Informe proyecto 1 inteligencia artificial

“maratón de robots”

Integrantes

Cruz Dahiana Bermudez Pineda
201556055

Cristhian Rendon Sánchez
201556246

Johan Sebastián Marulanda
201556060

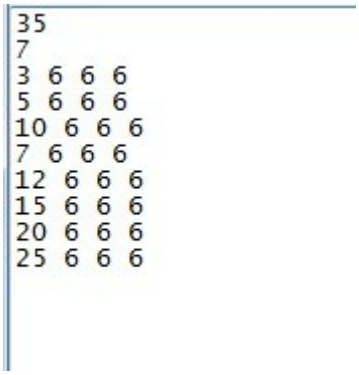
Profesor

Joshua David Triana Madrid

universidad del valle sede Tuluá

2018

Archivo texto:
para ingresar por texto la matriz el código abre el archivo y con un while lee cada linea y la guarda en una lista, por medio de un for pone los enemigos y los separa con un split que crea una lista con cada dato y después accede a cada posición para poner los enemigos.
En la siguiente imagen:
el primer dato es el tamaño de la carrera
el siguiente es cuantos obstáculos se van a poner
las siguientes lineas son la posición donde van a estar los obstáculos y los tres obstáculos



para almacenar los datos de los robots y enemigos hacemos uso de una matriz de 3 * n con un n no mayor a 50, así mismo para conocer en que lugar se encuentran los representamos de la siguiente manera .

- Tijera es representada con el numero 1.
- piedra es representada con el numero 2
- papel es representado con el numero 3
- vegetal es representado con el numero 4
- ladrón es representado con el numero 5
- homero es representado con el numero 6
- los espacios en los que no se encuentran ninguno de estos 6 elementos son 0.

también hacemos uso de algunos vectores que nos ayudan para guardar la información, uno de ellos contiene el costo g(n) actual que le toma a cada uno de los robots (tijera,piedra,papel) derrotar a el enemigo (vegetal,ladrón,homero).
Al igual que un vector copiaCosto el cual contiene el costo que lleva cada uno de los robots, para así decidir cual de los robots debe quedarse peleando y cuales seguirán avanzando.
Cada uno de ellos tienen una copia las cuales son respectivamente costosEnRama y costosMemoriza.

Tambien contamos con algunos enteros estaticos lo cuales son:

- posicionTijera: guarda la posición en la que esta la tijera.
- TijeraEspera: es utilizada para saber si la tijera se encuentra peliando con un

enemigo, si no esta peliando el valor de tijeraEspera es 0.
- copiaposicionTijera: copia de la variable posicionTijera.
- CopiatijeraEspera: copia de la variable TijeraEspera.
- PosicionPiedra: guarda la posición en la que esta la piedra.
- PiedraEspera:es utilizada para saber si la piedra se encuentra peliando con un enemigo, si no esta peliando el valor de PiedraEspera es 0.
- copiaposicionPiedra:copia de la variable posicionPiedra.
- CopiapiedraEspera:copia de la variable PiedraEspera.
- PosicionPapel: guarda la posición en la que esta el papel.
- PapelEspera:es utilizada para saber si el papel se encuentra peliando con un enemigo, si no esta peliando el valor de papelEspera es 0.
- copiaposicionPapel: copia de la variable posicionPapel.
- CopiapapelEspera: copia de la variable PapelEspera.

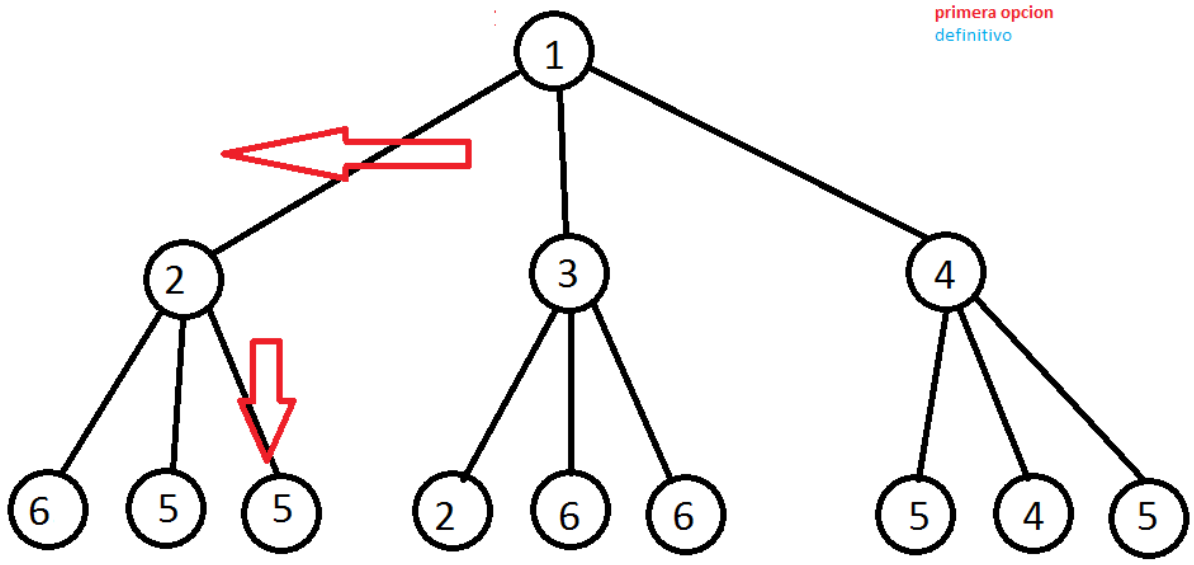
Metodo CalculaCosto:
Para especificar las reglas del maratón hacemos uso de varios if los cuales contiene las condiciones necesarias para hallar el costo g(n) que le toma a cada uno de los robots derrotar a el enemigo que este en ese momento en la posición siguiente a ellos, y guardar en el vector costo actual de la rama, y en el vector copiaCosto que tiene el costo actual mas lo que lleva de mas arriba.

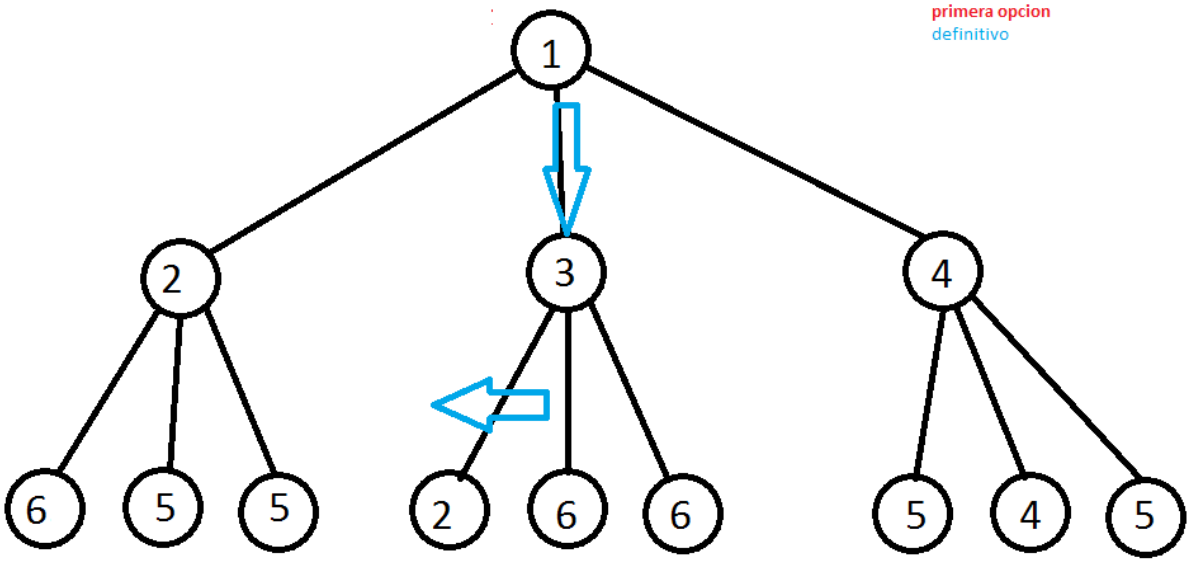
Método continuaJugando:
para ir aumentando cada unidad de tiempo hacemos uso de este método validando si lo que hay en la posición siguiente es 0 que hace referencia a que no hay nada, y si se encuentra la variable espera en 0 avanza una unidad, si se encuentra en un valor distinto a 0 se le disminuye una unidad de lo que llevaba , y si se encuentra con un robot que este peleando con un enemigo si ese robot es a quien odia (la piedra odia al papel, la tijera odia a la piedra y el papel odia la tijera) le multiplica por dos, su tiempo de espera en la variable correspondiente; si no es a quien odia divide la espera en dos.

Hacemos llamado a la función calcularcosto cuando para cada uno de los robots la posición siguiente sea diferente a 0, si los tres robots están en la misma posición por medio de varios if tomamos el valor menor en el vector copiaCosto para saber cual robot debe quedarse peleando.
Al igual hacemos las comparaciones para cuando dos robots estén la misma posición y tenga que escogerse entre esos dos cual se queda peleando.
También se tiene en cuenta el momento en que solo un robot llega a un enemigo y se calcula el costo.
El costo calculado se guarda en una variable llamada mejor para saber cual debe pelear.

Metodo compruebaGanador:
tenemos una variable ganador que inicia en false, por medio de un if validamos si la posición en la que se encuentra el robot es igual a el tamaño de la matriz la variable cambia a true.

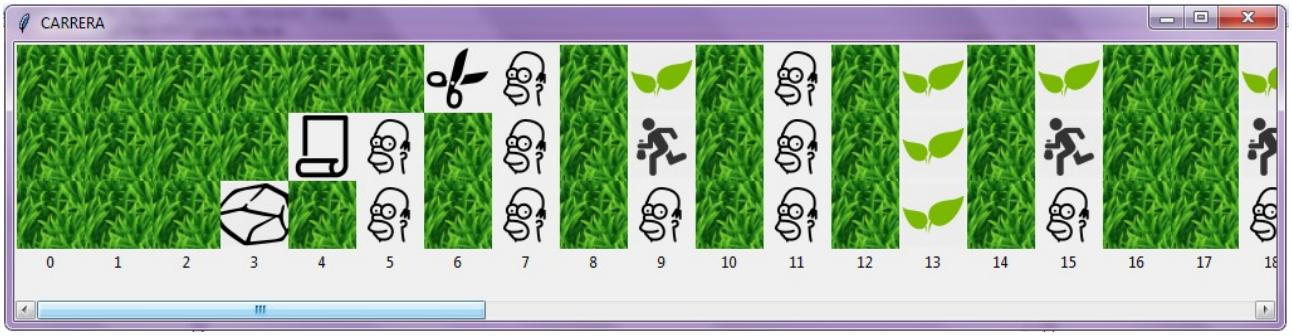
Tios: para saber si la decisión tomada es la mas apropiada hacemos uso de los tíos los cuales se encargan de mirar otro nivel más en profundidad del árbol, llegando así a dos niveles de profundidad.
Se toma el menor valor del primer nivel es decir los hijos del nodo raíz que estamos viendo en el momento, cada uno de los nodos del árbol son los costos que les toma a cada uno de los robots vencer los tres enemigos siguientes que se van encontrando, después de escoger el nodo de menor valor del primer nivel, en el siguiente paso se miran los nietos, es decir, los hijos del nodo que escogimos en primera instancia (el de menor valor del primer nivel), en este punto se realiza el procedimiento anterior, se escoge el nodo de menor valor, en este caso del segundo nivel pero ahora lo comparamos con los tíos, que serían los nodos del nivel inmediatamente superior obviamente en esta ocasión sin mirar el nodo padre, que ya habíamos escogido, y comparamos sus hijos para ver si hay algún camino que en primera instancia no parezca el óptimo pero para el segundo nivel sea de menor costo que el que pensamos anteriormente, luego si el valor de los caminos de alguno de los tíos, solo son dos como máximo, es menor se iría por este camino sino se va por el camino escogido al comienzo.

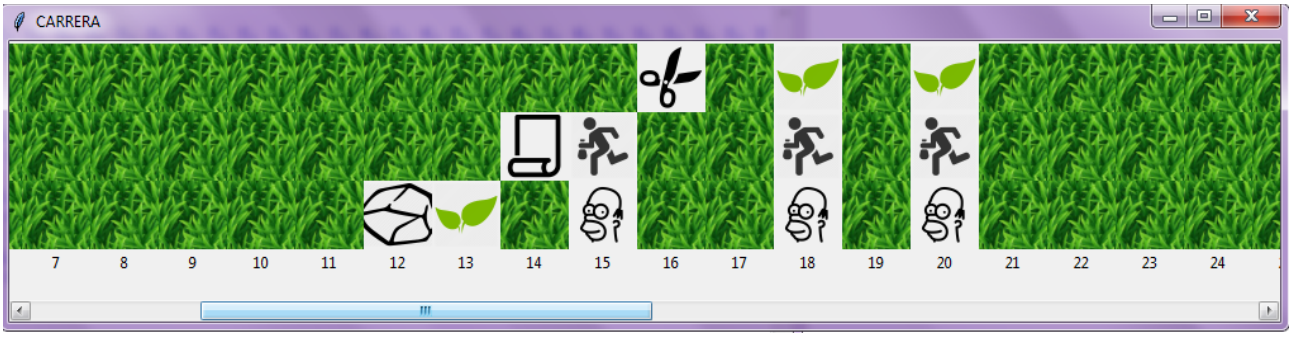




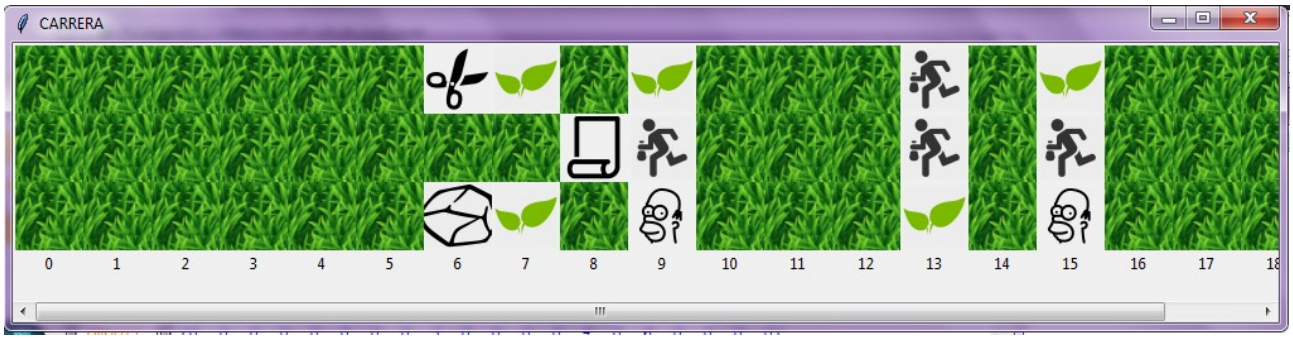
Pruebas:

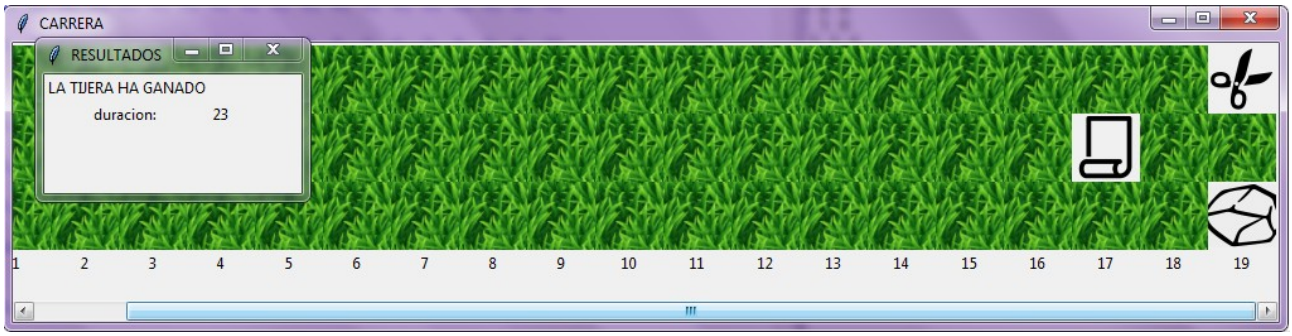
Número 1





Numero 2





Numero 3