

WUOLAH



cliptoner

www.wuolah.com/student/cliptoner



20593

Febrero2016-Ejercicio4.pdf

? Exámenes RESUELTOS | ADDA



2º Análisis y Diseño de Datos y Algoritmos



Grado en Ingeniería Informática - Tecnologías Informáticas



Escuela Técnica Superior de Ingeniería Informática
US - Universidad de Sevilla

Coucke's Academy
BY SARAH COUCKE, TEACHING SINCE 2005
www.couckesacademy.es

BRITISH COUNCIL
Aptis
Michigan

Cambridge English
Exam Preparation Centre

TRINITY
COLLEGE LONDON
Registration number 144775

Get your English certificate now!

Nervión

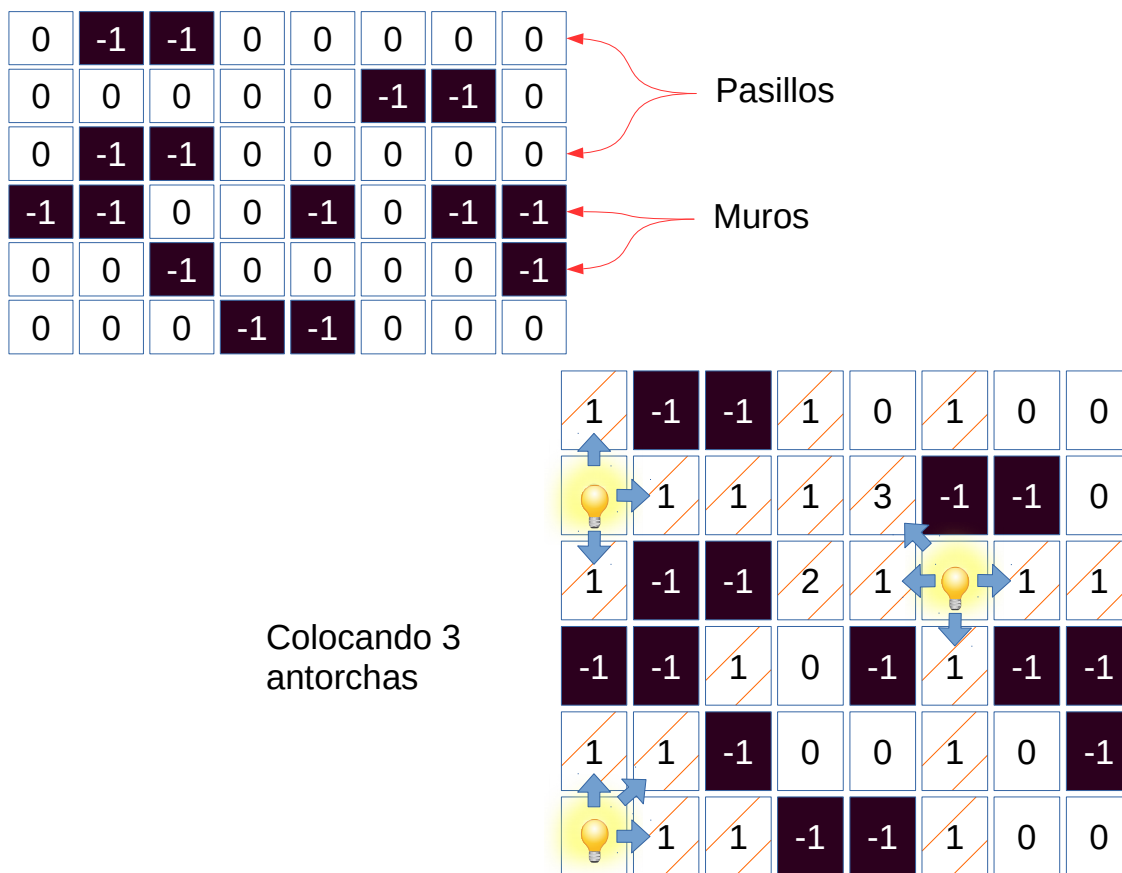
Avenida San Francisco Javier 24,
Planta Baja, Módulo 12C
954 65 98 99 - 605 54 50 19
nervion@couckesacademy.es

Macarena

Calle Don Fadrique 19
954 38 51 02 - 636 64 90 58
macarena@couckesacademy.es

Ejercicio 4: Grafos Virtuales

Tenemos un laberinto formado por “numFilas x numColumnas” casillas. Cada casilla contendrá un valor entero que nos dará información sobre si la casilla es un muro infranqueable (valor = -1), o si es un pasillo (valor ≥ 0). Los pasillos pueden estar iluminados (valor > 0) o a oscuras (valor = 0). El objetivo consistirá en iluminar todos los pasillos del laberinto con antorchas, sabiendo que una antorcha colocada en un pasillo iluminará todas las casillas en línea recta tanto en horizontal, vertical y diagonal, hasta llegar a un muro. Podemos ver un ejemplo en la figura de abajo.



Así pues, queremos saber el número mínimo de antorchas necesarias para que todas las casillas del laberinto estén iluminadas y en qué casillas debemos colocarlas.












Este problema se resolverá, partiendo de un laberinto sin ninguna antorcha, colocando una antorcha cada vez en la casilla desde la cual se consigan iluminar más casillas del laberinto. Para ello se modelará cada casilla del Laberinto mediante la clase Casilla:





















ADDA/EDA

Examen Febrero

Curso
2015/2016

 Casilla us.lsi.astar.darklab
 Casilla(int,int,int)  Casilla(Casilla)  getFila():int  getColumna():int  iluminar():void  esMuro():Boolean  estalluminada():Boolean  tieneAntorcha():Boolean  getValor():int  ponerAntorcha():void

 EstadoLaberinto us.lsi.astar.darklab
 datos: List<Casilla>  numAntorchas: Integer
 create(Integer[]):EstadoLaberinto  EstadoLaberinto(Integer[])  EstadoLaberinto(EstadoLaberinto)  getVecino(int,int):EstadoLaberinto  iluminarCasillasConAntorcha(int,int):void  calcularCasillasIluminadas(Casilla):void  getNeighborListOf():Set<EstadoLaberinto>  getNumeroCasillasOscuras():Long  getCasilla(int,int):Casilla  edgesOf():Set<SimpleEdge<EstadoLaberinto>>  isNeighbor(EstadoLaberinto):boolean  isValid():boolean  hashCode():int  equals(Object):boolean  toString():String

SE PIDE

1. ¿Determinar de qué tipo serán los vértices y aristas del grafo para resolver este problema? ¿Cuáles serán los vecinos de un vértice cualquiera? Justifique, razonadamente, qué algoritmo será el más adecuado para resolver el problema.
2. Implementar el cálculo de los estados vecinos, resultantes de colocar una antorcha en una casilla que **NO** sea un muro y que **NO** tenga ya una antorcha colocada. Para ello, se pide completar el código de los siguientes métodos:

```
public EstadoLaberinto getVecino(int f, int c) { //TODO }

public Set<EstadoLaberinto> getNeighborListOf() { //TODO }
```

3. ¿Qué habría que modificar en el código de los métodos pedidos en el apartado 2 si las antorchas iluminasen un máximo de dos casillas en todas direcciones? Razone la respuesta.

Notas:

- El coste de colocar una antorcha en una casilla cualquiera es siempre el mismo.
- Se cuenta con un método ya implementado que, dadas la fila y columna de la casilla donde se colocará una antorcha, actualiza el estado de las casillas del laberinto iluminadas por la antorcha recién colocada. Con el siguiente prototipo:

```
private void iluminarCasillasConAntorcha(int fila, int columna){}
```

Respuestas Ejercicio 4: Grafos Virtuales

1) Los vértices serán objetos de la clase EstadoLaberinto que se detalla en el enunciado del examen. Las aristas son no dirigidas. Los vecinos de un vértice cualquiera serán todos aquellos estados del laberinto iguales al actual, colocando una antorcha en cualquiera de las casillas sin antorcha (y que no sean muros). Dado que no hay pesos en los vértices y transiciones, este problema se podría resolver con A* o Dijkstra.

2)

```
public EstadoLaberinto getVecino(int f, int c) {  
    if(this.getCasilla(f,c)==null)  
        throw new IllegalArgumentException();  
  
    EstadoLaberinto vecino = new EstadoLaberinto(this);  
    Casilla c = vecino.getCasilla(f,c);  
    c.ponerAntorcha();  
    vecino.iluminarCasillasConAntorcha(f,c);  
  
    return vecino;  
}  
  
public Set<EstadoLaberinto> getNeighborListOf() {  
    return this.datos.stream().  
        filter((Casilla c) -> !c.esMuro() && !c.tieneAntorcha()).  
        <EstadoLaberinto> map((Casilla c) ->  
            this.getVecino(c.getFila(), c.getColumna())).  
        collect(Collectors.toSet());  
}
```

3) No es necesario cambiar el código de los métodos pedidos en el apartado 2, tan sólo habría que modificar la implementación del método “iluminarCasillasConAntorcha”, que es el encargado de actualizar el estado del problema.