar elementos al final.*

Pila

Definición

Una pila es una estructura de datos que permite:

- *Insertar elementos al final.*
- *Sacar el último elemento insertado.*

Pila

Definición

Una pila es una estructura de datos que permite:

- *Insertar elementos al final.*
- *Sacar el último elemento insertado.*

Es todo. Te lo debes imaginar como una pila de piedritas en donde vas poniendo piedritas hasta arriba y sacando de ahí.

Pila

Definición

Una pila es una estructura de datos que permite:

- *Insertar elementos al final.*
- *Sacar el último elemento insertado.*

Es todo. Te lo debes imaginar como una pila de piedritas en donde vas poniendo piedritas hasta arriba y sacando de ahí.

- Usualmente se implementa como una lista ligada o como un arreglo.

Pila

Definición

Una pila es una estructura de datos que permite:

- *Insertar elementos al final.*
- *Sacar el último elemento insertado.*

Es todo. Te lo debes imaginar como una pila de piedritas en donde vas poniendo piedritas hasta arriba y sacando de ahí.

- Usualmente se implementa como una lista ligada o como un arreglo.
- Se utiliza en las computadoras por ejemplo para la "pila de llamadas" (call stack), que es la manera de cómo llama funciones la computadora.

Cola

Cola

Una cola te permite:

Cola

Una cola te permite:

- Insertar elementos

Cola

Una cola te permite:

- Insertar elementos
- Sacar el primer elemento que se insertó.

Cola

Una cola te permite:

- Insertar elementos
- Sacar el primer elemento que se insertó.

Te lo imaginas como una cola de personas, en donde la primera que llegó es la primera que atienden, y los que llegan se ponen hasta atrás.

Cola

Una cola te permite:

- Insertar elementos
- Sacar el primer elemento que se insertó.

Te lo imaginas como una cola de personas, en donde la primera que llegó es la primera que atienden, y los que llegan se ponen hasta atrás.

- Usualmente es un arreglo dinámico.

Cola

Una cola te permite:

- Insertar elementos
- Sacar el primer elemento que se insertó.

Te lo imaginas como una cola de personas, en donde la primera que llegó es la primera que atienden, y los que llegan se ponen hasta atrás.

- Usualmente es un arreglo dinámico.
- Sirve para varias cosas.

Doble Cola

Doble Cola

- Es una cola y una pila a la vez:

Doble Cola

- Es una cola y una pila a la vez:
- Puedes sacar y meter elementos por ambos lados.

Doble Cola

- Es una cola y una pila a la vez:
- Puedes sacar y meter elementos por ambos lados.

Cola de Prioridad

Cola de Prioridad

- Se vale insertar elementos.

Cola de Prioridad

- Se vale insertar elementos.
- Se vale sacar (o mirar) el elemento “más grande”.

Cola de Prioridad

- Se vale insertar elementos.
- Se vale sacar (o mirar) el elemento "más grande".
- Usualmente se implementa como una heap (que a su vez es un arreglo) o una fibonacci heap, o algún otro tipo de "árbol".

Conjunto

Conjunto

- Permite insertar, remover e iterar sobre los elementos.

Conjunto

- Permite insertar, remover e iterar sobre los elementos.
- El chiste es que no permite que haya elementos repetidos.

Conjunto

- Permite insertar, remover e iterar sobre los elementos.
- El chiste es que no permite que haya elementos repetidos.
- Usualmente para hacer esto rápido, se mantienen en algún orden determinado

Índice:

Suena muy complicado... ¿qué uso?

Suena muy complicado... ¿qué uso?

- En teoría, diferentes estructuras de datos sirven para diferentes cosas.

Suena muy complicado... ¿qué uso?

- En teoría, diferentes estructuras de datos sirven para diferentes cosas.
- En la práctica, sin embargo, eso es falso. Siempre debes usar un arreglo (o "vector") (bueno, o arreglo de arreglos, etc.)

Suena muy complicado... ¿qué uso?

- En teoría, diferentes estructuras de datos sirven para diferentes cosas.
- En la práctica, sin embargo, eso es falso. Siempre debes usar un arreglo (o "vector") (bueno, o arreglo de arreglos, etc.)
- No le digan a nadie que les dije esto.

Suena muy complicado... ¿qué uso?

- En teoría, diferentes estructuras de datos sirven para diferentes cosas.
- En la práctica, sin embargo, eso es falso. Siempre debes usar un arreglo (o "vector") (bueno, o arreglo de arreglos, etc.)
- No le digan a nadie que les dije esto.
- ¿Por qué?

Suena muy complicado... ¿qué uso?

- En teoría, diferentes estructuras de datos sirven para diferentes cosas.
- En la práctica, sin embargo, eso es falso. Siempre debes usar un arreglo (o "vector") (bueno, o arreglo de arreglos, etc.)
- No le digan a nadie que les dije esto.
- ¿Por qué? La manera en que están construidas las computadoras hace que utilizar apuntadores sea lento.

Suena muy complicado... ¿qué uso?

- En teoría, diferentes estructuras de datos sirven para diferentes cosas.
- En la práctica, sin embargo, eso es falso. Siempre debes usar un arreglo (o "vector") (bueno, o arreglo de arreglos, etc.)
- No le digan a nadie que les dije esto.
- ¿Por qué? La manera en que están construidas las computadoras hace que utilizar apuntadores sea lento.
- Entonces, a menos que tengas una muuuuuy buena razón para no hacerlo, simplemente utiliza un arreglo dinámico (std::vector en C++, "lista" en python, etc.)

Índice:

Estructura de grafo

Estructura de grafo

- P: ¿Cómo guardamos una gráfica en la computadora?

Estructura de grafo

- P: ¿Cómo guardamos una gráfica en la computadora?
- R: Depende de qué queremos hacer con ella.

Estructura de grafo

- P: ¿Cómo guardamos una gráfica en la computadora?
- R: Depende de qué queremos hacer con ella.
- Hay tres principales maneras, con sus ventajas y desventajas:

Estructura de grafo

- P: ¿Cómo guardamos una gráfica en la computadora?
- R: Depende de qué queremos hacer con ella.
- Hay tres principales maneras, con sus ventajas y desventajas:
  1. Guardo una lista de aristas

Estructura de grafo

- P: ¿Cómo guardamos una gráfica en la computadora?
- R: Depende de qué queremos hacer con ella.
- Hay tres principales maneras, con sus ventajas y desventajas:
  1. Guardo una lista de aristas
  2. Matriz de adyacencia.

Estructura de grafo

- P: ¿Cómo guardamos una gráfica en la computadora?
- R: Depende de qué queremos hacer con ella.
- Hay tres principales maneras, con sus ventajas y desventajas:
  1. Guardo una lista de aristas
  2. Matriz de adyacencia.
  3. Listas de adyacencia (a cada nodo le guardo sus vecinos).

Estructura de grafo

- P: ¿Cómo guardamos una gráfica en la computadora?
- R: Depende de qué queremos hacer con ella.
- Hay tres principales maneras, con sus ventajas y desventajas:
  1. Guardo una lista de aristas
  2. Matriz de adyacencia.
  3. Listas de adyacencia (a cada nodo le guardo sus vecinos).
- Es bueno que los nodos estén numerados del 0 al $n-1$ para identificarlos. Si no está así, puedes asociar (por medio de "hashing") a cada nodo un número.

Lista de aristas

Ventajas de la lista de aristas:

- Añadir nodos y aristas es trivial.
- Recorrer todas las aristas es muy rápido.

Desventajas:

Lista de aristas

Ventajas de la lista de aristas:

- Añadir nodos y aristas es trivial.
- Recorrer todas las aristas es muy rápido.

Desventajas:

- Saber quiénes son los vecinos de un nodo particular es extremadamente lento. Hay que recorrer toooda la lista de aristas cada vez.

Lista de aristas

Ventajas de la lista de aristas:

- Añadir nodos y aristas es trivial.
- Recorrer todas las aristas es muy rápido.

Desventajas:

- Saber quiénes son los vecinos de un nodo particular es extremadamente lento. Hay que recorrer toooda la lista de aristas cada vez.
- Quitar aristas o vértices es muy lento.

Matriz de Adyacencia

Detalles: Usualmente un arreglo de arreglos (del mismo tamaño).

Matriz de Adyacencia

Detalles: Usualmente un arreglo de arreglos (del mismo tamaño).

Ventajas de la matriz:

- Dados dos nodos, es muy fácil y rápido saber si hay arista entre ellos.

Matriz de Adyacencia

Detalles: Usualmente un arreglo de arreglos (del mismo tamaño).

Ventajas de la matriz:

- Dados dos nodos, es muy fácil y rápido saber si hay arista entre ellos.
- Es fácil hacer operaciones de matrices ahí (después veremos que a veces es útil).

Matriz de Adyacencia

Detalles: Usualmente un arreglo de arreglos (del mismo tamaño).

Ventajas de la matriz:

- Dados dos nodos, es muy fácil y rápido saber si hay arista entre ellos.
- Es fácil hacer operaciones de matrices ahí (después veremos que a veces es útil).
- Agregar o quitar aristas es muy rápido.

Matriz de Adyacencia

Detalles: Usualmente un arreglo de arreglos (del mismo tamaño).

Ventajas de la matriz:

- Dados dos nodos, es muy fácil y rápido saber si hay arista entre ellos.
- Es fácil hacer operaciones de matrices ahí (después veremos que a veces es útil).
- Agregar o quitar aristas es muy rápido.
- A las computadoras les gusta trabajar en arreglos de números.

Matriz de Adyacencia

Detalles: Usualmente un arreglo de arreglos (del mismo tamaño).

Ventajas de la matriz:

- Dados dos nodos, es muy fácil y rápido saber si hay arista entre ellos.
- Es fácil hacer operaciones de matrices ahí (después veremos que a veces es útil).
- Agregar o quitar aristas es muy rápido.
- A las computadoras les gusta trabajar en arreglos de números.

Desventajas:

Matriz de Adyacencia

Detalles: Usualmente un arreglo de arreglos (del mismo tamaño).

Ventajas de la matriz:

- Dados dos nodos, es muy fácil y rápido saber si hay arista entre ellos.
- Es fácil hacer operaciones de matrices ahí (después veremos que a veces es útil).
- Agregar o quitar aristas es muy rápido.
- A las computadoras les gusta trabajar en arreglos de números.

Desventajas:

- Saber quiénes son los vecinos de un nodo particular es lento. Hay que recorrer tooodo el renglón buscando 1's.

Matriz de Adyacencia

Detalles: Usualmente un arreglo de arreglos (del mismo tamaño).

Ventajas de la matriz:

- Dados dos nodos, es muy fácil y rápido saber si hay arista entre ellos.
- Es fácil hacer operaciones de matrices ahí (después veremos que a veces es útil).
- Agregar o quitar aristas es muy rápido.
- A las computadoras les gusta trabajar en arreglos de números.

Desventajas:

- Saber quiénes son los vecinos de un nodo particular es lento. Hay que recorrer tooodo el renglón buscando 1's.
- Insertar o quitar nodos es muy lento.

Matriz de Adyacencia

Detalles: Usualmente un arreglo de arreglos (del mismo tamaño).

Ventajas de la matriz:

- Dados dos nodos, es muy fácil y rápido saber si hay arista entre ellos.
- Es fácil hacer operaciones de matrices ahí (después veremos que a veces es útil).
- Agregar o quitar aristas es muy rápido.
- A las computadoras les gusta trabajar en arreglos de números.

Desventajas:

- Saber quiénes son los vecinos de un nodo particular es lento. Hay que recorrer tooodo el renglón buscando 1's.
- Insertar o quitar nodos es muy lento.
- Utilizas mucha memoria. Si por ejemplo hay pocas aristas con respecto al número de vértices, estarás guardando una matriz enorme llena de 0's.

Listas de Adyacencia

Detalles: Usualmente un arreglo de arreglos, donde el arreglo en la posición i es la lista de vecinos del vértice i. Además,

Listas de Adyacencia

Detalles: Usualmente un arreglo de arreglos, donde el arreglo en la posición i es la lista de vecinos del vértice i. Además,

Ventajas de la lista de adyacencia:

Listas de Adyacencia

Detalles: Usualmente un arreglo de arreglos, donde el arreglo en la posición i es la lista de vecinos del vértice i. Además,

Ventajas de la lista de adyacencia:

- Saber quiénes son los vecinos de un nodo particular es muy rápido.

Listas de Adyacencia

Detalles: Usualmente un arreglo de arreglos, donde el arreglo en la posición i es la lista de vecinos del vértice i. Además,

Ventajas de la lista de adyacencia:

- Saber quiénes son los vecinos de un nodo particular es muy rápido.
- Agregar vértices es (comparativamente) rápido.

Desventajas:

Listas de Adyacencia

Detalles: Usualmente un arreglo de arreglos, donde el arreglo en la posición i es la lista de vecinos del vértice i. Además,

Ventajas de la lista de adyacencia:

- Saber quiénes son los vecinos de un nodo particular es muy rápido.
- Agregar vértices es (comparativamente) rápido.

Desventajas:

- Agregar o quitar aristas es (ligeramente) más lento.

Listas de Adyacencia

Detalles: Usualmente un arreglo de arreglos, donde el arreglo en la posición i es la lista de vecinos del vértice i. Además,

Ventajas de la lista de adyacencia:

- Saber quiénes son los vecinos de un nodo particular es muy rápido.
- Agregar vértices es (comparativamente) rápido.

Desventajas:

- Agregar o quitar aristas es (ligeramente) más lento.
- Saber si dos nodos dados son vecinos es lento.

Lista de Adyacencia ordenada

- Para acelerar ciertas operaciones (pero alentar otras) podemos insistir en que los vecinos estén ordenados (con cualquier orden fijo).

Lista de Adyacencia ordenada

- Para acelerar ciertas operaciones (pero alentar otras) podemos insistir en que los vecinos estén ordenados (con cualquier orden fijo).
- Eso hace que agregar aristas o vértices sea más lento.

Lista de Adyacencia ordenada

- Para acelerar ciertas operaciones (pero alentar otras) podemos insistir en que los vecinos estén ordenados (con cualquier orden fijo).
- Eso hace que agregar aristas o vértices sea más lento.
- Sin embargo, buscar si dos nodos particulares son vecinos será más rápido.

¿Cuál utilizo?

- En la mayoría de las aplicaciones, tienes un grafo fijo y trabajas en él.

¿Cuál utilizo?

- En la mayoría de las aplicaciones, tienes un grafo fijo y trabajas en él.
- Entonces, si tienes suficiente memoria, podría valer la pena utilizar ambas representaciones, para obtener todas las ventajas de ambas.

¿Cuál utilizo?

- En la mayoría de las aplicaciones, tienes un grafo fijo y trabajas en él.
- Entonces, si tienes suficiente memoria, podría valer la pena utilizar ambas representaciones, para obtener todas las ventajas de ambas.
- Si vas a modificar el grafo seguido, entonces depende.

Gráficas al vuelo

- A veces no guardamos toda la gráfica en la memoria.

Gráficas al vuelo

- A veces no guardamos toda la gráfica en la memoria.
- A veces más bien tenemos una función `Vecinos(Nodo n)`, que nos regresa un vector con una lista (vector) de nodos que son sus vecinos. Posiblemente con sus pesos.

Gráficas al vuelo

- A veces no guardamos toda la gráfica en la memoria.
- A veces más bien tenemos una función `Vecinos(Nodo n)`, que nos regresa un vector con una lista (vector) de nodos que son sus vecinos. Posiblemente con sus pesos.
- Esto, para el análisis de eficiencia de algoritmos, es equivalente a tener las listas de adyacencia, salvo que quizás "encontrar los vecinos" podría ser una operación no trivial.