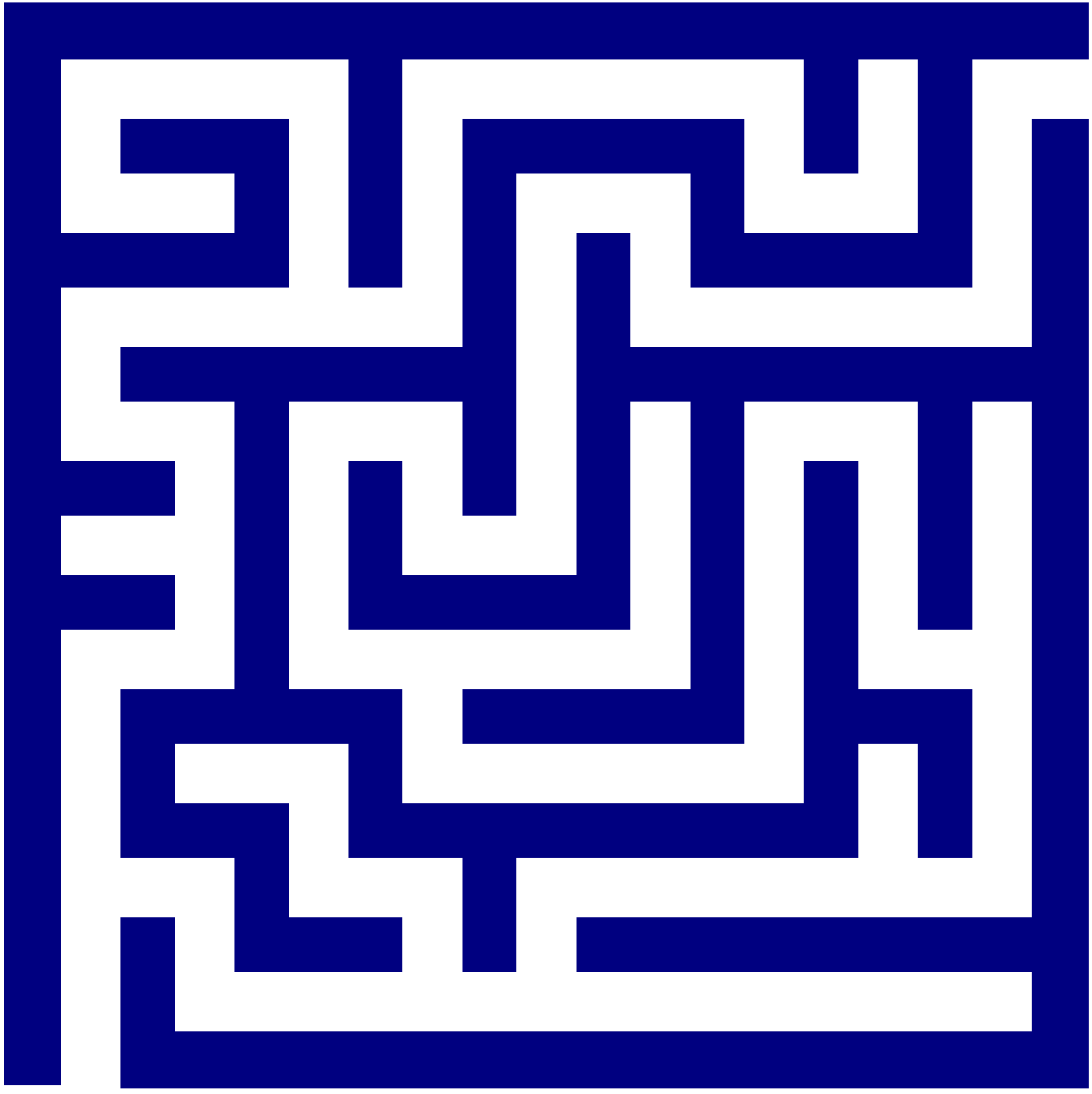
Curso de nivelación de algoritmos

Backtracking & TADs



Laberinto (sin ciclos)

Función BuscarSalida a partir de un punto p:

Laberinto (sin ciclos)

Función BuscarSalida a partir de un punto p:

- Avanzar desde p hasta llegar a:

Laberinto (sin ciclos)

Función BuscarSalida a partir de un punto p:

- Avanzar desde p hasta llegar a:
  - La salida del laberinto:
  - Un camino sin salida:
  - Una bifurcación:

Laberinto (sin ciclos)

Función BuscarSalida a partir de un punto p:

- Avanzar desde p hasta llegar a:
  - La salida del laberinto:
    - ¡Listo!
  - Un camino sin salida:
  - Una bifurcación:

Laberinto (sin ciclos)

Función BuscarSalida a partir de un punto p:

- Avanzar desde p hasta llegar a:
  - La salida del laberinto:
    - ¡Listo!
  - Un camino sin salida:
    - Volver a la última bifurcación.
  - Una bifurcación:

Laberinto (sin ciclos)

Función BuscarSalida a partir de un punto p:

- Avanzar desde p hasta llegar a:
  - La salida del laberinto:
    - ¡Listo!
  - Un camino sin salida:
    - Volver a la última bifurcación.
  - Una bifurcación:
    - **Para** cada opción posible p_i, BuscarSalida a partir de p_i.

Backtracking

Algoritmo general de backtracking:

- **Si** llegamos a una solución, mostrarla (caso base).
- Sea L la lista de formas en que se puede avanzar en la construcción de una potencial solución.
- **Para** cada elemento e de L:
  - Avanzar e.
  - Seguir construyendo recursivamente la solución.
  - Retroceder e.

Backtracking

Algoritmo general de backtracking:

- **Si** llegamos a una solución, mostrarla (caso base).
- Sea L la lista de formas en que se puede avanzar en la construcción de una potencial solución.
- **Para** cada elemento e de L:
  - Avanzar e.
  - Seguir construyendo recursivamente la solución.
  - Retroceder e.

Permite recorrer en forma exhaustiva y sistemática el espacio de soluciones a un problema dado.

Se aplica a problemas cuyan soluciones puedan construirse gradualmente.

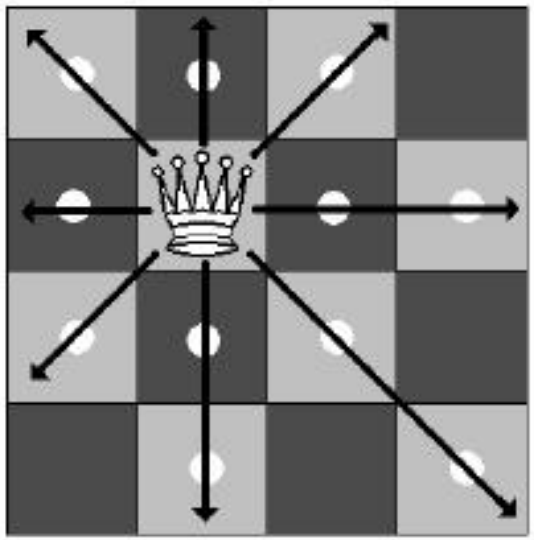

SPAIN

Problema de las 8 Reinas

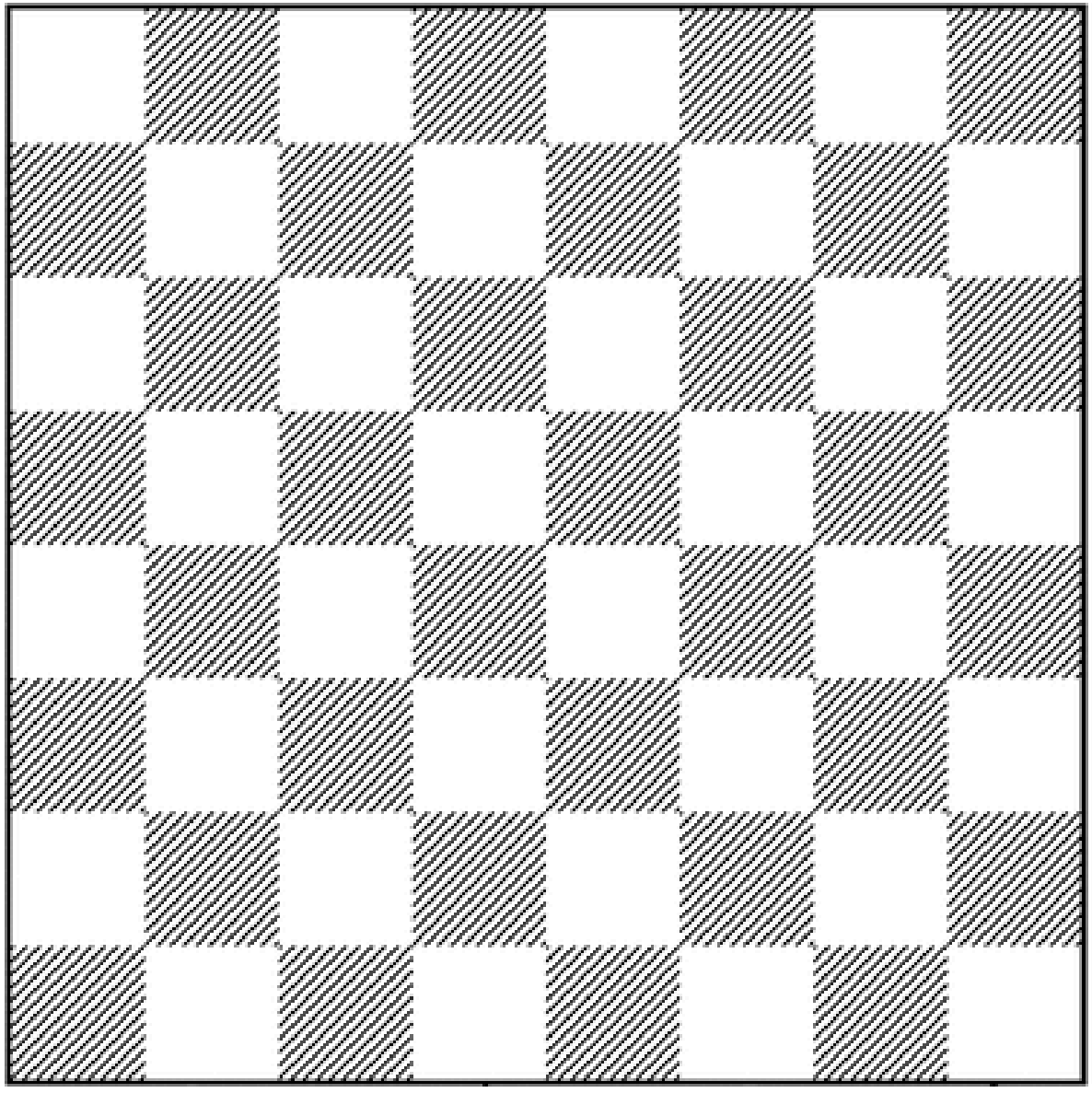
Colocar 8 reinas en un tablero de ajedrez sin que se amenacen.

(Tablero: 8x8.)

http://www.brainmetrix.com/8-queens/



Problema de las 8 Reinas



Problema de las 8 Reinas

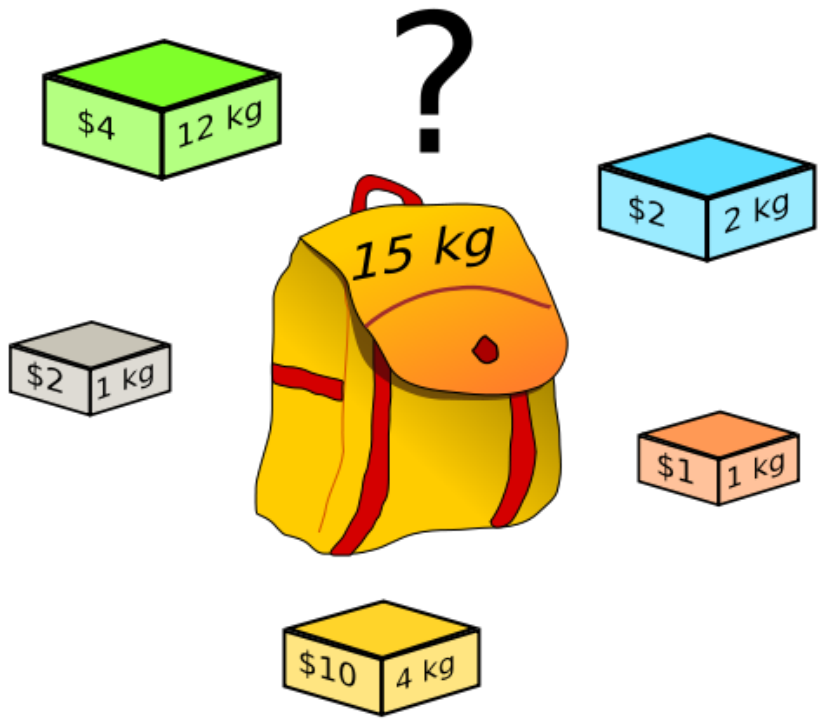
Algoritmo de backtracking:

Función OchoReinas(*col*, *tablero*):
    Si $col = 8$:
        Mostrar *tablero*.
    Si no:
        Para cada casillero c no amenazado en la columna *col*:
            Poner reina en el casillero c.
            OchoReinas($col + 1$, *tablero*)
            Sacar reina del casillero c.

Llamado principal: OchoReinas(0, tablero vacío de 8x8)

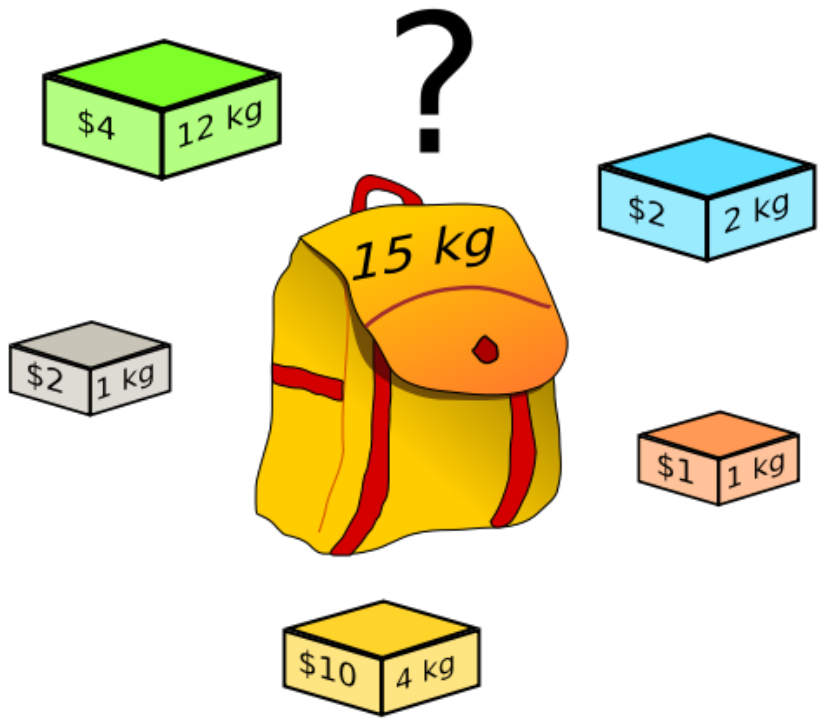
Problema de la mochila

Tenemos una mochila que puede llevar un máximo de N kg. Además, tenemos K ítems, cada uno con un peso (en kg) y un valor (en $). El problema consiste en encontrar la combinación de ítems con mayor valor que quepa en la mochila.



Problema de la mochila

Tenemos una mochila que puede llevar un máximo de N kg. Además, tenemos K ítems, cada uno con un peso (en kg) y un valor (en \$). El problema consiste en encontrar la combinación de ítems con mayor valor que quepa en la mochila.



Solución: todos los ítems menos el verde.

Tipos abstractos de datos

Un Tipo de dato abstracto (TAD) es un conjunto de datos al cual se le asocian operaciones. El TAD provee de una interfaz con la cual es posible realizar las operaciones permitidas, abstrayéndose de la manera en como estén implementadas dichas operaciones.

TAD Pila(ELEM)

Operaciones:

- CrearPila() $\rightarrow$ Pila(ELEM): Crea una pila vacía.
- Apilar(x, P): Inserta el elem. x sobre el tope de la pila P.
- Vacía?(P) $\rightarrow \mathbb{B}$: Dice si la pila P está vacía.
- Tope(P) $\rightarrow$ *ELEM*: Devuelve el elemento del tope de P.
  Precondición: $\neg$Vacía?(P).
- Desapilar(P): Borra el elemento del tope de P.
  Precondición: $\neg$Vacía?(P).

donde P : Pila(ELEM), x : ELEM (entero, char, etc.).

Las pilas tienen estrategia LIFO (*last in, first out*).

Problema: Paréntesis balanceados

Dado un string, determinar si los caracteres { }, [ ], ( ) están balanceados correctamente.

¿Se les ocurre una solución usando el tipo Pila?

Problema: Paréntesis balanceados

Dado un string, determinar si los caracteres { }, [ ], ( ) están balanceados correctamente.

¿Se les ocurre una solución usando el tipo Pila?

$p \leftarrow$ CrearPila()
Para cada caracter c en el string de entrada:
    Si c es '{', '[' o '(':
        Apilar(c, p)
    Si c es '}', ']' o ')':
        $d \leftarrow$ Tope(p)
        Desapilar(p)
        Si c no se corresponda con d: responder que no.
Si Vacía?(p), responder que sí. Si no, responder que no.

TAD Cola(ELEM)

Operaciones:

- CrearCola() $\rightarrow$ Cola(ELEM): Crea una cola vacía.
- Encolar(x, C): Agrega el elemento x al final de la cola C.
- Vacía?(C) $\rightarrow \mathbb{B}$: Dice si la cola C está vacía.
- Primero(C) $\rightarrow$ *ELEM*: Devuelve el primer elemento de C.
  Precondición: $\neg$Vacía?(C).
- Desencolar(C): Borra el primer elemento de C.
  Precondición: $\neg$Vacía?(C).

donde C : Cola(ELEM), x : ELEM (entero, char, etc.).

Las pilas tienen estrategia FIFO (*first in, first out*).

TAD Conjunto(ELEM)

Operaciones:

- CrearConjunto() $\rightarrow$ Conjunto(ELEM): Crea un conjunto vacío.
- Agregar(x, C): Agrega el elemento x al conjunto C.
- Pertenece?(x, C) $\rightarrow \mathbb{B}$: Dice si el elemento x está en C.
- Eliminar(x, C): Elimina el elemento x de C.
- Tamaño(C) $\rightarrow \mathbb{Z}$: Devuelve la cantidad de elementos de C.
- Iguales?(C_1, C_2) $\rightarrow \mathbb{B}$: Dice si los dos conjuntos son iguales.

donde C, C_1, C_2 : Conjunto(ELEM), x : ELEM.

TAD Conjunto(ELEM)

Operaciones (cont.):

- Unión(C_1, C_2) $\rightarrow$ Conjunto(ELEM): Devuelve un nuevo conjunto con $C_1 \cup C_2$.
- Intersección(C_1, C_2) $\rightarrow$ Conjunto(ELEM): Devuelve un nuevo conjunto con $C_1 \cap C_2$.
- Diferencia(C_1, C_2) $\rightarrow$ Conjunto(ELEM): Devuelve un nuevo conjunto con $C_1 \setminus C_2$.

donde C, C_1, C_2 : Conjunto(ELEM), x : ELEM.

Repaso de la clase de hoy

- Backtracking.
- Laberinto, candado numérico, 8 Reinas, Mochila.
- Tipos abstractos de datos (TADs).
- Pilas, Colas y Conjuntos.