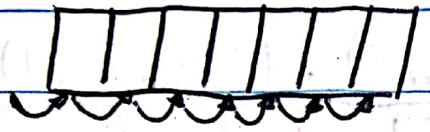
Algoritmos de búsqueda

⇒ Busqueda Secuencial

→ Es el algoritmo de búsqueda más ~~sonft. es~~ sencillo

→ Busca un elemento en una lista o un vector

→ Complejidad $O(n)$ → (peor de los casos)



⇒ Binary Search

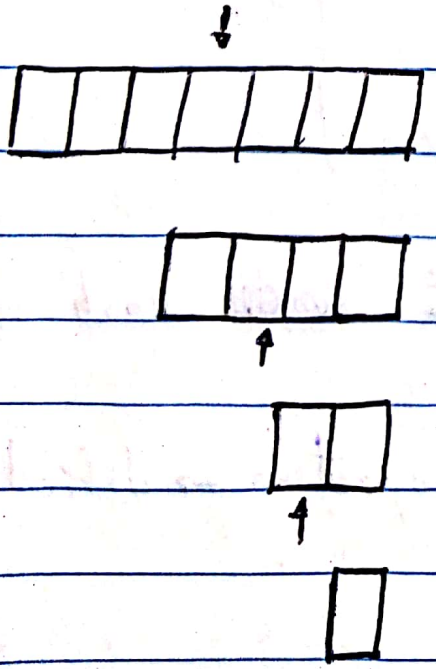
→ Busca un número en un arreglo ordenado

→ Compara el elemento buscado con el central.

→ Si el central es el elemento buscado termina

→ Si el central es menor que el buscado, se realiza la búsqueda en la mitad superior.

→ Si el central es mayor, se realiza la búsqueda en la mitad inferior.



⇒ Interpolation Search

→ Modificación de Binary Search

→ El código es prácticamente el mismo, a excepción del cálculo del elemento central.

→ En cada etapa, trata de calcular donde está el elemento central.

$$middle = low + \frac{((buscado - a[low]) * (high - low))}{a[high] - a[low]}$$

⇒ Hashing

→ Mapea grandes datasets a pequeños dataset

→ Provee una forma de buscar rápidamente

→ Determina una función de Hash que permite buscar y encontrar un índice para una llave.

f(llave) → índice único

→ Se puede aplicar en encriptación SHA-256 / MD5...

→ La forma más básica de hash es la función identidad

→ La función de hash es inyectiva → difícil encontrarlas

→ Técnicas de Hashing

- Restas sucesivas
- Cuadrado medio
- Folding
- Aritmética Modular
- Truncado

⇒ Colisiones

- Cuando una función tiene la misma imagen para distintas pre-imagenes
- Difícil de evitar.

Diseño de Algoritmos

Divide y conquista

- Técnica de diseño de algoritmos que divide el problema en subproblemas
- Conquista los subproblemas resolviéndolos recursivamente
- Combina las soluciones de los subproblemas para obtener la solución del problema original.
- Es la técnica más básica (generalmente se ve en intro)
- Ejemplos de algoritmos divide y conquista
  - Mergesort
  - Binary Search

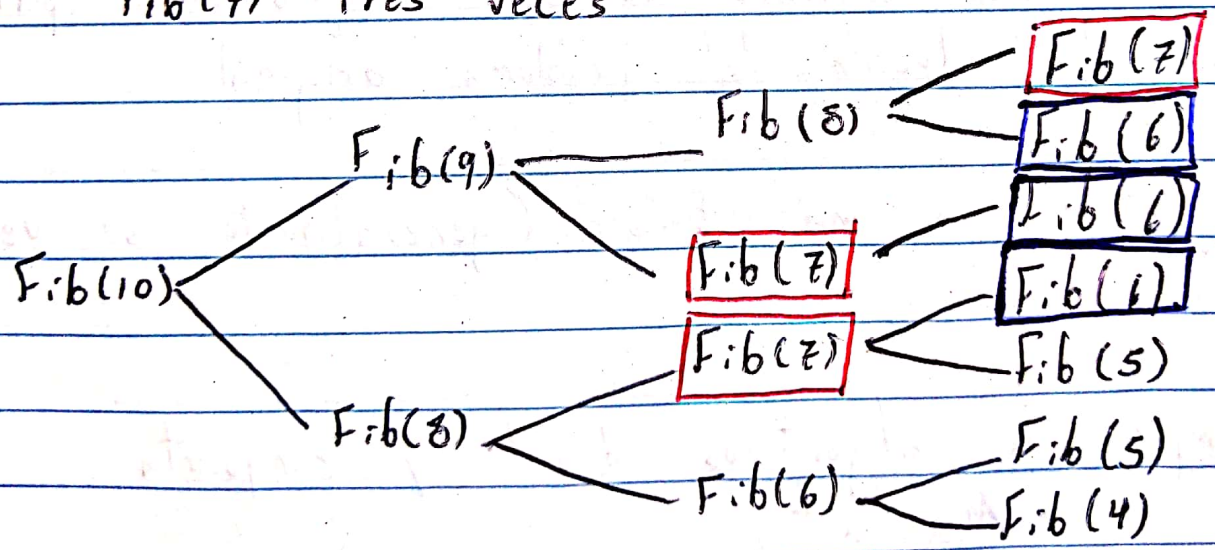
Programación dinámica

- Similar a divide y conquista
- Siempre combina resultados parciales.
- Los problemas que se resuelven con programación

dinámica tienen las siguientes características.

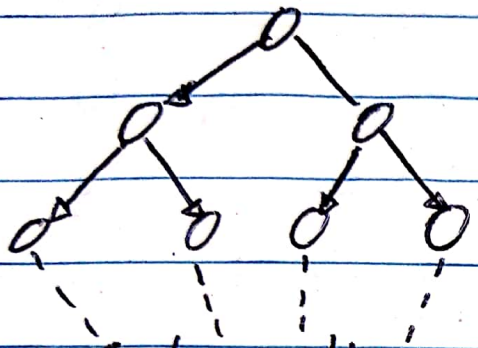
- Subestructura óptima: la solución de un problema se puede obtener mediante la combinación de sus subproblemas
- Los subproblemas se traslapan, es decir, los mismos subproblemas se resuelven N veces.

- Consider Fibonacci
  - Subestructuras óptima: cada subproblema se resuelve con la combinación de otros. Hay dos casos base Fib(0) y Fib(1)
  - Los subproblemas se traslapan:
    para calcular fib(0) se coloca la Fib(7) tres veces



- Programación dinámica calcula fib utilizando arreglos para mantener resultados, parciales y un recluta.

Divide y conquista



Cada problema es diferente

Programación dinámica