Temas principales del curso

1. **Prólogo: sobre el uso de matemáticas para algoritmos**
2. **Complejidad de algoritmos:**
   1) **Recurrencias**
   2) **Notación asimptótica y propiedades**
   3) **Clases de complejidad**
3. **Algoritmos sobre** *stable marriage*
4. **Algoritmos sobre grafos**
5. **T.S. de geometría computacional**
6. **Algoritmos sobre cadenas**
7. **Algoritmos sobre flujos**
8. **Algoritmos probabilísticos:** *streaming - sketching - sampling*

Prólogo: sobre el uso de matemáticas para algoritmos

People who analyze algorithms have double happiness. First of all they experience the sheer beauty of elegant mathematical patterns that surround elegant computational procedures. Then they receive a practical payoff when their theories make it possible to get other jobs done more quickly and more economically. ***D. E. Knuth***

Computer science is no more about computers than astronomy is about telescopes.

Edsger Dijkstra

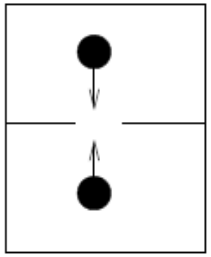
Ejemplos:

1. **Algoritmos con un componente aleatorio**
2. **Verificar si un algoritmo es correcto**
3. **Analizar complejidad de un algoritmo**

1. Algoritmos con un componente aleatorio

Ejemplo 1: Toma de decisión distribuida

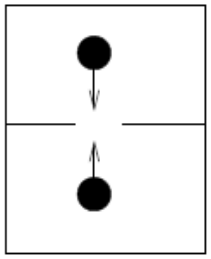
Pasar por una puerta angosta SIN recurrir a un supervisor.



1. Algoritmos con un componente aleatorio

Ejemplo 1: Toma de decisión distribuida

Pasar por una puerta angosta SIN recurrir a un supervisor.



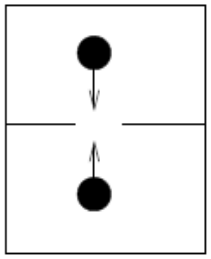
Solución:

```
cada uno lanza una moneda;
el resultado determina la acci'on (tratar de pasar o dejar pasar)
Si nadie puede avanzar, se repite lo anterior.
```

1. Algoritmos con un componente aleatorio

Ejemplo 1: Toma de decisión distribuida

Pasar por una puerta angosta SIN recurrir a un supervisor.



Solución:

```
cada uno lanza una moneda;
el resultado determina la acci'on (tratar de pasar o dejar pasar)
Si nadie puede avanzar, se repite lo anterior.
```

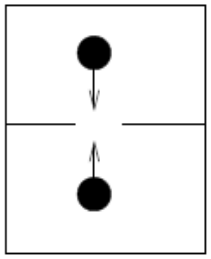
Ejemplo 2: Calcular *heavy hitters* in big data

¿Cómo calcular los valores que más ocurren en un stream de datos sin guardarlos?

1. Algoritmos con un componente aleatorio

Ejemplo 1: Toma de decisión distribuida

Pasar por una puerta angosta SIN recurrir a un supervisor.



Solución:

```
cada uno lanza una moneda;
el resultado determina la acci'on (tratar de pasar o dejar pasar)
Si nadie puede avanzar, se repite lo anterior.
```

Ejemplo 2: Calcular *heavy hitters* in big data

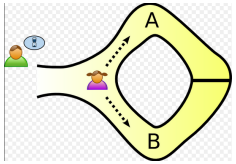
¿Cómo calcular los valores que más ocurren en un stream de datos sin guardarlos?

Ejemplo 3: Hacer cálculos conservando la privacidad

¿Cómo calcular el ingreso promedio de n personas sin revelar a los demás el ingreso de cada uno?

Ejemplo 4: Zero-knowledge proofs

Convencer a alguien de saber un secreto sin revelar el secreto.



¿Alicia sabe abrir la puerta secreta?

2. Verificar si un algoritmo es correcto

Recuérdense el significado en un algoritmo de:

$x = x + 1$

2. Verificar si un algoritmo es correcto

Recuérdense el significado en un algoritmo de:

$x = x + 1$

Quiz: describe efecto del siguiente código si $x, y \in \mathcal{N}$:

$x = x + y$

$y = x - y$

$x = x - y$

2. Verificar si un algoritmo es correcto

Ejemplo 1: calcular $a^n, n \in \mathcal{N}$

Dado a **y** n

$k = 0; \quad b = 1$

while $k \neq n$

$k = k + 1$

$b = b * a$

regresa b

2. Verificar si un algoritmo es correcto

Ejemplo 1: calcular $a^n, n \in \mathcal{N}$

Dado a **y** n

$k = 0; \quad b = 1$

$b = a^k, k \leq n$

while $k \neq n$

$k = k + 1$

$b = b * a$

regresa b

2. Verificar si un algoritmo es correcto

Ejemplo 1: calcular $a^n, n \in \mathcal{N}$

Dado a **y** n

$k = 0; \quad b = 1$

$b = a^k, k \leq n$

while $k \neq n$

$b = a^k, k < n$

$k = k + 1$

$b = b * a$

regresa b

2. Verificar si un algoritmo es correcto

Ejemplo 1: calcular $a^n, n \in \mathcal{N}$

Dado a **y** n

$k = 0; \quad b = 1$

$b = a^k, k \leq n$

while $k \neq n$

$b = a^k, k < n$

$k = k + 1$

$b = a^{k-1}, k \leq n$

$b = b * a$

$b = a^k, k \leq n$

regresa b

$b = a^k, k = n$

2. Verificar si un algoritmo es correcto

Ejemplo 2: calcular $a^n, n \in \mathcal{N}$

Dado a **y** n

$k = n; \quad b = 1; \quad c = a$

while $k \neq 0$

if $(k \ \ mod\ 2) = 0$

$k = k\ div\ 2$

$c = c * c$

else

$k = k - 1$

$b = b * c$

regresa b

2. Verificar si un algoritmo es correcto

Ejemplo 2: calcular $a^n, n \in \mathcal{N}$

Dado a **y** n

$k = n; \quad b = 1; \quad c = a$

$$a^n = b * c^k, k \geq 0$$

while $k \neq 0$

if $(k \ \ mod\ 2) = 0$

$k = k\ div\ 2$

$c = c * c$

else

$k = k - 1$

$b = b * c$

regresa b

$$a^n = b * c^k, k = 0$$

3. Analizar complejidad de un algoritmo

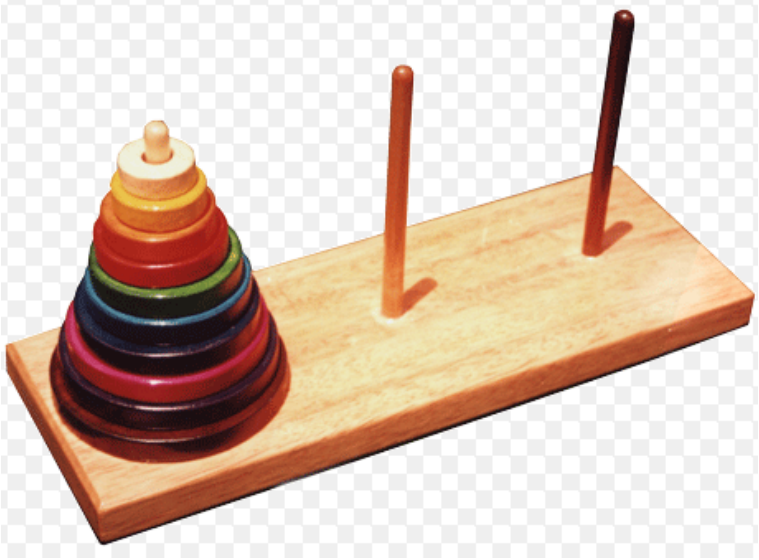
1. **Análisis de complejidad (en el tiempo o memoria):**
   estudio de cómo la complejidad cambia en función del tamaño (o parámetro fundamental) de un problema.

2. **Caso de particular interés aquí: algoritmos recursivos.**

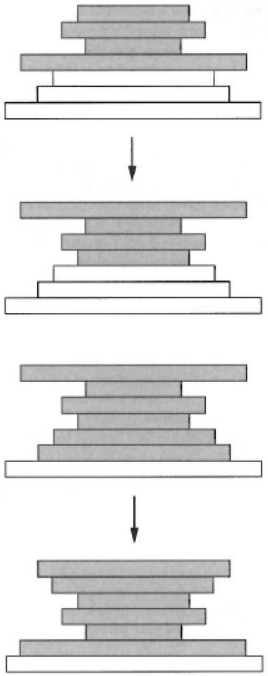
3. **Construir un algoritmo es muy ligado a derivar la ecuación de recursión de su complejidad.**

Ejemplo 1: Caja de Dominos

Ejemplo 2: Torres de Hanoi



Ejemplo 3: Ordenar hotcakes según tamaño



Ejemplo 4: Mergesort