Estudiante 1: Germán Alejo Domínguez
Estudiante 2: Óscar Gómez González

Problema 1:

Utilizar las técnicas de búsqueda no informada vistas en clases de teoría que estén incluidas en AIMA y mostrar los resultados obtenidos.

NOMBRE DEL PROBLEMA: Problema del puente y el farol

Estados: Tendremos un array de enteros indicando la posiciona la que se encuentran, (1-no ha cruzado, 0-si ha cruzado) además las últimas dos posiciones del array indicaran la posición del farolillo y el tiempo consumido respectivamente.

Estado inicial: El estado inicial será e array todo a 1(nadie ha cruzado) excepto la última posición con el tiempo, que se inicializara a 0.

Acciones: Una acción se considerara cruzar el puente por uno o dos personas además del farolillo, para esto se cambiaran las posiciones en el array a 0 según vayan cruzando de un lado a otro.

Modelo de transición: Se transaccionara al cruzar, cambiando el valor de la posición del array de 1 a 0 o viceversa dependiendo de hacía que lado crucemos.

Función objetivo: Todos han cruzado (array entero a 0), y el tiempo máximo es de 15 no sobrepasando dicho valor.

Coste: Consideraremos como coste el tiempo tardado, y el número de pasos, tendremos en cuenta que el tiempo nunca puede reducirse a 15 ni sobrepasarse además el número de pasos óptimos es 5.

Observaciones: Tendremos un array con los valores de tiempo que cada persona necesita para cruzar el rio, estas posiciones serán relativas al array de estado, es decir, el primero tendrá el tiempo en la primera posición del array de tiempo para cruzar(crossingTimes).

Resultados obtenidos:

```
PuenteDemo breadth -->
Action[name==AB]
Action[name==A]
Action[name==CD]
Action[name==B]
Action[name==AB]
pathCost : 5.0
nodesExpanded : 98
queueSize : 109
maxQueueSize : 110

PuenteDemo depth graph -->
Action[name==AB]
Action[name==B]
Action[name==CD]
Action[name==A]
Action[name==AB]
pathCost : 5.0
nodesExpanded : 167
queueSize : 5
maxQueueSize : 17

PuenteDemo depth limited graph -->
Action[name==AB]
Action[name==A]
Action[name==CD]
Action[name==B]
Action[name==AB]
pathCost : 5.0
nodesExpanded : 308

PuenteDemo uniform cost graph -->
Action[name==AB]
Action[name==A]
Action[name==CD]
Action[name==B]
Action[name==AB]
pathCost : 5.0
nodesExpanded : 147
queueSize : 72
maxQueueSize : 100

Puente Heuristic graph -->
Action[name==AB]
Action[name==B]
Action[name==CD]
Action[name==A]
Action[name==AB]
pathCost : 5.0
nodesExpanded : 64
queueSize : 69
maxQueueSize : 73
BUILD SUCCESSFUL (total time: 0 seconds)
```

Problema 2:

Implementar una función heurística admisible, emplearla en las búsquedas informadas indicadas en el tema correspondiente, y mostrar los resultados obtenidos.

Para la realización de la heurística hemos pensado no considerar ni la restricción de tiempo ni la del farolillo. Para el cálculo de dicha heurística recorreremos el array de estados, buscando entre las personas la que no hayan cruzado aun y calcularemos el total de número de personas que aún no han cruzado.

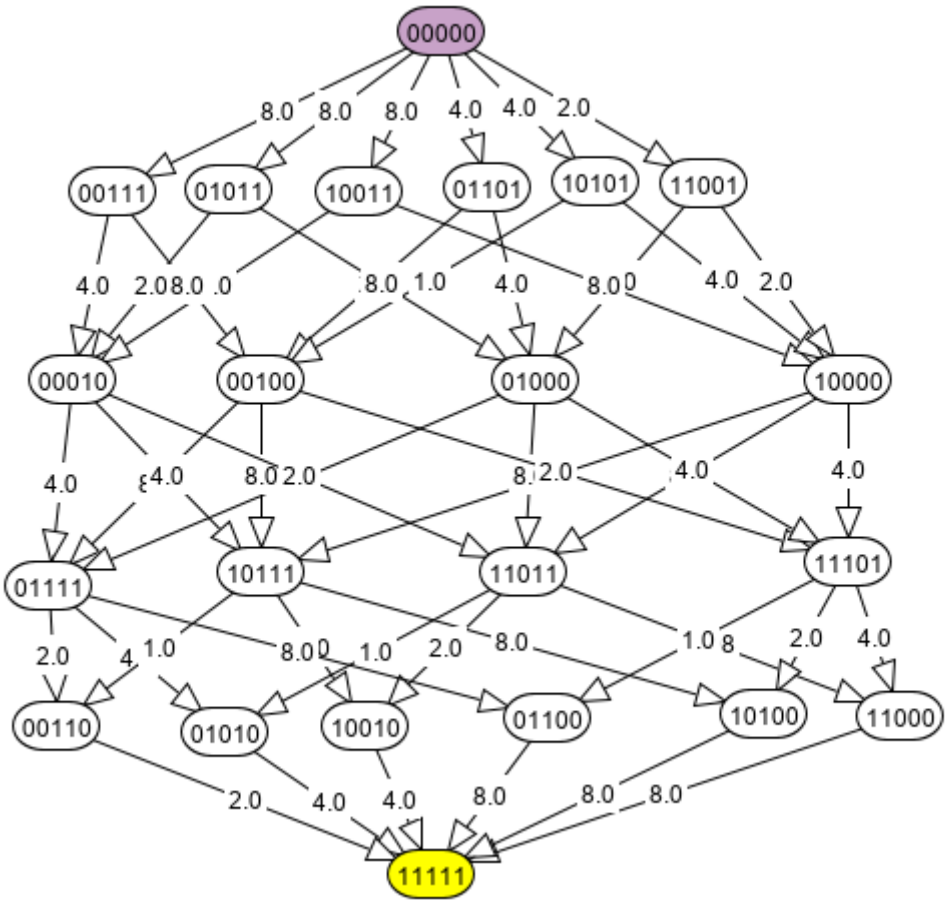
Resultado obtenido:

```
Puente Heuristic graph -->
Action[name==AB]
Action[name==B]
Action[name==CD]
Action[name==A]
Action[name==AB]
pathCost : 5.0
nodesExpanded : 64
queueSize : 69
maxQueueSize : 73
```

Problema 3:

Crear con la herramienta Search el espacio de búsqueda sobre el que se podrán realizar los posibles recorridos.

No contemplamos las combinaciones en las que solo se mueve una persona en la ida ya que en este problema nunca será algo beneficioso.



Goal Node reached! (00000 --> 11001 --> 01000 --> 01111 --> 00110 --> 11111 (Goal))