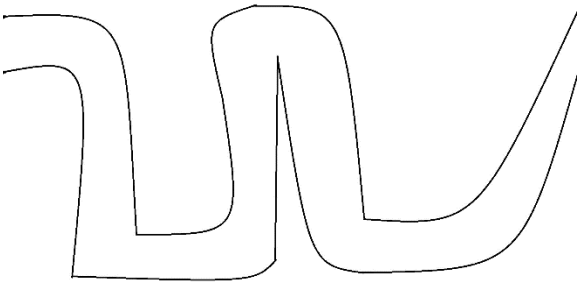
Instituto Tecnológico de Costa Rica, Escuela de Computación, Análisis de Algoritmos
Prof. Rodrigo Núñez N, proyecto II – 22%

The roadrunner

El roadrunner logra encontrar su camino de un punto de partida a uno de llegada siendo extremadamente ágil en hacer maniobras a su paso a gran velocidad.

Se le ha encomendado a usted crear el mejor roadrunner, que sea capaz de ser rápido y a la vez ágil en sus maniobras.

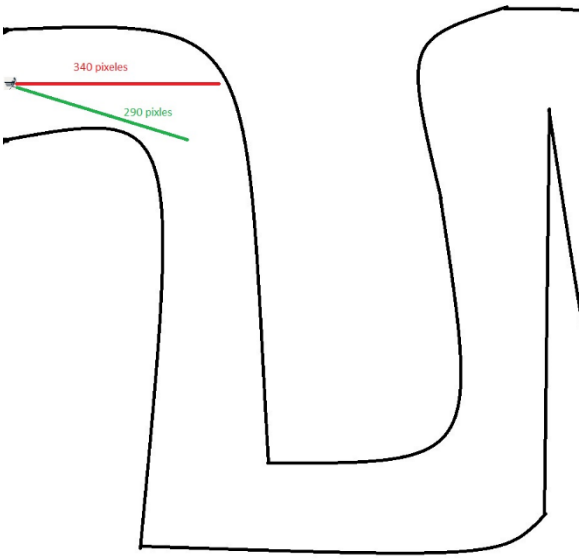
Su roadrunner va a salir de un punto de partida a la izquierda de una pista, y debe llegar a su punto de llegada al final a la derecha de la pista, esto recorriendo un camino preestablecido. Para ello se le ha proporcionado un archivo de la pista de 1800x904, el cual, tiene debidamente marcada la pista desde el punto origen hasta el destino:



El roadrunner que mide 20x20 pixels, debe salir desde el centro al inicio de la pista, hasta la salida de la pista al extremo derecho en el menor tiempo posible.

El roadrunner cuando corre tiene una visibilidad de 180 grados desde su centro de visión, de 0 a 180. Le velocidad del correcaminos es de 20 pixeles en 500ms (milisegundos), y el tamaño del correcaminos es de 20x20 pixeles.

El roadrunner que a usted se le asignado crear sigue una lista de instrucciones estándar, por ejemplo:



Si el correcaminos inicia de frente entonces no debe girar, 0 grados de giro; entonces tiene una línea libre de 340 pixeles antes de pegar con el límite de la pista. Es decir que una vez que inicia, duraría 8500ms en pegar. Si no queremos que pegue, tendría que avanzar en dirección de 90 grados por 8000ms, es decir, 320 pixeles hasta un punto seguro que no choque con el borde de la pista. Luego de eso, podría girar 90 grados a la derecha para empezar a bajar.

Las instrucciones tienen el siguiente formato:

<grado de dirección>,<tiempo en milisegundos>

90,8000 // de mi centro de vision se ubica en el grado 90 y avanza por 8000ms a la velocidad de 20pixeles/0.5 segundos.

En el caso de la línea verde, el correcaminos toma una mejor decisión. Si el correcaminos apunta a los 108 grados y avanza durante 7250ms en esa dirección, no chocaría y ya va más avanzado en la pista hacia la salida. Esa instrucción sería:
108,7250

Nótese que no hay espacios antes ni después de la coma ",". Cada instrucción para el correcaminos se separa por cambio de línea y son dadas por medio de un archivo de texto.

Para averiguar el conjunto o lista de instrucciones óptimas para hacer que el roadrunner vaya del punto de salida al punto de llegada en el menor tiempo posible, el estudiante va a diseñar un algoritmo genético de punto flotante.

Por la naturaleza y el margen de error de un GA de punto flotante, el cual depende mucho de la precisión de la representación cromosómica en el número de punto de flotante, el cruce y otras operaciones; además de lo randomizado y a veces probabilista que puede volverse un GA, lo ideal es poder ensayar múltiples aproximaciones del GA para finalmente obtener lo que se considera la respuesta óptima.

El estudiante diseñará el GA de tal forma que posea suficientes parámetros para configurar su comportamiento, por ejemplo, configurarle parámetros que inciden directamente en la función de fitness, tamaño de población, población inicial, generaciones, tipo de cruce, mutación, distribución de tendencias u otro.

Para que el estudiante no escriba todos los posibles ensayos para tratar de obtener el mejor resultado del genético; implementará un algoritmo voraz o de programación dinámica que genere un matroid M=(S,I) que cumpla lo siguiente:

- S es la lista de configuraciones con las que se va a probar una corrida del GA, siendo esas configuraciones de K variables
- I es igual a { } conjunto vacío ó
- x e I, donde x es la lista de instrucciones producidas por una corrida del GA con una configuración en S que ha obtenido el menor tiempo en llegar a la meta

Para poder obtener un resultado rápido, el matroid deberá ejecutarse a modo de algoritmo paralelo usando openMP, el cuál debe ajustar la carga de work and Depth de forma dinámica según el procesador donde se calcule el matroid.

Al terminar la ejecución en paralelo del matroid M, se tendrá la mejor lista de instrucciones calculada por su GA y bajo las configuraciones dadas, dicha lista de instrucciones deberá escribirse en un archivo llamado run_<fecha>_<su nombre>.txt

La cantidad de configuraciones de M son a criterio del estudiante según la forma en que diseñó ya sea el voraz o el algoritmo dinámico que genera las configuraciones para las corridas de prueba en M.

Otros aspectos

- El proyecto es individual
- Se debe desarrollar en C++
- El profesor proveerá un programa para que puedan probar visualmente el recorrido obtenido en su algoritmo
- Fecha límite para consulta viernes 31 de Julio
- Fecha de revisión final el 12 y 14 de agosto
- La revisión podrá ser en dos modalidades:
    - En caso de que el proyecto no se termine a cabalidad, se revisarán los algoritmos completados y su correctitud
    - Si se completa a cabalidad con una nota mínima de 80, entrará en una competencia con otro compañero designado por el profesor. El resultado que logre llevar al roadrunner más rápido a la meta ganará inmediatamente 10 puntos extras. Adicionalmente entrarán en un ranking global de tiempos de

carrera y ganarán 3 puntos extras por cada competidor al que supere

- Cualquier sospecha de copia anulará por completo todo el trabajo obteniendo una nota de cero