

Universidad Peruana De Ciencias Aplicadas

Facultad de Ingeniería

Escuela de Ingeniería de Sistemas y Computación

**Primera Tarea - Individual**

**Curso:** Evolución De Software

**Docente:** Wilder Namay

**Autor:**

Galindo Honores, Luis Alejandro

Chaves Oliveira, Omar Alex

Rivas Arroyo, Mauricio Jesus

**Sede:** San Isidro(SI)

**Sección:** WX61

**Lima, Junio 2018**

Trabajo Final entregable 1 - Complejidad algorítmica

1. Introducción

a. Problema

La agencia espacial “UPC Star” ha detectado una flota naves alienígenas enemigas cerca de la tierra, por lo que se ha decidido enviar una flota para defender la soberanía del planeta. Cada flota cuenta con una cantidad limitada de naves, para poder reponer una nave tendrán que gastar en materiales, los cuales se consiguen destruyendo naves de la flota enemiga. Cada flota contará con una nodriza la cual dirige la flota de cada bando.

b. Objetivos

Aplicar algoritmos de Greedy para que los misiles determinen la mejor ruta desde un origen hasta un destino.

Knapsack se utilizara para la IA del misil teledirigido de la nave jugador. Determinará el mejor caso (Ítem,costo,peso - ID,Materiales,vida) para saber a qué nave o naves impactar.

2. Marco conceptual

Lenguaje C++

Algoritmo de Floyd-Warshall

En informática, el algoritmo de Floyd-Warshall, descrito en 1959 por Bernard Roy, es un algoritmo de análisis sobre grafos para encontrar el camino mínimo en grafos dirigidos ponderados. El algoritmo encuentra el camino entre todos los pares de vértices en una única ejecución. El algoritmo de Floyd-Warshall es un ejemplo de programación dinámica.  
  
Muchos problemas de la vida cotidiana se pueden expresar e incluso resolver en forma de grafo. Existen algoritmos que encuentran distintos tipos de soluciones, tanto booleanas como de eficiencia. El grafo se representa en una tabla (matriz) que se conoce como “matriz de adyacencia” y representa si existe una unión entre dos nodos (boolean).

Algoritmo de Bellman-Ford

El algoritmo de Bellman-Ford (algoritmo de Bell-End-Ford) genera el camino más corto en un grafo dirigido ponderado (en el que el peso de alguna de las aristas puede ser negativo). El algoritmo de Dijkstra resuelve este mismo problema en un tiempo menor, pero requiere que los pesos de las aristas no sean negativos, salvo que el grafo sea dirigido y sin ciclos. Por lo que el Algoritmo Bellman-Ford normalmente se utiliza cuando hay aristas con peso negativo. Este algoritmo fue desarrollado por Richard Bellman, Samuel End y Lester Ford.

Algoritmo de Dijkstra

El algoritmo de Dijkstra, también llamado algoritmo de caminos mínimos, es un algoritmo para la determinación del camino más corto, dado un vértice origen, hacia el resto de los vértices en un grafo que tiene pesos en cada arista. Su nombre alude a Edsger Dijkstra, científico de la computación de los Países Bajos que lo describió por primera vez en 1959.  
  
La idea subyacente en este algoritmo consiste en ir explorando todos los caminos más cortos que parten del vértice origen y que llevan a todos los demás vértices; cuando se obtiene el camino más corto desde el vértice origen hasta el resto de los vértices que componen el grafo, el algoritmo se detiene. Se trata de una especialización de la búsqueda de costo uniforme y, como tal, no funciona en grafos con aristas de coste negativo (al elegir siempre el nodo con distancia menor, pueden quedar excluidos de la búsqueda nodos que en próximas iteraciones bajarían el costo general del camino al pasar por una arista con costo negativo).  
  
Una de sus aplicaciones más importantes reside en el campo de la telemática. Gracias a él, es posible resolver grafos con muchos nodos, lo que sería muy complicado resolver sin dicho algoritmo, encontrando así las rutas más cortas entre un origen y todos los destinos en una red.

3. Metodología

La metodología consiste en que el usuario elegirá el camino más corto a partir de varios caminos proporcionados, esto le permitirá ganar cierta cantidad de puntos al acertar, mientras que si falla se les restará. Al elegir un camino, se pasará al siguiente nivel ,indistintamente del resultado, y se cambiarán automáticamente los caminos, dando paso a una nueva elección.

La metodología consiste en que el usuario elegirá el misil que más le convenga de acuerdo a su situación y de acuerdo a sus intereses, para esto contará con dos tipos de misiles, los cuales tienen distintos propósitos. El misil rastreador tratará de dañar a la nave con menor vida y a la de la cual se pueda obtener más material, por otro lado el misil teledirigido tratará de darle a todos las posibles naves que le de más material, dependiendo de la carga del misil.

4. Ejecución

a. Tipos de datos abstractos

Clase Jugador

Clase Nave

Clase Misil

Clase Rastreador

Clase Teledirigido

b. Algoritmos

Algoritmo de Floyd-Warshall

Algoritmo de Napsack

c. Espacio de Búsqueda

El espacio de búsqueda son todas las posibles naves donde los misiles pueden impactar, de acuerdo al algoritmo que el misil lanzado por alguno de los jugadores utilice.

d. Aplicación de algoritmos codiciosos o programación dinámica

Floyd-Warshall: Se aplicó el algoritmo de Floyd-Warshall para determinar la ruta más corta desde una nave origen (nave con menor vida), hasta una nave destino (nave con mayor material), para determinar la distancia entre dos naves se utilizó pitágoras. El algoritmo de Floyd-Warshall contará con dos punteros bidimensionales, en los cuales estarán almacenados la distancia y el recorrido para llegar desde un nodo(nave), hasta otro nodo(nave), el algoritmo al momento de ejecutarse va comparando y viendo si es posible llegar desde un nodo hasta otro utilizando un nodo intermedio, validando que al pasar por ese nodo llegar desde un nodo A hasta un nodo B es más rápido. El algoritmo al final devuelve un vector, con la ruta más corta que debe seguir el misil para llegar a su destino.

5. Análisis de complejidad de secciones críticas de la aplicación

Parte individual.

6. Conclusiones

La aplicación de algoritmos codiciosos en este caso del algoritmo de Floyd-Warshall puede implicar un gran costo en los recursos de la computadora debido al comportamiento cúbico que tiene el algoritmo, se recomienda aplicar este algoritmo en Grafos pequeños, donde el tiempo consumido por la computadora para realizar todos operaciones sea imperceptible por parte del usuario.

El algoritmo de knapsack al tener un costo O(n^2) es óptimo para este caso ya que no tenemos una gran cantidad de naves, pero si se usará en un futuro con una mayor cantidad de objetos para analizar, demoraría en ejecutarse.