Práctica 2:Divide Y Vencerás

La idea de esta Práctica es usar 4 tipos diferentes de ordenación para ordenar una lista de MapNodes que representan los cuadrados de un juego del estilo Tower Defense para aprovecharnos de estos algoritmos y tratar de ganar salvando así La Universidad de la sabiduría.

Los 4 algoritmos de ordenación que hemos usado son NoSort consistente en buscar las x casillas más rentables sin ningun tipo de orden, el segundo es InsertionSort, es decir ordenación por inserción, el tercero es MergeSort, es decir ordenación por mezcla y finalmente QuickSort u ordenación rápida.

He ejecutado el código unas cuantas veces para comprobar los resultados que se obtienen y poder ver cual es el algoritmo que mas nos rentaría usar. Las siguientes tablas muestran el tamaño del mapa, y el tiempo que se tarda en calcular las casillas mas eficientes con cada algoritmo.

Datos

Prueba 1

Tamaño del Mapa (n)	NoSort	InsertionSort	MergeSort	QuickSort
496	0.0445	0.4782	0.0937	0.0358
946	0.0803	1.5974	0.1858	0.0877
1540	0.1331	3.9449	0.3295	0.1684
2278	0.1890	8.5897	0.5068	0.3311
3160	0.2797	17.1356	0.7491	0.5714

Prueba 2

Tamaño del Mapa (n)	NoSort	InsertionSort	MergeSort	QuickSort
496	0.0452	0.4859	0.0913	0.0369
946	0.0788	1.5314	0.1853	0.0861
1540	0.1292	3.9565	0.3313	0.1669

2278	0.1899	8.4454	0.5336	0.3338
3160	0.2769	16.4754	0.7645	0.5698

Prueba 3

Tamaño del Mapa (n)	NoSort	InsertionSort	MergeSort	QuickSort
496	0.0453	0.5076	0.0877	0.0367
946	0.0829	1.5576	0.1746	0.0853
1540	0.1288	3.8949	0.3188	0.1709
2278	0.1898	8.5556	0.5079	0.3291
3160	0.2658	16.9000	0.7358	0.5605

Análisis de los resultados

Como puedo apreciar en las tablas, surgen unos resultados inesperados ya que en principio el algoritmo más eficiente debería de ser el MergeSort o el QuickSort pero sin embargo el que saca mejores tiempos es el NoSort, esto de primeras puede dar a entender que nos deberíamos de quedar si o si con el NoSort pero en caso de que el mapa creciese y tuviésemos un n más grande seguramente el QuickSort llegaría un momento en el que se quedaría más o menos estable debido a su O(nlog(n)) y sin embargo como el otro tiene complejidad O(k*n) llegará un punto en el que lo superará.

En resumen si el mapa se va quedar de este tamaño y con el mismo numero de torretas eligiría usar en NoSort, sin embargo en caso de que alguno de estos dos parámetros creciese según avancen los niveles optaría por el QuickSort que va a ser mucho mas consistente y no va a subir tanto de complejidad.