DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico I

Algoritmos y Estructuras de Datos III
Primer Cuatrimestre de 2015

Integrante	LU	Correo electrónico
Noriega Francisco	XXX/XX	mail
Ezequiel	XXX/XX	mail
Brian	XXX/XX	mail
Aldasoro Agustina	86/13	agusaldasoro@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (54 11) 4576-3359
http://www.fcen.uba.ar

Resumen

Habiéndonos sido dado una serie de tres problemáticas a resolver, se plantean sus respectivas soluciones acorde a los requisitos pedidos. Se adjunta una descripción de cada problema y su solución, conjunto a su análisis de correctitud y de complejidad sumado a su experimentación. El lenguaje elegido para llevar a cabo el trabajo es C++.

Índice

1. Problema 1: ZombieLand

1.1. Descripción de la problemática

Dado un mapa con n ciudades, en cada una de ellas se encuentra una determinada cantidad de Zombies y una determinada cantidad de soldados (ambas también dadas). El objetivo del problema es exterminar a la invasión zombie, para ello es necesario un enfrentamiento *zombies vs soldados* por cada ciudad. Con el fin de que este enfrentamiento sea positivo, es decir se logre matar a todos los zombies de una ciudad, es necesario que la cantidad de zombies no sea diez veces más grande que la cantidad de soldados.

Como los soldados se encuentran atrincherados, se puede optar entre llevar a cabo un enfrentamiento o no (ya que no se desea provocar un ataque donde ya se sabe que se va a perder). Los soldados acuartelados no pueden moverse de la ciudad en la que están, pero sí se cuenta con una dotación de soldados extra que se la puede ubicar en cualquiera de las n ciudades acorde a lo deseado. La cantidad de soldados extra es ilimitada, más los recursos para trasladarlos no lo son. El traslado de cada soldado extra a una ciudad tiene un costo específico que varía acorde a la ciudad elegida. Siempre que sea económicamente posible, se puede trasladar una cantidad de soldados indistinta por ciudad.

Debido a que los recursos económicos son finitos, no siempre va a ser posible salvar a las n ciudades. Lo que se desea en este problema es maximizar la cantidad de ciudades salvadas, siempre y cuando se cumpla con el presupuesto. Es decir, se deben dar las cantidades de soldados necesarias para cada ciudad de modo que la cantidad de ciudades salvadas sea la óptima. El algoritmo debe tener una complejidad temporal de $O(n.log(n))$, siendo n la cantidad de ciudades del país.

Aca se podría poner unos dibujitos de soluciones óptimas como para que quede más lindo

1.2. Resolución propuesta y justificación

1.3. Análisis de la complejidad

```
struct Pepe {

    ...

};
```

1.4. Código fuente

Ejemplo de Algoritmo

```
for cada fila de la imagen do
    for cada columna de la imagen do
        if es una fila de rojos y verdes then
            if es un píxel rojo then
                canal verde ← canal verde del píxel de arriba
                canal azul ← canal azul del píxel de arriba a la izquierda
            else
                //Pixel Verde
                canal rojo ← canal rojo del píxel de la derecha
                canal azul ← canal azul del píxel de arriba
        else
            //Fila de azules y verdes
            if es un píxel verde then
                canal rojo ← canal rojo del píxel de arriba
                canal azul ← canal azul del píxel de la izquierda
            else
                //Píxel Azul
                canal rojo ← canal rojo del píxel de abajo a la derecha
                canal verde ← canal verde del píxel de la derecha
```

1.5. Experimentación

2. Problema 2: Alta Frecuencia

2.1. Descripción de la problemática

Se quiere transmitir información secuencialmente mediante un enlace el mayor tiempo continuo posible. Para poder utilizar este enlace, se debe pagar por minuto. El precio varía acorde a la frecuencia ofrecida. No todas las frecuencias funcionan a todo horario, es decir existe un horario acotado donde cada frecuencia está activa. Los datos del precio y horario de cada frecuencia son dados. Se desea optimizar este problema acorde al presupuesto gastado, es decir gastar el menor dinero posible, transmitiendo información siempre que haya alguna frecuencia activa. Se debe contar con una complejidad de $O(n.log(n))$.

Aca poner ejemplitos de cual es la solucion optima cuando hay varias senales y cuando tenes que camibar de senial porque se prendio una mas barata.

2.2. Resolución propuesta y justificación

2.3. Análisis de la complejidad

2.4. Código fuente

2.5. Experimentación

3. Problema 3: El señor de los caballos

3.1. Descripción de la problemática

En este problema, se presenta un tablero de ajedrez de tamanõ nxn el cual cuenta con cierta cantidad de caballos ubicados cada uno en una posición en el tablero sin respetar ninguna norma. Lo que se quiere lograr es *cubrir* todo el tablero. Un casillero se considera cubierto si hay un caballo en él o bien, si es una posición en la cual algún caballo existente puede moverse con un sólo movimiento. Para lograr este cometido, puede ser necesario agregar nuevas fichas *caballo* al tablero. No existe un límite en la cantidad de caballos para agregar, pero lo que se busca es dar una solución con la mínima cantidad de caballos posibles.

En la figura 1 se pueden ver todas las casillas que están cubiertas por un sólo caballo.

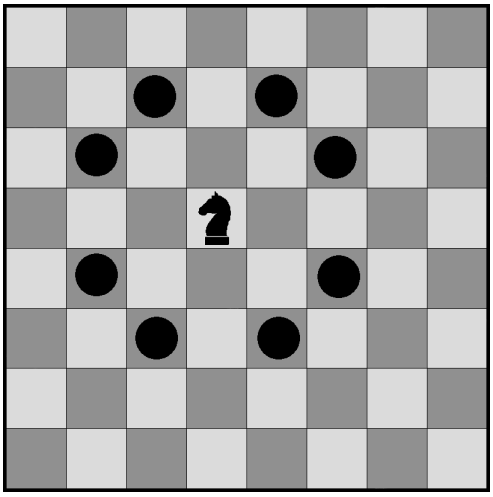

Figura 1: Casillas que *cubre* un caballo

A continuación se pueden apreciar dos soluciones al problema de cubrir el tablero.

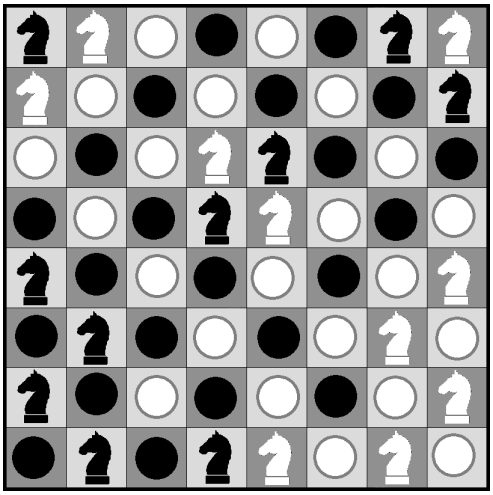

Figura 2: Solución 1

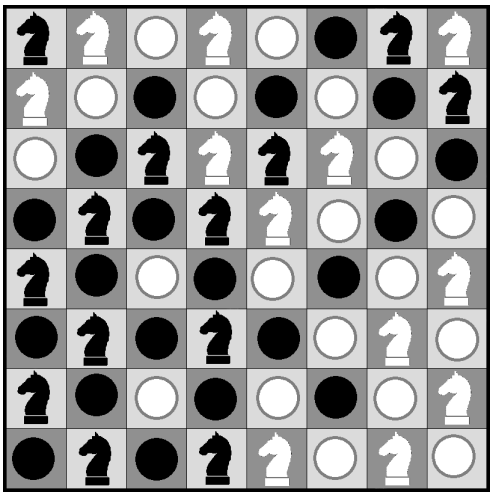

Figura 3: Solución 2

En la solución 1 se necesitaron 20 caballos, en cambio en la dos 25; es decir 5 caballos más. Por lo tanto podemos establecer que la solución 2 no es la óptima y este caso no es lo que buscamos resolver. Por otro lado, la figura 1 es óptima? Después sabremos...

3.2. Resolución propuesta y justificación

3.3. Análisis de la complejidad

3.4. Código fuente

3.5. Experimentación