Búsqueda Binaria

Miguel Raggi

Algoritmos
Escuela Nacional de Estudios Superiores
UNAM

26 de febrero de 2019

Índice:

Índice:

¿Qué es una estructura de datos?

¿Qué es una estructura de datos?

- Es la manera de guardar y organizar datos en una computadora.

¿Qué es una estructura de datos?

- Es la manera de guardar y organizar datos en una computadora.
- Abstractamente, lo podemos pensar como una lista o un conjunto de cosas en la computadora.

¿Qué es una estructura de datos?

- Es la manera de guardar y organizar datos en una computadora.
- Abstractamente, lo podemos pensar como una lista o un conjunto de cosas en la computadora.
- Para entender qué son, hay dos maneras que debemos entender primero:

¿Qué es una estructura de datos?

- Es la manera de guardar y organizar datos en una computadora.
- Abstractamente, lo podemos pensar como una lista o un conjunto de cosas en la computadora.
- Para entender qué son, hay dos maneras que debemos entender primero:
- ¿Cómo guardamos datos en una computadora?

¿Qué es una estructura de datos?

- Es la manera de guardar y organizar datos en una computadora.
- Abstractamente, lo podemos pensar como una lista o un conjunto de cosas en la computadora.
- Para entender qué son, hay dos maneras que debemos entender primero:
- ¿Cómo guardamos datos en una computadora?
- ¿Cómo usamos/encontramos datos en una computadora?

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

Operaciones

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

Operaciones

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

Operaciones

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

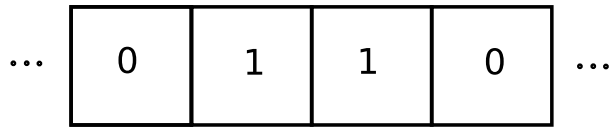
Operaciones

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

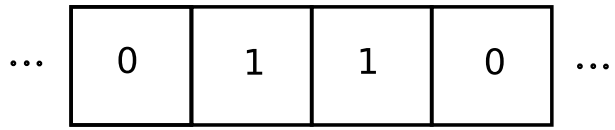
Memoria

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



Memoria

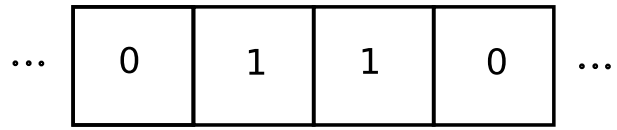
- La memoria en la computadora se puede ver como una muy larga banda de 0's y 1's



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

Memoria

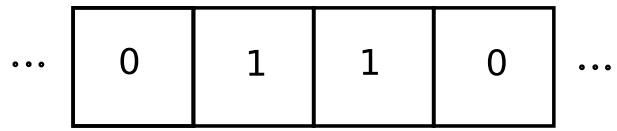
- La memoria en la computadora se puede ver como una muy larga banda de 0's y 1's



- 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

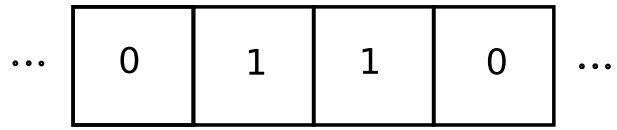
- La memoria en la computadora se puede ver como una muy larga banda de 0's y 1's



- 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

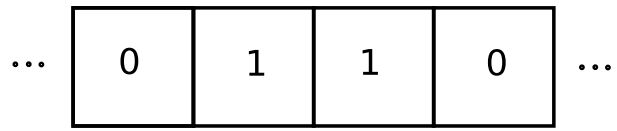
- La memoria en la computadora se puede ver como una muy larga banda de 0's y 1's



- 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

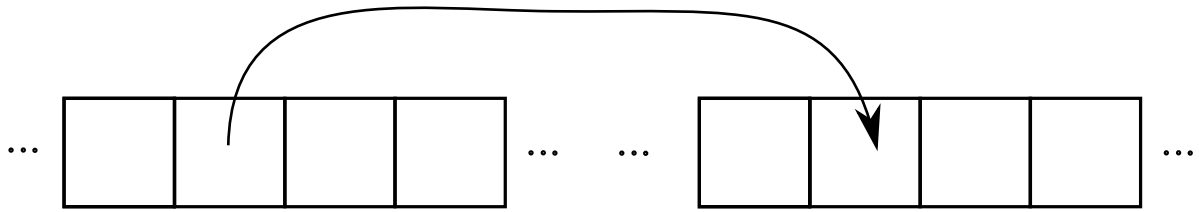
- La cosa más básica que podemos hacer es guardar los datos en memoria contigua, es decir, una cosa tras otra. Cuando termina una, podemos poner la siguiente y luego la siguiente y etc.
- Hacer esto se le llama poner los objetos en un arreglo.
- El problema que tiene esto es que si tienes una lista larga,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

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!
- Las "listas" de python son casi arreglos, por eso es que añadir y quitar cosas del final es rápido, pero hacerlo en el medio es lento.

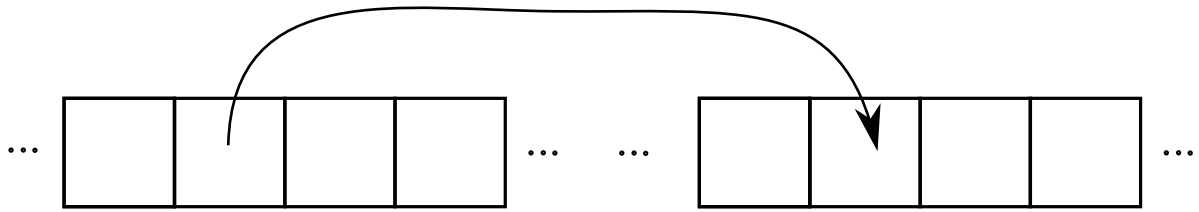
Apuntadores

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



Apuntadores

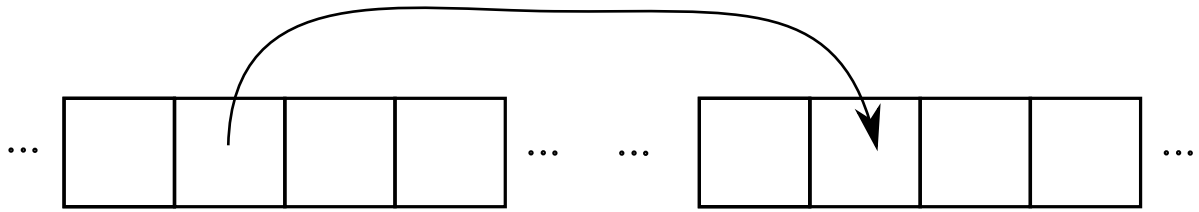
- En un cuadrito, podemos poner una "flecha" que va a otro lugar diferente de la memoria. Eso se llama un apuntador.



- 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

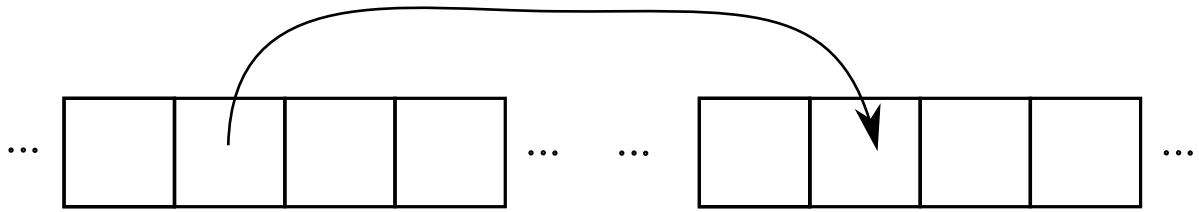
- En un cuadrito, podemos poner una "flecha" que va a otro lugar diferente de la memoria. Eso se llama un apuntador.



- 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

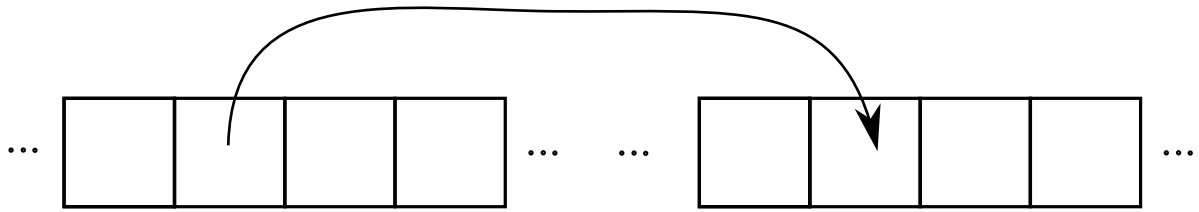
- En un cuadrito, podemos poner una "flecha" que va a otro lugar diferente de la memoria. Eso se llama un apuntador.



- 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.

C++ vector y list

- Veamos cómo usar C++ array, vector y list.

Índice:

Búsqueda

- Uno de los problemas más importantes que hacemos que la computadora resuelva por nosotros es del de buscar.

Búsqueda

- Uno de los problemas más importantes que hacemos que la computadora resuelva por nosotros es del de buscar.
- ¿Qué es buscar?

Búsqueda

- Uno de los problemas más importantes que hacemos que la computadora resuelva por nosotros es del de buscar.
- ¿Qué es buscar?
- Por ejemplo, si tenemos una lista de todos los empleados de una compañía, queremos encontrar a un empleado en particular, usando su número de trabajador, o su apellido, o etc.

Búsqueda

- Uno de los problemas más importantes que hacemos que la computadora resuelva por nosotros es del de buscar.
- ¿Qué es buscar?
- Por ejemplo, si tenemos una lista de todos los empleados de una compañía, queremos encontrar a un empleado en particular, usando su número de trabajador, o su apellido, o etc.
- Importante: ¿Qué estamos suponiendo que la computadora puede hacer para buscar?

Búsqueda

- Uno de los problemas más importantes que hacemos que la computadora resuelva por nosotros es del de buscar.
- ¿Qué es buscar?
- Por ejemplo, si tenemos una lista de todos los empleados de una compañía, queremos encontrar a un empleado en particular, usando su número de trabajador, o su apellido, o etc.
- Importante: ¿Qué estamos suponiendo que la computadora puede hacer para buscar?
- Dependiendo de lo que supongamos que puede hacer, la respuesta cambia.

Búsqueda

- Uno de los problemas más importantes que hacemos que la computadora resuelva por nosotros es del de buscar.
- ¿Qué es buscar?
- Por ejemplo, si tenemos una lista de todos los empleados de una compañía, queremos encontrar a un empleado en particular, usando su número de trabajador, o su apellido, o etc.
- Importante: ¿Qué estamos suponiendo que la computadora puede hacer para buscar?
- Dependiendo de lo que supongamos que puede hacer, la respuesta cambia.
- Claro, dependiendo de la situación en particular, podemos hacer diferentes cosas.

Búsqueda

- Uno de los problemas más importantes que hacemos que la computadora resuelva por nosotros es del de buscar.
- ¿Qué es buscar?
- Por ejemplo, si tenemos una lista de todos los empleados de una compañía, queremos encontrar a un empleado en particular, usando su número de trabajador, o su apellido, o etc.
- Importante: ¿Qué estamos suponiendo que la computadora puede hacer para buscar?
- Dependiendo de lo que supongamos que puede hacer, la respuesta cambia.
- Claro, dependiendo de la situación en particular, podemos hacer diferentes cosas.
- Vamos a ver qué tipo de información sería útil tener antes de empezar a buscar. Pensemos por un momento que no tenemos computadora y tenemos que hacer todo a mano.

Primer Algoritmo Chafa

- Por ejemplo, supongamos que nos dan una lista gigante de empleados, todos en desorden, y nos piden que encontremos al empleado “Crescencio Creciente”.

Primer Algoritmo Chafa

- Por ejemplo, supongamos que nos dan una lista gigante de empleados, todos en desorden, y nos piden que encontremos al empleado “Crescencio Creciente”.
- En realidad no podemos hacer otra cosa mas que agarrar al primer empleado, ver si es Crescencio, y si no, seguir con el segundo, etc. hasta encontrarlo.

Primer Algoritmo Chafa

- Por ejemplo, supongamos que nos dan una lista gigante de empleados, todos en desorden, y nos piden que encontremos al empleado “Crescencio Creciente”.
- En realidad no podemos hacer otra cosa mas que agarrar al primer empleado, ver si es Crescencio, y si no, seguir con el segundo, etc. hasta encontrarlo.
- ¿Qué ventajas y desventajas tiene este algoritmo?

Primer Algoritmo Chafa

- Por ejemplo, supongamos que nos dan una lista gigante de empleados, todos en desorden, y nos piden que encontremos al empleado "Crescencio Creciente".
- En realidad no podemos hacer otra cosa mas que agarrar al primer empleado, ver si es Crescencio, y si no, seguir con el segundo, etc. hasta encontrarlo.
- ¿Qué ventajas y desventajas tiene este algoritmo?
  - Ventaja: Es muy fácil de programar.

Primer Algoritmo Chafa

- Por ejemplo, supongamos que nos dan una lista gigante de empleados, todos en desorden, y nos piden que encontremos al empleado "Crescencio Creciente".
- En realidad no podemos hacer otra cosa mas que agarrar al primer empleado, ver si es Crescencio, y si no, seguir con el segundo, etc. hasta encontrarlo.
- ¿Qué ventajas y desventajas tiene este algoritmo?
  - Ventaja: Es muy fácil de programar.
  - Desventaja: Hay que revisar a cada uno de los empleados para ver si es Crescencio. Si son 100,000, esto podría tomar un buen tiempo.

Primer Algoritmo Chafa

- Por ejemplo, supongamos que nos dan una lista gigante de empleados, todos en desorden, y nos piden que encontremos al empleado "Crescencio Creciente".
- En realidad no podemos hacer otra cosa mas que agarrar al primer empleado, ver si es Crescencio, y si no, seguir con el segundo, etc. hasta encontrarlo.
- ¿Qué ventajas y desventajas tiene este algoritmo?
  - Ventaja: Es muy fácil de programar.
  - Desventaja: Hay que revisar a cada uno de los empleados para ver si es Crescencio. Si son 100,000, esto podría tomar un buen tiempo.
- ¿Cuánto tiempo toma, si hay n empleados?

Primer Algoritmo Chafa

- Por ejemplo, supongamos que nos dan una lista gigante de empleados, todos en desorden, y nos piden que encontremos al empleado "Crescencio Creciente".
- En realidad no podemos hacer otra cosa mas que agarrar al primer empleado, ver si es Crescencio, y si no, seguir con el segundo, etc. hasta encontrarlo.
- ¿Qué ventajas y desventajas tiene este algoritmo?
  - Ventaja: Es muy fácil de programar.
  - Desventaja: Hay que revisar a cada uno de los empleados para ver si es Crescencio. Si son 100,000, esto podría tomar un buen tiempo.
- ¿Cuánto tiempo toma, si hay n empleados?
  - En el peor de los casos, Crescencio será el último empleado.

Primer Algoritmo Chafa

- Por ejemplo, supongamos que nos dan una lista gigante de empleados, todos en desorden, y nos piden que encontremos al empleado "Crescencio Creciente".
- En realidad no podemos hacer otra cosa mas que agarrar al primer empleado, ver si es Crescencio, y si no, seguir con el segundo, etc. hasta encontrarlo.
- ¿Qué ventajas y desventajas tiene este algoritmo?
  - Ventaja: Es muy fácil de programar.
  - Desventaja: Hay que revisar a cada uno de los empleados para ver si es Crescencio. Si son 100,000, esto podría tomar un buen tiempo.
- ¿Cuánto tiempo toma, si hay n empleados?
  - En el peor de los casos, Crescencio será el último empleado.
  - En el mejor de los casos, será el primero.

Primer Algoritmo Chafa

- Por ejemplo, supongamos que nos dan una lista gigante de empleados, todos en desorden, y nos piden que encontremos al empleado "Crescencio Creciente".
- En realidad no podemos hacer otra cosa mas que agarrar al primer empleado, ver si es Crescencio, y si no, seguir con el segundo, etc. hasta encontrarlo.
- ¿Qué ventajas y desventajas tiene este algoritmo?
  - Ventaja: Es muy fácil de programar.
  - Desventaja: Hay que revisar a cada uno de los empleados para ver si es Crescencio. Si son 100,000, esto podría tomar un buen tiempo.
- ¿Cuánto tiempo toma, si hay n empleados?
  - En el peor de los casos, Crescencio será el último empleado.
  - En el mejor de los casos, será el primero.
  - En promedio, habrá que hacer $\frac{n}{2} \in \Theta(n)$ comparaciones para encontrar a Crescencio.

¿Podemos mejorar?

¿Podemos mejorar?

- En realidad, en esta situación, no. Finalmente tenemos que verlos a todos por lo menos una vez, para ver si está o no Crescencio.

¿Podemos mejorar?

- En realidad, en esta situación, no. Finalmente tenemos que verlos a todos por lo menos una vez, para ver si está o no Crescencio.
- PERO, supongamos que ahora nos dan la lista de empleados, y nos dicen que todos los días nos van a estar preguntando por alguien diferente.

¿Podemos mejorar?

- En realidad, en esta situación, no. Finalmente tenemos que verlos a todos por lo menos una vez, para ver si está o no Crescencio.
- PERO, supongamos que ahora nos dan la lista de empleados, y nos dicen que todos los días nos van a estar preguntando por alguien diferente.
- Quizás, entonces, valdría la pena pasar un tiempo poniendo la lista en orden alfabético, para que futuras búsquedas sean más fáciles.

¿Podemos mejorar?

- En realidad, en esta situación, no. Finalmente tenemos que verlos a todos por lo menos una vez, para ver si está o no Crescencio.
- PERO, supongamos que ahora nos dan la lista de empleados, y nos dicen que todos los días nos van a estar preguntando por alguien diferente.
- Quizás, entonces, valdría la pena pasar un tiempo poniendo la lista en orden alfabético, para que futuras búsquedas sean más fáciles.
- ¿Cómo ordenamos la lista? Eso es la segunda parte de la clase, o quizás otra clase.

¿Podemos mejorar?

- En realidad, en esta situación, no. Finalmente tenemos que verlos a todos por lo menos una vez, para ver si está o no Crescencio.
- PERO, supongamos que ahora nos dan la lista de empleados, y nos dicen que todos los días nos van a estar preguntando por alguien diferente.
- Quizás, entonces, valdría la pena pasar un tiempo poniendo la lista en orden alfabético, para que futuras búsquedas sean más fáciles.
- ¿Cómo ordenamos la lista? Eso es la segunda parte de la clase, o quizás otra clase.
- Pero supongamos que ya la tenemos ordenada. ¿Cómo encontramos a Crescencio más rápido?

Búsqueda Binaria

Búsqueda Binaria

- Otra vez, importa mucho qué nos permitimos hacer y qué no.

Búsqueda Binaria

- Otra vez, importa mucho qué nos permitimos hacer y qué no.
- Quizás podemos ir directo al archivero que tiene a la “C” y luego buscar ahí a Crescencio.

Búsqueda Binaria

- Otra vez, importa mucho qué nos permitimos hacer y qué no.
- Quizás podemos ir directo al archivero que tiene a la “C” y luego buscar ahí a Crescencio.
- Chin, pero dentro de la “C” están Cárdenas, Colosi, Castro, Cabrera, etc.

Búsqueda Binaria

- Otra vez, importa mucho qué nos permitimos hacer y qué no.
- Quizás podemos ir directo al archivero que tiene a la “C” y luego buscar ahí a Crescencio.
- Chin, pero dentro de la “C” están Cárdenas, Colosi, Castro, Cabrera, etc.
- Dentro de la C, está difícil también.

Búsqueda Binaria

- Otra vez, importa mucho qué nos permitimos hacer y qué no.
- Quizás podemos ir directo al archivero que tiene a la "C" y luego buscar ahí a Crescencio.
- Chin, pero dentro de la "C" están Cárdenas, Colosi, Castro, Cabrera, etc.
- Dentro de la C, está difícil también.
- Podríamos hacer sub-folders para apellidos que empiecen con "Ca", otros con "Cb", etc.

Búsqueda Binaria

- Otra vez, importa mucho qué nos permitimos hacer y qué no.
- Quizás podemos ir directo al archivero que tiene a la "C" y luego buscar ahí a Crescencio.
- Chin, pero dentro de la "C" están Cárdenas, Colosi, Castro, Cabrera, etc.
- Dentro de la C, está difícil también.
- Podríamos hacer sub-folders para apellidos que empiecen con "Ca", otros con "Cb", etc.
- ¿Y luego?

Búsqueda Binaria

- Otra vez, importa mucho qué nos permitimos hacer y qué no.
- Quizás podemos ir directo al archivero que tiene a la "C" y luego buscar ahí a Crescencio.
- Chin, pero dentro de la "C" están Cárdenas, Colosi, Castro, Cabrera, etc.
- Dentro de la C, está difícil también.
- Podríamos hacer sub-folders para apellidos que empiecen con "Ca", otros con "Cb", etc.
- ¿Y luego?
- En algún momento nos tenemos que poner a buscar.

Búsqueda Binaria

- Vamos a pensar que podemos fijarnos en alguien en particular, y ver si al que estamos buscando está antes, o después.

Búsqueda Binaria

- Vamos a pensar que podemos fijarnos en alguien en particular, y ver si al que estamos buscando está antes, o después.
- Podemos encontrarlo descartando en cada momento a la MITAD de las cosas.

Búsqueda Binaria

- Vamos a pensar que podemos fijarnos en alguien en particular, y ver si al que estamos buscando está antes, o después.
- Podemos encontrarlo descartando en cada momento a la MITAD de las cosas.
- Para buscar dentro de una lista ordenada, podemos hacer lo mismo: Lo comparamos con el del medio. Luego, dependiendo si es más grande o más chico, descartamos la mitad, y volvemos a empezar, pero sólo con la mitad de la lista.

Búsqueda Binaria

- Vamos a pensar que podemos fijarnos en alguien en particular, y ver si al que estamos buscando está antes, o después.
- Podemos encontrarlo descartando en cada momento a la MITAD de las cosas.
- Para buscar dentro de una lista ordenada, podemos hacer lo mismo: Lo comparamos con el del medio. Luego, dependiendo si es más grande o más chico, descartamos la mitad, y volvemos a empezar, pero sólo con la mitad de la lista.
- A este proceso se le llama Búsqueda Binaria (binary search) y es muy utilizado en el mundo.

Búsqueda Binaria

- Vamos a pensar que podemos fijarnos en alguien en particular, y ver si al que estamos buscando está antes, o después.
- Podemos encontrarlo descartando en cada momento a la MITAD de las cosas.
- Para buscar dentro de una lista ordenada, podemos hacer lo mismo: Lo comparamos con el del medio. Luego, dependiendo si es más grande o más chico, descartamos la mitad, y volvemos a empezar, pero sólo con la mitad de la lista.
- A este proceso se le llama Búsqueda Binaria (binary search) y es muy utilizado en el mundo.
- Si en cada momento descartamos a la mitad de las cosas, entonces el algoritmo usa tiempo en $O(\log(n))$.

Pseudocódigo de Búsqueda Binaria: recursivo

Input: Una lista ordenada de n números y un número k a buscar.
Output: La posición en la que está el número, si es que está, y n si no.
BusquedaBinaria(A,k,a=0,b=n):

- Si $a >= b$, regresa n
- $c = (a + b)/2$
- Si $A[c] == k$, regresa c.
- Si $A[c] < k$:
  - regresa BusquedaBinaria(A,k,a,c)
- Si $A[c] > k$,
  - regresa BusquedaBinaria(A,k,c,b)

Código en python: recursivo

Input: Una lista ordenada A, un elemento k a buscar, el índice de inicio, el último índice.
Output: El índice en que está el número k, (si es que está), y $|A|$ si no.

```
1   def binarySearch(A, k, primero=0, ultimo=|A|):
2   if primero+1 >= ultimo: # casi iguales
3     if A[primero] == k:
4       return primero
5     else:
6       return len(A)
7
8   medio = (primero + ultimo)//2
9
10  if k < A[medio]:
11    return binarySearch(A,k,primero,medio)
12  else if A[medio] < k:
13    return binarySearch(A,k,medio,ultimo)
14
15  return medio
16
```

Código en python: no recursivo

Input: Una lista ordenada A, un elemento k a buscar.
Output: El índice en que está el número k, (si es que está), y $|A|$ si no.

```
1   def binarySearch(A, k):
2   primero = 0
3   ultimo = len(A)
4
5   while primero < ultimo:
6     medio = (primero + ultimo)//2
7     if A[medio] == k:
8       return medio
9     if k < A[medio]:
10      ultimo = medio
11    else:
12      primero = medio
13
14  return len(A)
```

Partition Point

En C++ hay una función *maravillosa* llamada `std::partition_point`.

true true true true false false

Partition Point

En C++ hay una función *maravillosa* llamada `std::partition_point`.

true true true true false false

Otros candidatos menos útiles. Digamos que buscamos el número 8:

1. `std::binary_search` 1233788891010 $\rightarrow$ true
2. `std::lower_bound`: 12337888910 $\rightarrow$ 5

Partition Point

En C++ hay una función *maravillosa* llamada `std::partition_point`.

true true true true false false

Otros candidatos menos útiles. Digamos que buscamos el número 8:

1. `std::binary_search` 1233788891010 $\rightarrow$ true
2. `std::lower_bound`: 12337888910 $\rightarrow$ 5
3. `std::upper_bound` 12337888910 $\rightarrow$ 8
4. `std::equal_range` = `[std::lower_bound,std::upper_bound)`

12337888910 $\rightarrow$ $[5, 8)$

Ejercicios

- https://www.hackerrank.com/challenges/ctci-ice-cream-parlor