

# Análisis de Algoritmos

## Unidad 2: Algoritmos y Problemas

Ing. Matias Valdenegro T.

Universidad Tecnológica Metropolitana

18 de enero de 2012

Seleccion

Multiplicacion

Busqueda

- Busqueda Interna

- Busqueda Externa

- Busqueda en Texto

- Busqueda Virtual

Algoritmos Aleatorios

Seleccion

Multiplicacion

Busqueda

- Busqueda Interna

- Busqueda Externa

- Busqueda en Texto

- Busqueda Virtual

Algoritmos Aleatorios

# Descripcion del Problema

Dado un arreglo  $A$  de tamaño  $N$ , se pide encontrar el  $k$ -esimo elemento mas grande (o mas pequeño del arreglo). Existen varios subproblemas:

- ▶ Encontrar el minimo.
- ▶ Encontrar el maximo.
- ▶ Encontrar la mediana.

Un supuesto tipico es que el arreglo esta desordenado.

# Primera aproximacion

- ▶ Ordenar el arreglo A de mayor a menor.
- ▶ Elegir el k-esimo elemento del arreglo.
- ▶ Costo  $\Omega(n \log n)$

## Segunda aproximacion

Ejecutar un algoritmo similar al de ordenacion por seleccion, en el cual:

- ▶ Se busca el elemento mas grande del arreglo y se reemplaza por el primero.
- ▶ Se busca el elemento mas grande del arreglo, desde 1 en adelante y se reemplaza por el segundo.
- ▶ Asi hasta completar  $K$  iteraciones.

## Segunda aproximacion

```
void select(int *a, int n, int k)
{
    for(int i = 0; i < k; i++) {
        int maximo = i;

        for(int j = i; j < n; j++) {
            if(a[j] > a[maximo]) {
                maximo = j;
            }
        }

        intercambiar(a, i, maximo);
    }
}
```

Cuanto es el costo de este algoritmo?

## Segunda aproximacion

```
void select(int *a, int n, int k)
{
    for(int i = 0; i < k; i++) {
        int maximo = i;

        for(int j = i; j < n; j++) {
            if(a[j] > a[maximo]) {
                maximo = j;
            }
        }

        intercambiar(a, i, maximo);
    }
}
```

Cuanto es el costo de este algoritmo?  $O(kn)$



# Aproximaciones de Ejemplo

- ▶ Usar el metodo particion de quicksort(). (Lo veremos en la proxima unidad).
- ▶ Usar estructuras de datos.
  - ▶ Arbol Binario de Busqueda.
  - ▶ Heap.
  - ▶ Tabla de Hash por Intervalos.

# Torneo

Otra forma es usar el esquema de torneo:

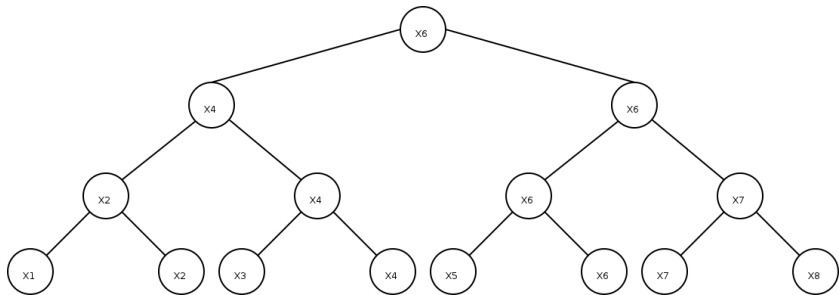
- ▶ Comparar cada par de elementos del arreglo.
- ▶ Pasar los ganadores a la siguiente ronda y comparar de la misma forma.
- ▶ Eventualmente se llegara a un ganador absoluto. Este es el maximo/minimo elemento del arreglo (Dependiendo de que comparacion se hace).
- ▶ El 2do mas grande/pequeño, se encuentra entre los perdedores directos contra el mas grande, y se organiza otro torneo para decidir cual es el 2do.
- ▶ Asi se puede repetir desde la raiz del arbol de torneo hacia abajo, para encontrar el k-esimo elemento mas grande/pequeño.

## ejemplo

Si tenemos un arreglo con los siguientes elementos:

- ▶ X1: 0
- ▶ X2: 2
- ▶ X3: 3
- ▶ X4: 4
- ▶ X5: 5
- ▶ X6: 10
- ▶ X7: 8
- ▶ X8: 6

Cual es el arbol que genera el torneo?



# Costo

Para un arreglo de  $N$  elementos, cuales son los siguientes costos del algoritmo de torneo?

- ▶ Construir el arbol.

# Costo

Para un arreglo de  $N$  elementos, cuales son los siguientes costos del algoritmo de torneo?

- ▶ Construir el arbol.
- ▶ Encontrar el mayor elemento.

# Costo

Para un arreglo de  $N$  elementos, cuales son los siguientes costos del algoritmo de torneo?

- ▶ Construir el arbol.
- ▶ Encontrar el mayor elemento.
- ▶ Encontrar el menor elemento.

# Costo

Para un arreglo de  $N$  elementos, cuales son los siguientes costos del algoritmo de torneo?

- ▶ Construir el arbol.
- ▶ Encontrar el mayor elemento.
- ▶ Encontrar el menor elemento.
- ▶ Encontrar el segundo mayor.



# Costo

Para un arreglo de  $N$  elementos, cuales son los siguientes costos del algoritmo de torneo?

- ▶ Construir el arbol.
- ▶ Encontrar el mayor elemento.
- ▶ Encontrar el menor elemento.
- ▶ Encontrar el segundo mayor.
- ▶ Encontrar el segundo menor.

# Costo

Para un arreglo de  $N$  elementos, cuales son los siguientes costos del algoritmo de torneo?

- ▶ Construir el arbol.
- ▶ Encontrar el mayor elemento.
- ▶ Encontrar el menor elemento.
- ▶ Encontrar el segundo mayor.
- ▶ Encontrar el segundo menor.

Selección

**Multiplicación**

Busqueda

Busqueda Interna

Busqueda Externa

Busqueda en Texto

Busqueda Virtual

Algoritmos Aleatorios

# Potencias

Si queremos calcular:

$$x^N = \underbrace{x \cdot x \cdot x \cdot x \cdot \dots \cdot x}_{N \text{ veces}}$$

Minimizando el numero de multiplicaciones a realizar, como se puede hacer?

# Algoritmo trivial

```
int potencia(int x, int n)
{
    int resultado = 1;

    for(int i = 0; i < n; i++) {
        resultado *= x;
    }

    return resultado;
}
```

Cual es la complejidad de este algoritmo?

# Optimizacion

Existe alguna forma de mejorar este algoritmo? (Con el fin de usar menos multiplicaciones). Por ejemplo:

$$x^4 = (x^2)^2, \quad 2 \text{ multiplicaciones}$$

$$x^5 = x * (x^2)^2, \quad 3 \text{ multiplicaciones}$$

$$x^{10} = (x^5)^2, \quad 4 \text{ multiplicaciones}$$

# Potenciación optima

Escriba un algoritmo que usando los ejemplos anteriores, calcule  $x^N$  con un minimo de multiplicaciones.  
Calcule la complejidad de su algoritmo.

## Algoritmo

```
int potencia(int x, int n)
{
    int resultado = 1;

    while(n > 0) {
        if(n % 2 == 1) {
            resultado *= x;

            if(n == 1) {
                return resultado;
            }
        }
        x *= x;
        n /= 2;
    }
    return resultado;
}
```



Seleccion

Multiplicacion

Busqueda

- Busqueda Interna

- Busqueda Externa

- Busqueda en Texto

- Busqueda Virtual

Algoritmos Aleatorios

# Busqueda

El problema consiste en buscar un elemento en una estructura de datos. Se retorna la posicion o alguna otra característica que permita acceder al elemento:

- ▶ Arreglos: Índice donde se encuentra el elemento.
- ▶ Listas Enlazadas: Puntero al Nodo que contiene el elemento.
- ▶ Diccionarios: Se busca por llave y se retorna el valor asociado.

# Tipos de Búsqueda

Existen varios tipos de búsqueda:

- ▶ Interna: Se busca en datos que están almacenados en memoria RAM.
- ▶ Externa: Se busca en datos que no están almacenados en memoria RAM, por ejemplo, en un Disco Duro.
- ▶ Texto: Se busca una cadena de caracteres dentro de otra cadena de caracteres o un texto.
- ▶ Virtual: Se busca en un espacio de búsqueda matemático o “virtual”. Por ejemplo, en problemas de Optimización.

# Busqueda Interna

# Busqueda en Arreglos

Sin tomar supuestos, escriba un algoritmo que permita buscar un elemento  $e$  en un arreglo  $a$  de largo  $n$ .  
Calcule el costo en notacion  $O$  de su algoritmo.

# Busqueda en Arreglos

```
int buscar(int *a, int largo, int e)
{
    for(int i = 0; i < largo; i++) {
        if(a[i] == e) {
            return i;
        }
    }


    return -1;
}
```

El costo de este algoritmo es  $4n$ , el cual esta en  $O(n)$ .


## Busqueda Binaria

Suponiendo que el arreglo esta ordenado de menor a mayor.


```
int busquedaBinaria(int i, int f, int *a, int e) {  
    if(i > f) {  
        return -1;  
    }  
  
    int m = (i + f) / 2;  
  
    if(a[m] > e) {  
        return busquedaBinaria(i, m - 1, a, e);  
    } else if(a[m] < e) {  
        return busquedaBinaria(m + 1, f, a, e);  
    } else {  
        return m; /* Encontrado! */  
    }  
}
```




2	9	11	15	28	33	40	47	51	64	76	77	82	85	94
---	---	----	----	----	----	----	----	----	----	----	----	----	----	----



2	9	11	15	28	33	40	47	51	64	76	77	82	85	94
---	---	----	----	----	----	----	----	----	----	----	----	----	----	----



2	9	11	15	28	33	40	47	51	64	76	77	82	85	94
---	---	----	----	----	----	----	----	----	----	----	----	----	----	----



2	9	11	15	28	33	40	47	51	64	76	77	82	85	94
---	---	----	----	----	----	----	----	----	----	----	----	----	----	----



# Costo de la Búsqueda Binaria

Si calculamos el numero de llamadas recursivas que se realiza en la búsqueda binaria:

# Costo de la Búsqueda Binaria

Si calculamos el numero de llamadas recursivas que se realiza en la búsqueda binaria:

$$T(n) = T\left(\frac{n}{2}\right) + 1$$

La cual tiene solucion:

# Costo de la Búsqueda Binaria

Si calculamos el numero de llamadas recursivas que se realiza en la búsqueda binaria:

$$T(n) = T\left(\frac{n}{2}\right) + 1$$

La cual tiene solucion:

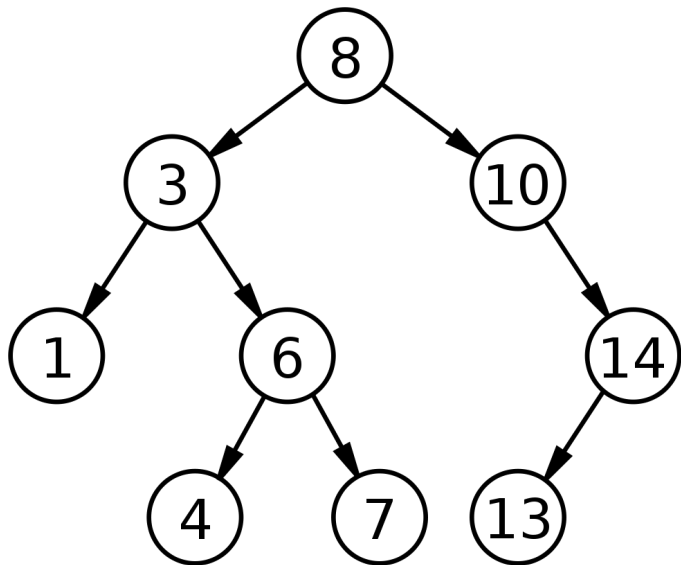
$$T(n) = O(\log_2 n)$$

Por ende la complejidad temporal de la Búsqueda Binaria es  $O(n)$ .

# Busqueda en Arboles Binarios

Un Arbol Binario de Busqueda es un Arbol que almacena llaves y valores. La condicion de Arbol Binario de Busqueda es:

- ▶ Las llaves con valor menor a la raiz se insertan en el subarbol izquierdo.
- ▶ Las llaves con valor mayor a la raiz se insertan en el subarbol derecho.



# Insercion

```
struct NodoArbol {
    NodoArbol *izq, *der;
    Llave llave;
    Valor valor;
};

void insertar(NodoArbol* &arbol, NodoArbol *nuevo)
{
    if(arbol == NULL) {
        arbol = nuevo;
    } else if(nuevo->llave < arbol->llave) {
        insertar(arbol->izq, nuevo);
    } else {
        insertar(arbol->der, nuevo);
    }
}
```

# Busqueda

```
Valor buscar(NodoArbol *arbol, Llave llave)
{
    if(arbol == NULL) {
        return NULL;
    } else if(llave < arbol->llave) {
        buscar(arbol->izq, llave);
    } else if(llave > arbol->llave) {
        buscar(arbol->der, llave);
    } else {
        return arbol->valor;
    }
}
```

# Costo de la Búsqueda en Arboles Binarios

- ▶ Costos de la Construcción/Inserción del Arbol:
  - ▶ Peor Caso:



# Costo de la Búsqueda en Arboles Binarios

- ▶ Costos de la Construcción/Inserción del Árbol:
  - ▶ Peor Caso:  $O(n)$
  - ▶ Mejor Caso:

# Costo de la Búsqueda en Árboles Binarios

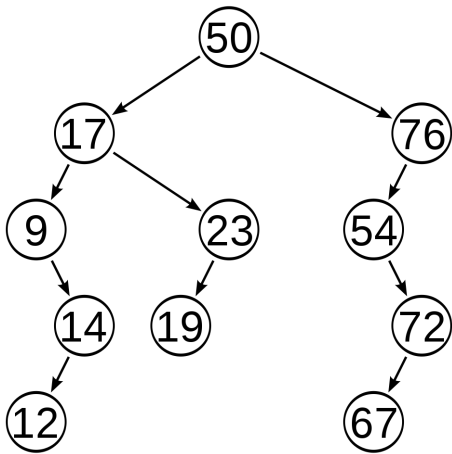
- ▶ Costos de la Construcción/Inserción del Árbol:
  - ▶ Peor Caso:  $O(n)$
  - ▶ Mejor Caso:  $O(\log n)$
- ▶ Costos de la Búsqueda en el Árbol:
  - ▶ Peor Caso:

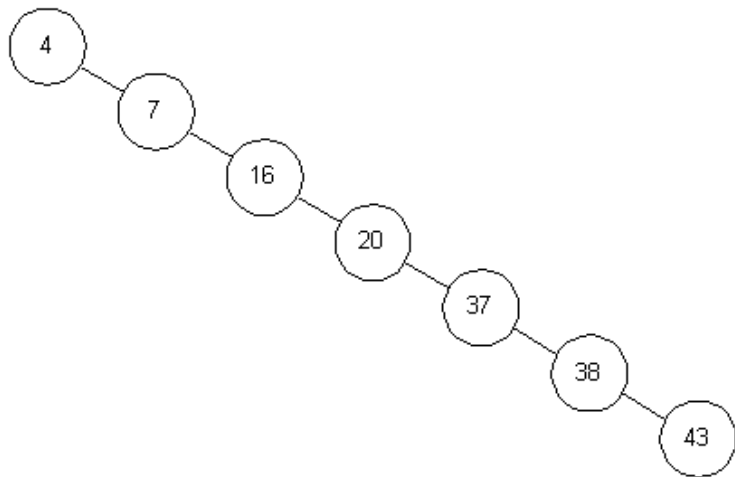
# Costo de la Búsqueda en Arboles Binarios

- ▶ Costos de la Construcción/Inserción del Arbol:
  - ▶ Peor Caso:  $O(n)$
  - ▶ Mejor Caso:  $O(\log n)$
- ▶ Costos de la Búsqueda en el Arbol:
  - ▶ Peor Caso:  $O(n)$
  - ▶ Mejor Caso:

# Costo de la Búsqueda en Árboles Binarios

- ▶ Costos de la Construcción/Inserción del Árbol:
  - ▶ Peor Caso:  $O(n)$
  - ▶ Mejor Caso:  $O(\log n)$
- ▶ Costos de la Búsqueda en el Árbol:
  - ▶ Peor Caso:  $O(n)$
  - ▶ Mejor Caso:  $O(\log n)$





# Costos en Arboles AVL

- ▶ Costos de la Construcción/Inserción del Arbol:
  - ▶ Peor Caso:

# Costos en Arboles AVL

- ▶ Costos de la Construcción/Inserción del Árbol:
  - ▶ Peor Caso:  $O(\log n)$
  - ▶ Mejor Caso:



# Costos en Arboles AVL

- ▶ Costos de la Construcción/Inserción del Árbol:
  - ▶ Peor Caso:  $O(\log n)$
  - ▶ Mejor Caso:  $O(\log n)$
- ▶ Costos de la Búsqueda en el Árbol:
  - ▶ Peor Caso:

# Costos en Arboles AVL

- ▶ Costos de la Construcción/Inserción del Árbol:
  - ▶ Peor Caso:  $O(\log n)$
  - ▶ Mejor Caso:  $O(\log n)$
- ▶ Costos de la Búsqueda en el Árbol:
  - ▶ Peor Caso:  $O(\log n)$
  - ▶ Mejor Caso:

# Costos en Arboles AVL

- ▶ Costos de la Construcción/Inserción del Árbol:
  - ▶ Peor Caso:  $O(\log n)$
  - ▶ Mejor Caso:  $O(\log n)$
- ▶ Costos de la Búsqueda en el Árbol:
  - ▶ Peor Caso:  $O(\log n)$
  - ▶ Mejor Caso:  $O(\log n)$

# Busqueda en Listas Enlazadas

Como se puede buscar un elemento en una lista enlazada? Sin suponer nada sobre la lista.

# Busqueda en Listas Enlazadas

```
struct Nodo {  
    Nodo *prox;  
    T dato;  
}  
  
Nodo *buscar(Nodo *lista, T dato)  
{  
    Nodo *p = lista;  
  
    while(p != NULL) {  
        if(p->dato == dato) {  
            return p;  
        }  
        p = p->prox;  
    }  
    return NULL;  
}
```

# Busqueda en Listas Enlazadas

```
Nodo *buscar(Nodo *lista, T dato)
{
    Nodo *p = lista;
    Nodo *ret = NULL;

    while(p != NULL) {
        if(p->dato == dato) {
            ret = p;
        }
        p = p->prox;
    }
    return ret;
}
```

Encuentre la diferencia en complejidad con respecto a la slide anterior.

# Busqueda Externa

# Busqueda Externa

Como se menciona previamente, la Busqueda Externa es buscar en estructuras de datos que no estan almacenadas en la memoria RAM, por ejemplo, buscar archivos en el Disco Duro o en un Sistema de Archivos.

- ▶ La principal metrica de optimizacion de algoritmos en este caso es el numero de accesos a la memoria externa.
- ▶ Tipicamente el acceso a un disco duro toma varios milisegundos, comparado con nanosegundos en el caso de la memoria RAM.
- ▶ Por ende, veremos algoritmos y estructuras de datos

## Ejemplos

Bases de Datos (especialmente de gran tamaño, Sistemas de Archivos (FAT, NTFS, ext3), etc.



# Solucion Trivial

La solución trivial a la búsqueda externa es simplemente recorrer toda la memoria externa en busca del dato que se quiere buscar. El costo es  $O(n)$ , lo cual es inaceptable para valores incluso pequeños de  $N$ .

## Porque?

Un disco duro de 7200 RPM posee un tiempo promedio de búsqueda de  $\sim 10ms$ . Si buscar un registro en un disco toma 20 lecturas, entonces esa búsqueda toma aproximadamente 0,2 segundos. (Bastante).

Nota: 20 lecturas es un valor cercano a  $\log_2(1000000) \sim 19,93$

# Indices

Una forma muy eficaz de buscar datos es usar un indice. Se implementa un Diccionario, donde se almacenan pares llave-valor, y se busca por la llave para obtener el valor. En el caso de la busqueda externa, tipicamente se asocian llaves numericas a bloques de datos.

Un indice trivial es tener un arreglo ordenado A de tamaño N, y otro arreglo B de tamaño M, con  $M < N$ , de tal forma que el arreglo B contenga el indice de la posicion del elemento  $i = 0, \frac{N}{M}, \frac{2N}{M}, \frac{3N}{M}, \dots, N$  del arreglo A.

3	4	5	6	7	8	9	10	14	16	19
---	---	---	---	---	---	---	----	----	----	----

Clave →

Posición →

3	6	9	16
0	3	6	9



3	4	5	6	7	8	9	10	14	16	19
---	---	---	---	---	---	---	----	----	----	----

Clave	→	3	6	9	16
Posición	→	0	3	6	9

- Cual es el costo de crear el indice?
- Cual es el costo de buscar en el indice?

# Problemas con el Indice?

- ▶ Si la informacion es estatica, el indice no debe cambiar. Pero si la informacion es dinamica?
- ▶ Que pasa si es necesario insertar informacion en el indice?
- ▶ Como se minimiza el numero de busquedas en la memoria externa?

## Opciones

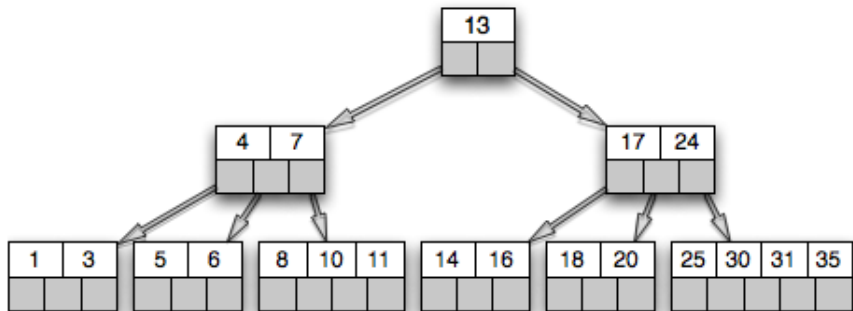
- ▶ Es posible usar indices jerarquicos, con el fin de “comprimir” mas la informacion del indice.
- ▶ Es posible usar bloques parcialmente usados con el fin de dejar espacio para informacion futura.

# B-Tree

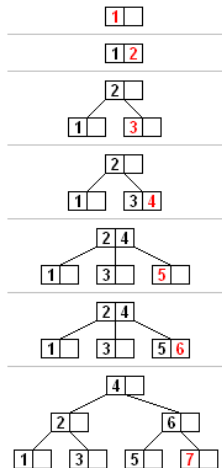
El B-Tree es una generalización del Arbol Binario de Búsqueda, el cual cumple las siguientes condiciones:

1. Cada nodo tiene a lo mas  $m$  hijos.
2. Cada nodo, excepto la raíz, tiene al menos  $\frac{m}{2}$  hijos.
3. La raíz tiene al menos 2 hijos a menos que sea una hoja.
4. Todas las hojas están al mismo nivel del árbol, y contienen los bloques de datos.
5. Un nodo que no es hoja con  $k$  hijos contiene  $k - 1$  llaves.

Aquí  $m$  se denomina el orden del árbol.







# Alturas de un B-Tree

Las alturas de un B-Tree de orden  $m$  con  $n$  llaves son:

- ▶ Mejor caso:  $O(\log_m n)$
- ▶ Peor caso:  $O(\log_{\frac{m}{2}} n)$

Dadas estas alturas, cuales son los costos de busqueda en un B-Tree?

# Como se usa?

- ▶ El problema del indice es basicamente: “Encontrar la direccion fisica de un archivo logico en la memoria externa”.
- ▶ Entonces, como se puede usar un B-Tree como un indice?
- ▶ Que desventaja debe tener?

## Busqueda en Texto

# Definicion del Problema

La busqueda en texto consiste en encontrar la posicion de una cadena de caracteres definida (el patron) dentro de otra cadena de caracteres (el texto). Por ejemplo, se quiere buscar “aba” dentro del texto “aabaabbaabbaabababbbbabababababababbaa”.

# Notacion

- ▶  $P$  es la cadena de caracteres del patron.
- ▶  $T$  es la cadena de caracteres del texto.
- ▶  $N$  es el largo de  $P$ .
- ▶  $M$  es el largo de  $T$ .

## Algoritmo Trivial

```
for(i = 0; i < m - n; i++) {  
    j = 1;  
  
    while(j < n && T[i + j] == P[j]) {  
        j++;  
  
        if(j == (n - 1)) {  
            return i;  
        }  
    }  
}  
return -1; /* Fallo */
```

Cual es la complejidad de este algoritmo?

## Algoritmo Trivial

```
for(i = 0; i < m - n; i++) {  
    j = 1;  
  
    while(j < n && T[i + j] == P[j]) {  
        j++;  
  
        if(j == (n - 1)) {  
            return i;  
        }  
    }  
}  
return -1; /* Fallo */
```

Cual es la complejidad de este algoritmo?  $O(mn)$



# Automatas Finitos Deterministicos

Si se recuerda el ramo de Teoria de Automatas, se puede usar un AFD para buscar un patron en el texto. Recordemos que un automata es:

- ▶ Un conjunto de estados  $Q$ . (Generalmente son numeros enteros).
- ▶ Un estado inicial  $q_0$
- ▶ Un conjunto de estados finales  $A$ .
- ▶ Una funcion de transicion  $\delta(q, a)$  que determina el proximo estado a partir del estado actual y el caracter actual del String.

La pregunta es: Podemos construir un AFD que encuentre el patron en el texto?

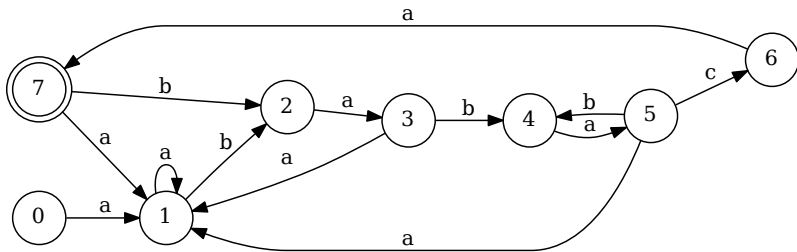
# Algoritmo AFD

```
q = 0;
for(i = 0; i < m; i++) {
    q = transicion(q, T[i]);

    if(q == n) {
        return i - m;
    }
}

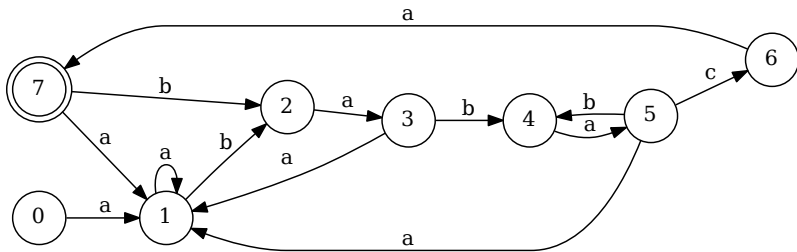
return -1 /* Error */
```

# Ejemplo



Que patron busca este AFD?

## Ejemplo



Que patron busca este AFD?  $P = ababaca$ .

# Preprocesamiento

- ▶ El costo del algoritmo para ejecutar el AFD es  $O(m)$ .
- ▶ Pero se requiere el AFD, el cual se genera preprocesando el patron a buscar, lo que toma entre  $O(n^3|\Sigma|)$  y  $O(n|\Sigma|)$
- ▶ Por ende el costo total es  $O(m + n|\Sigma|)$

# Algoritmo de Knuth-Morris-Prat

## Idea

Cuando ocurre una discordancia entre el patron y el texto, el patron contiene informacion sobre donde podria comenzar el proximo calce.

Esta informacion se puede preprocesar en una tabla T.

i	0	1	2	3	4	5	6
P[i]	A	B	C	D	A	B	D
T[i]	-1	0	0	0	0	1	2

# KMP: Busqueda

```
j = 1;
while(i + j < m) {
    while(j < n && T[i + j] == P[j]) {
        j++;

        if(j == (n - 1)) { return i; }
    }

    i = i + j - T[j];

    j = (T[j] > -1) ? T[j] : 0;
}
return -1; /* Fallo */
```

# KMP: Preprocesamiento

```
int T[n]; T[0] = -1; T[1] = 0;
int pos = 2, cnd = 0;

while(pos < n) {
    if(P[pos - 1] == P[cnd]) {
        cnd++; T[pos] = cnd; pos++;
    } else if(cnd > 0) {
        cnd = T[cnd];
    } else {
        T[pos] = 0; pos++;
    }
}
```



# Costos

Cuales son los costos de:

- ▶ Preprocesamiento:

# Costos

Cuales son los costos de:

- ▶ Preprocesamiento:  $O(n)$
- ▶ Búsqueda:

# Costos

Cuales son los costos de:

- ▶ Preprocesamiento:  $O(n)$
- ▶ Búsqueda:  $O(n + m)$

# Busqueda Virtual

# Busqueda Virtual

Este tipo de busqueda se realiza sobre espacios matematicos o espacios no tangibles (virtuales), por ejemplo:

- ▶ Buscar el valor maximo de una funcion sobre un intervalo (dominio).
- ▶ Encontrar el camino minimo entre ciertos vertices de un grafo.
- ▶ Encontrar las raices de una ecuacion no-lineal.

En todos estos casos, no se conoce el valor que se esta buscando, pero si alguna propiedad que el valor cumple.

# Algoritmos Geneticos

Un algoritmo genetico es un algoritmo denominado “evolutivo”, los cuales se inspiran en la teoria de la evolucion. Usan los siguientes conceptos:

- ▶ Cromosomas: Una posible solucion al problema (o candidato en la busqueda).
- ▶ Funcion de Fitness: Funcion que evalua (retorna un valor numerico) que tan “bueno” es un cromosoma.
- ▶ Operacion de Mutacion: Hace un cambio en un cromosoma, generando una nueva poblacion.
- ▶ Cruce (o Cruzamiento): “Une” o reproduce 2 cromosomas en un cromosoma hijo.

# Para que sirven?

- ▶ Permiten encontrar soluciones aproximadas a problemas complejos.
- ▶ Típicamente se usan para diseñar objetos.
- ▶ Se usan bastante en Optimización, especialmente cuando la función objetivo es compleja.

La gran utilidad de los algoritmos genéticos está en que la función de mutación generalmente incluye un grado de aleatoriedad, lo cual genera soluciones que el diseñador del algoritmo no considero.

# Aplicaciones

Dominio	Tipos de Aplicacion
Diseño	Diseño de semiconductores, Diseño de aviones.
Planificacion	Asignacion de recursos, Planificacion de tareas.
Robotica	Planificacion de Trayectorias.
Procesamiento de Señales	Diseño de filtros.
Optimizacion Combinatorial	Vendedor Viajero, Rutas, Coloreado de Grafos y Particionado.



# Cromosomas

Un cromosoma se puede representar como:

- ▶ Cadenas de bits.
- ▶ Numeros reales, enteros, etc.
- ▶ Alguna estructura de datos.
- ▶ En realidad puede ser cualquier valor.

# Modificacion de Cromosomas

La modificacion de los cromosomas de la poblacion consiste en las operaciones de Cruce y Mutacion:

- ▶ El cruce se efectua entre 2 cromosomas (los padres), y producen un cromosoma hijo.
- ▶ La mutacion se aplica al cromosoma hijo, y tiene una componente aleatoria.

Estas operaciones logran el movimiento en el espacio de busqueda.

# Fitness

La función de Fitness evalúa cada cromosoma en la población, con el fin de determinar los mejores sujetos en la población.

- ▶ Es el único “enlace” con el problema que se resuelve.
- ▶ Una mejor solución debería aumentar o reducir el valor de fitness, dependiendo del objetivo.

# Ejecucion de un Algoritmo Genetico

1. Generar una poblacion de cromosomas inicial (valor inicial).
2. Repetir N generaciones (o hasta que se cumpla alguna condicion):
  - 2.1 Evaluar los cromosomas de la poblacion con la funcion de fitness.
  - 2.2 Cruzar y mutar los cromosomas, generando una nueva poblacion.
  - 2.3 Evaluar los cromosomas de la nueva poblacion con la funcion de fitness.
  - 2.4 Crear la nueva poblacion final con los mejores cromosomas de la poblacion nueva.

## Ejemplo: Calculo de una raiz cuadrada

Queremos calcular  $f(x) = \sqrt{x}$ .

- ▶ Cromosoma: Un numero de punto flotante, representado IEEE-754 como un String de bits.
- ▶ Fitness: La funcion  $|x - c^2|$ , donde  $c$  es el valor en punto flotante del cromosoma.
- ▶ Mutacion: Cambiar un bit al azar de la representacion binaria del cromosoma.
- ▶ Cruce: Ninguno (no aplica porque hay poblaciones de 1 cromosoma).

## Ejemplo: Minimizacion con Restricciones

Queremos minimizar  $f(x, y) = 5x - 3y$  sujeto a  $x^2 + y^2 = 136$ .

- ▶ Cromosoma: Dos numeros de punto flotante  $(x, y)$ , representado IEEE-754 como un String de bits.
- ▶ Fitness: La funcion  $5x - 3y$ , donde  $x$  e  $y$  son los valores en punto flotante del cromosoma.
- ▶ Mutacion: Cambiar un bit al azar de la representacion binaria de la componente  $x$  del cromosoma, y recalcular la componente  $y$  segun la restriccion.
- ▶ Cruce: Ninguno (no aplica porque hay poblaciones de 1 cromosoma).

# Algunas Notas

- ▶ El numero de generaciones necesarias para producir un resultado aceptable varia con el problema.
- ▶ Si se usa un generador de numeros aleatorios, la calidad de este influye en las soluciones.
- ▶ Como regla, no usar el `rand()` de C, posee una distribucion muy mala y la implementacion es muy simple. Prefiera un algoritmo de generacion de numeros pseudoaleatorios como Mersenne-Twister.

Selección

Multiplicación

Busqueda

Busqueda Interna

Busqueda Externa

Busqueda en Texto

Busqueda Virtual

Algoritmos Aleatorios



# Algoritmos Aleatorios

Los algoritmos aleatorios son algoritmos que toman decisiones al azar.

- ▶ Requieren un generador de numeros pseudoaleatorios.
- ▶ Generalmente (pero no es requisito) tienen exito en su objetivo, pero el tiempo de ejecucion no esta acotado.
- ▶ Su complejidad se mide usando el Caso Promedio.
- ▶ Una clase de algoritmos aleatorios son los algoritmos Monte-carlo.

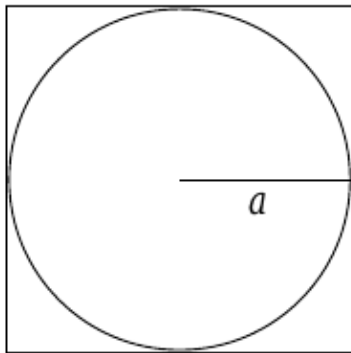
# Metodos Montecarlo

Los metodos Montecarlo son algoritmos que usan muestreo aleatorio para realizar sus calculos. Tiene una estructura que basicamente es:

1. Definir un dominio de entradas posibles.
2. Generar entradas aleatoriamente usando una distribucion de probabilidad.
3. Ejecutar un calculo deterministico sobre las entradas.
4. Combinar los resultados.

### Ejemplo: Calculo de $\pi^1$

Se tomamos un cuadrado de radio  $2a$  con un círculo de radio  $a$  circunscrito en el.



El area del cuadrado es  $4a^2$  y el del circulo es  $\pi a^2$ .

<sup>1</sup>Clase Auxiliar CC40A, Rodrigo Paredes.

## Ejemplo: Calculo de $\pi$

La razon entre el area del circulo y el cuadrado es:

$$\frac{\pi a^2}{4a^2} = \frac{\pi}{4}$$

Como podemos estimar el area del circulo y del cuadrado, usando un algoritmo aleatorio?

## Ejemplo: Calculo de $\pi$

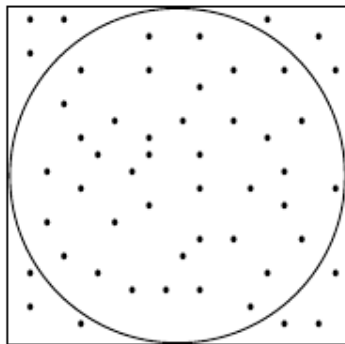
La razon entre el area del circulo y el cuadrado es:

$$\frac{\pi a^2}{4a^2} = \frac{\pi}{4}$$

Como podemos estimar el area del circulo y del cuadrado, usando un algoritmo aleatorio?

Podemos generar puntos al azar dentro del cuadrado, usando una distribucion uniforme, y si calculamos el numero de puntos dentro del cuadrado y dentro del circulo, tenemos una estimacion gruesa del area.

## Ejemplo: Calculo de $\pi$



Por ejemplo, con 55 puntos al azar, un valor estimado es  $\pi \sim 3,127$ .

## Ejemplo: Integracion

Si queremos calcular en forma numerica:

$$I = \int_b^a f(x) dx$$

Podemos realizar lo siguiente:

1. Generar  $N$  numeros  $x_i$  en  $[a, b]$ , usando una distribucion uniforme.
2. Calcular:

$$A = \frac{(b-a)}{N} \sum_{i=1}^N f(x_i)$$

# Ejemplo: Integracion

- ▶ Es bastante evidente que cuando  $N \rightarrow \infty$ ,  $E[A] \rightarrow I$ .
- ▶ El algoritmo funciona, pero converge a un valor exacto lentamente.
- ▶ Este algoritmo numerico es util cuando la funcion  $f(x)$  es compleja de evaluar en forma simbolica.
- ▶ Ademias, la complejidad del algoritmo solo depende de  $N$ , es un algoritmo  $O(n)$ , y si  $N$  es fijo, entonces el algoritmo tiene complejidad  $O(1)$ .



## Ejemplo: Busqueda

```
int buscar(int *a, int n, int e)
{
    int i = 0;

    do {
        i = azar(0, n - 1);

    } while(a[i] != e);

    return i;
}
```

## Ejemplo: Búsqueda con Montecarlo

```
int buscarMontecarlo(int *a, int n, int e, int k)
{
    do {
        int i = azar(0, n - 1);
        k--;

        if(a[i] == e) {
            return i;
        }

    } while(k > 0);

    return -1;
}
```

## Ejemplo: Analisis

Basandonos en el ejemplo anterior, el algoritmo puede fallar, cual es la probabilidad de que tenga exito?

## Ejemplo: Analisis

Basandonos en el ejemplo anterior, el algoritmo puede fallar, cual es la probabilidad de que tenga exito?

$$P[\textit{exito}] = 1 - \left(\frac{1}{n}\right)^k$$

# Aplicaciones

Los Metodos Montecarlo se usan principalmente en:

- ▶ Fisica: Modelamiento Molecular, Fisica Cuantica.
- ▶ Ingenieria: Minería, Analisis de Riesgo, Simulacion general.
- ▶ Robotica: SLAM y el Filtro Kalman.
- ▶ Computacion Grafica: Simulaciones y calculo en general, por ejemplo, Iluminacion Global.
- ▶ Otros: Integracion, Optimizacion, Problemas Inversos y Matematica Computacional.

# Ejercicios

1. Diseñe un algoritmo aleatorio que encuentre un Alumno “mejor que el promedio” en un arreglo de Alumnos.

# Ejercicios

1. Diseñe un algoritmo aleatorio que encuentre un Alumno “mejor que el promedio” en un arreglo de Alumnos.
2. Diseñe un algoritmo aleatorio que compare 2 arreglos en tiempo constante (independiente del largo de los arreglos).

# Ejercicios

1. Diseñe un algoritmo aleatorio que encuentre un Alumno “mejor que el promedio” en un arreglo de Alumnos.
2. Diseñe un algoritmo aleatorio que compare 2 arreglos en tiempo constante (independiente del largo de los arreglos).
3. Diseñe un algoritmo aleatorio (Montecarlo) que encuentre los valores que maximizan de la funcion  
$$f(x, y, z) = 139 - x^2 - y^2 - 2z$$



# Bibliografia

Para esta unidad, se recomienda leer:

1. Capítulos 5, 6 y 11 del Libro “Algoritmos Computacionales: Introduccion al Analisis y Diseño” de Sara Baase y Allen Van Gelder.