Introducción a las Ciencias de la Computación

Ernesto Rodriguez

Universidad del Itsmo

erodriguez@unis.edu.gt

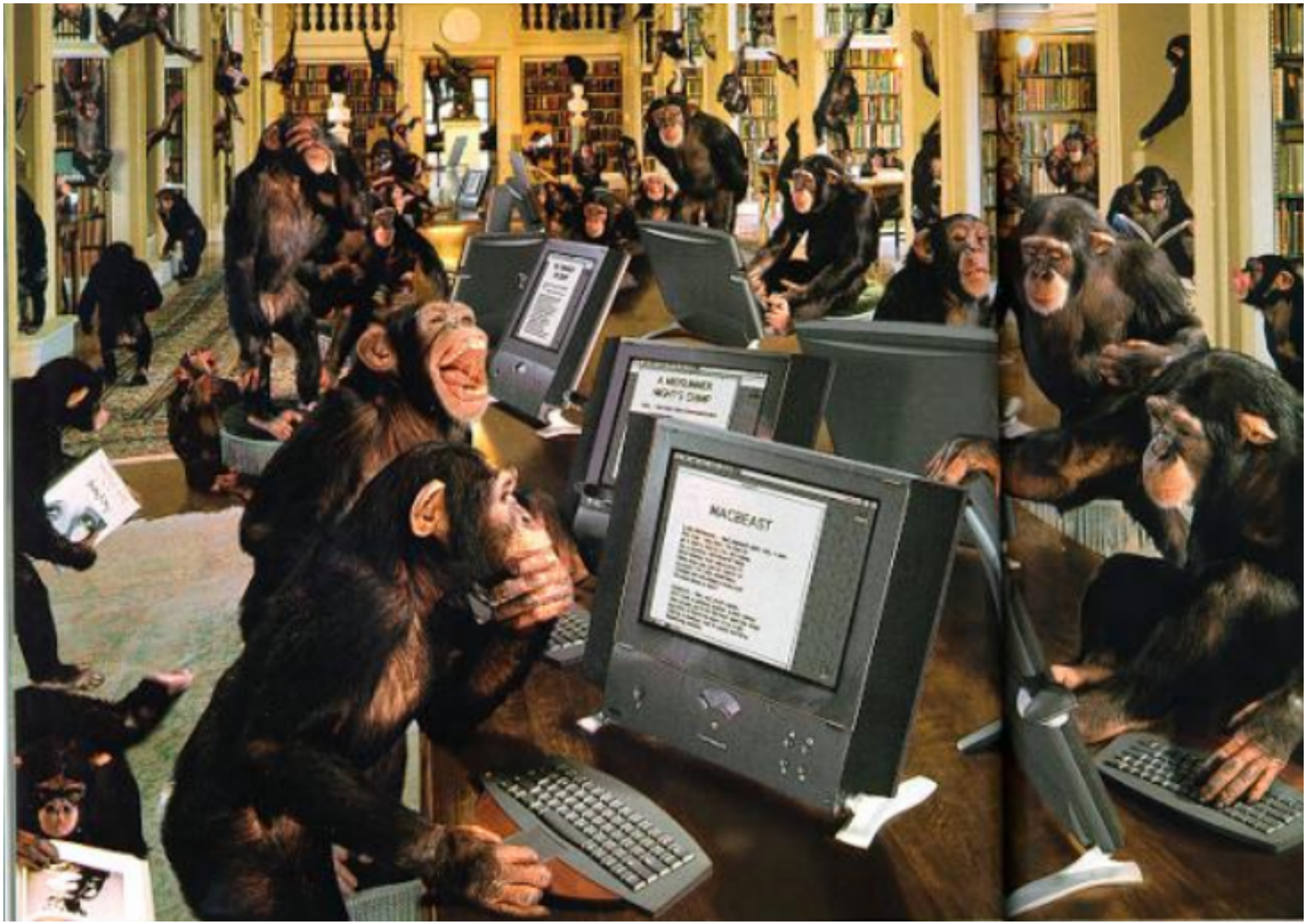
¿En que consisten las ciencias de la computación?

Ejemplo #1: Un programa que crea laberintos

- Todo laberinto debe poder resolverse
- Todo laberinto debe ser divertido
  - Las soluciones deben ser *unicas*
  - Todos los cuartos deben ser *accesibles*

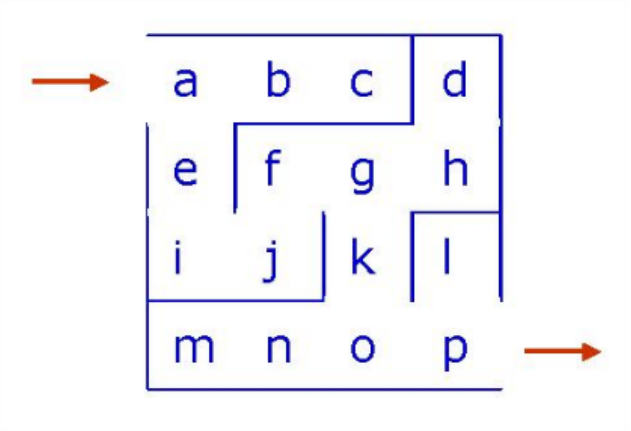
¿Como lo construimos?

Bueno, a hackear



Como razonar sobre el problema

- Empezar con un laberinto completo
- Eliminar paredes al azar hasta tener un buen laberinto.
- A cada cuarto se le puede dar un nombre



Formulación Matematica:

- Un conjunto de cuartos: $\{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p\}$
- Parejas de cuartos que estan conectados. **Ejemplo:** $\langle a, b \rangle$ o $\langle g, k \rangle$
- A esta estructura matematica se le conoce como un *grafo*

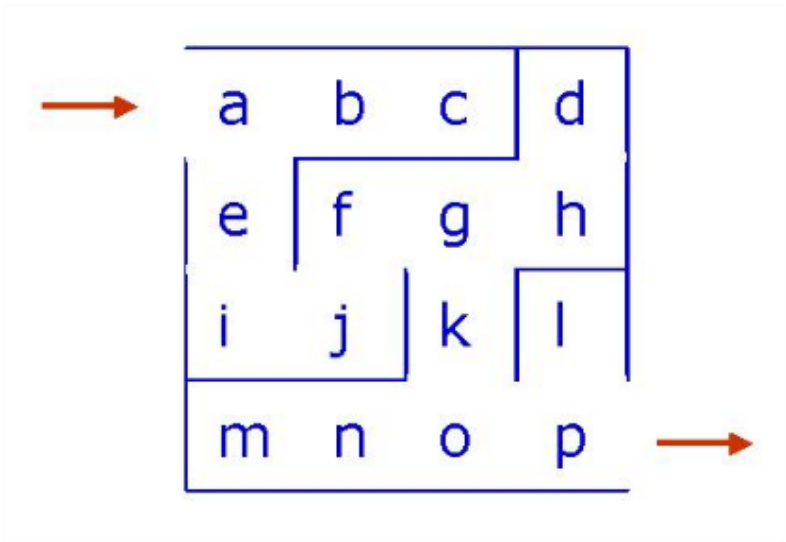
¿Por que utilizamos la matematica?

- ¿Por que es util formular un problema como conjuntos y parejas?
- Las estructuras de datos se definen matematicamente
- La matematica nos ayuda a razonar acerca de la exactitud y eficiencia de nuestros algoritmos.
- Las estructuras matematicas nos ayudan a *pensar* – nos *abstraen* de los detalles inecesarios para evitar "hackear"

Laberintos como Grafos

- **Definición:** Un grafo es un conjunto de *nodos* y *vertices* -- Son una de las estructuras de datos más utilizadas en las CC
- **Definición:** Un *laberinto* es un grafo con dos nodos especiales -- denominados como entrada y salida

Interpretación: Cada *nodo* del grafo representa un cuarto. Un *vertice* $\langle a, b \rangle$ indica que el cuarto a esta conectado con el cuarto b.

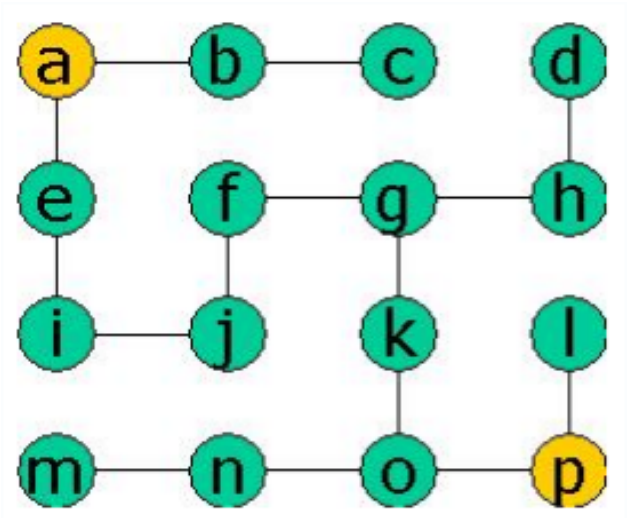


$$\left\langle \left\{ \begin{bmatrix} \langle a,e \rangle & \langle e,i \rangle & \langle i,j \rangle \\ \langle f,j \rangle & \langle f,g \rangle & \langle g,h \rangle \\ \langle d,h \rangle & \langle g,k \rangle & \langle a,b \rangle \\ \langle m,n \rangle & \langle n,o \rangle & \langle b,c \rangle \\ \langle k,o \rangle & \langle o,p \rangle & \langle l,p \rangle \end{bmatrix} \right\}, a, p \right\rangle$$

Laberintos como Grafos (visualización)

- Los grafos son objetos abstractos por lo cual es util una forma intuitiva de visualizarlos. Una opcion es un diagrama, donde los *nodos* se visualizan como puntos y los *vertices* como lineas.

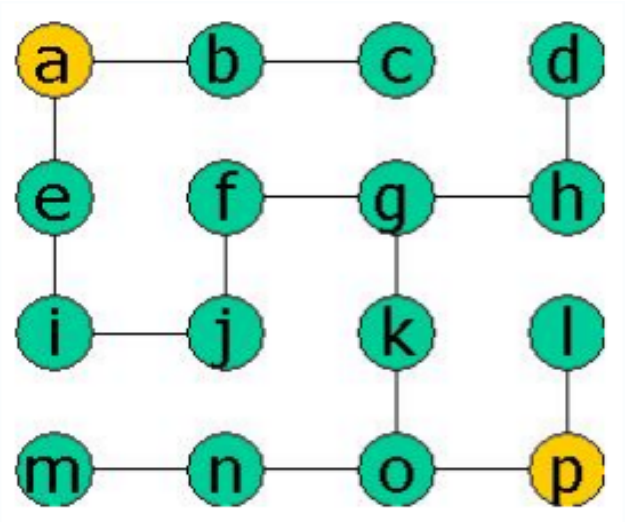
$$\left\langle \left\{ \begin{bmatrix} \langle a,e\rangle & \langle e,i\rangle & \langle i,j\rangle \\ \langle f,j\rangle & \langle f,g\rangle & \langle g,h\rangle \\ \langle d,h\rangle & \langle g,k\rangle & \langle a,b\rangle \\ \langle m,n\rangle & \langle n,o\rangle & \langle b,c\rangle \\ \langle k,o\rangle & \langle o,p\rangle & \langle l,p\rangle \end{bmatrix} \right\}, a, p \right\rangle$$



Tomar en cuenta que la que el diagrama es una **visualización** del grafo, no el grafo actual.

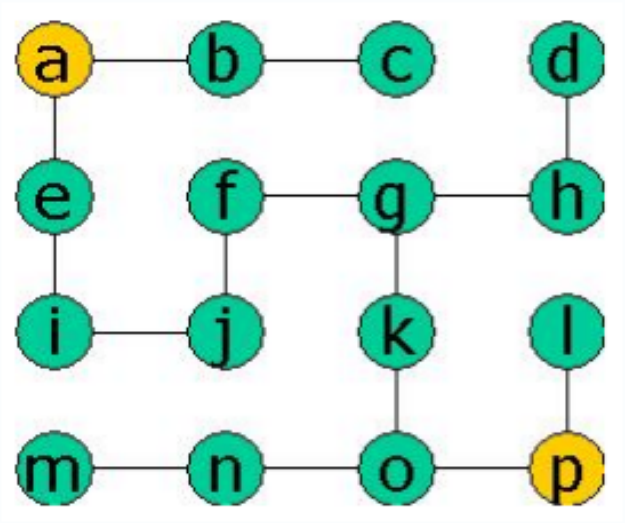
Soluciones unicas

- ¿Que *propiedad* debe tener un laberinto para que exista una solución?
- ¿Que propiedad debe tener un laberinto para que la solución sea unica?



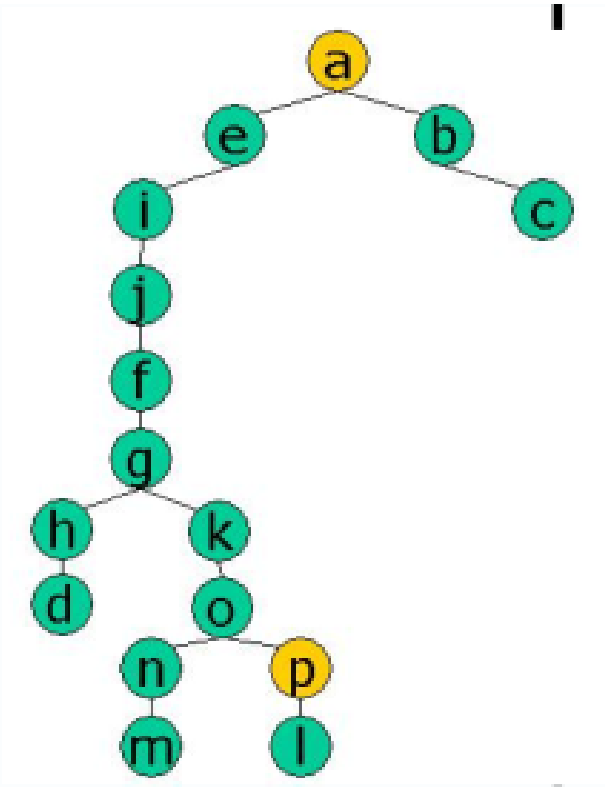
Soluciones unicas

- ¿Que *propiedad* debe tener un laberinto para que exista una solución?
  - Debe existir un camino entre *a* y *b*
- ¿Que propiedad debe tener un laberinto para que la solución sea unica?
  - El grafo debe ser un *arbol*



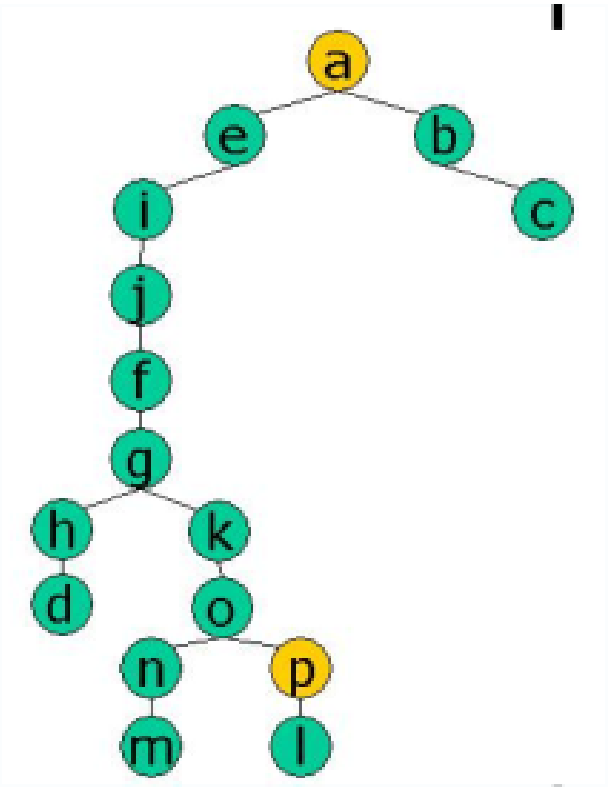
Laberintos como arboles

- **Definición** Un *arbol* es un grafo que:
  - Existe un unico *nodo raiz*
  - Cada nodo tiene un *padre*
- **Definición:** Un *arbol abarcador* es un arbol que incluye todos los nodos.
  - ¿Por que es importante tener un *arbol abarcador*?



Laberintos como arboles

- **Definición:** Un *arbol* es un grafo que:
  - Existe un unico *nodo raiz*
  - Cada nodo tiene un *padre*
- **Definición:** Un *arbol abarcador* es un arbol que incluye todos los nodos.
  - ¿Por que es importante tener un *arbol abarcador*?
    - Todos los cuartos se pueden alcanzar desde la raiz
    - El arbol no tiene ciclos



Algoritmo

- Ya que conocemos la estructura de datos, podemos considerar el algoritmo
- **Definición:** Un *algoritmo* es una serie de instrucciones para controlar un proceso computable.
- **Ejemplo:** El algoritmo de Kruskal:
  - Agregar una pareja al azar siempre y cuando no cree un ciclo
  - Repetir hasta haber construido un *arbol abarcador*
- **Ejemplo:** El algoritmo de *Busqueda de Uniones*:
  - Colocar a los nodos para los que exista un camino en una misma *clase de equivalencia*
  - Antes de agregar un vertice $\langle x, y \rangle$, revisar que x y y no se encuentren en la misma *clase de equivalencia*.

¿Que tan rapido es nuestro algoritmo?

- ¿Es rapido generar laberintos?
  - ¿Que tan rapido se generan los laberintos?
  - ¿Que significa “rapido”?
- Aparte de construir algoritmos, las ciencias de la computación tambien analizan su rendimiento.

Rendimiento y escalabilidad

- Supongamos que podemos escoger entre tres algoritmos.
- Dado un grafo con n nodos, los algritmos toman $100n$, $7n^2$ y 2^n microsegundos (μs).

Tamaño	**Rendimiento**		
n	$100n$	$7n^2$	2^n
1	$100\mu s$	$7\mu s$	$2\mu s$
5	$5ms$	$175\mu s$	$32\mu s$
10	$1ms$	$7ms$	$1ms$
100	$100ms$	$7s$	10^{16} años
1000	$1s$	$12m$	-
10000	$10s$	$20h$	-

El tiempo "-" indica tiempos más grandes que la edad del universo.