UNIVERSITAT DE LLEIDA

Escuela Politécnica Superior

Grado en Ingeniería Informática

Inteligencia artificial

Primera práctica de Inteligencia Artificial

Alumnos:

Carolina Romeu Farré

Pol Llagostera Blasco

Data: 24 de noviembre del 2012

Curso 2012-2013

Decisiones

Para implementar el algoritmo BFS utilizamos como referencia el DFS, ya que el código era el mismo exceptuando el tipo de cola, de LIFO (DFS) a FIFO (BFS).

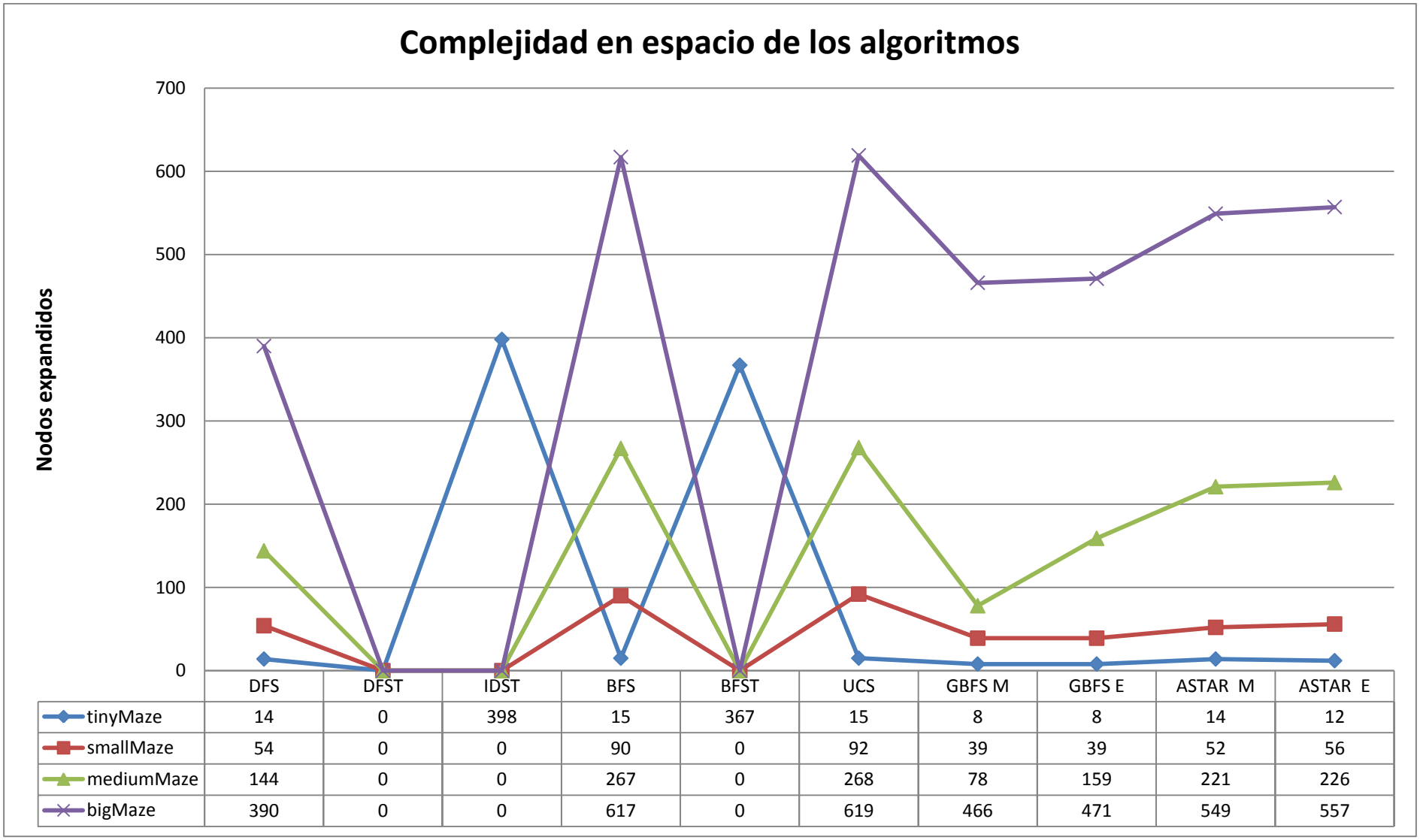
Para implementar el BFS i el DFS en árbol, utilizamos los mismos algoritmos pero quitando el diccionario de *expanded*, ya que en árbol no se puede recordar el recorrido.

Llegados a este punto, nos dimos cuenta de que utilizábamos código casi idéntico, por tanto creamos dos funciones llamadas *blind* y *blindTree* donde pusimos el código genérico para BFS y DFS (lo mismo para el árbol) y desde su función llamábamos al tipo de cola.

En implementación del IDS en árbol, tuvimos algunos problemas al elaborar el código del DLS, en especial en la parte del *cutoff*, ya que teníamos que jugar con las dos funciones en el paso de parámetros y retornos.

El UCS y el A*, el código es casi el mismo, con la diferencia de que en el A* en la prioridad del *fringe* sumamos el coste mas la heurística del nodo y en el UCS solo se prioriza con el coste. Después miramos en el *fringe* cuando haya un nodo repetido con coste mayor que el actual, si es así lo introducimos con la nueva prioridad al *fringe.*

En el voraz, la diferencia con el BFS y DFS es que cambias el tipo de cola del *fringe* por una con prioridad, e introduces los nodos con prioridad heurística.



	DFS	DFST	IDST	BFS	BFST	UCS	GBFS M	GBFS E	ASTAR M	ASTAR E
tinyMaze	14	0	398	15	367	15	8	8	14	12
smallMaze	54	0	0	90	0	92	39	39	52	56
mediumMaze	144	0	0	267	0	268	78	159	221	226
bigMaze	390	0	0	617	0	619	466	471	549	557

Análisis de la gráfica

Los resultados de la gráfica nos muestran que los algoritmos en árbol son muy ineficientes, eso es debido a que no recuerdan los nodos ya expandidos y por tanto han de empezar desde el principio explorando de nuevo las ramas. Por eso tienen una complejidad en espacio y tiempo superiores a todos los demás, pudiendo tardar días en encontrar una solución. En la gráfica no han sido calculados por la larga espera.

El caso donde esto se muestra claramente es en el DFST (DFS en árbol). En este caso nunca se encuentra ningún camino hacia el nodo objetivo ya que al no recordar los nodos por donde ha pasado, se queda trabado entre dos nodos de la primera rama que explora. Ej. si A es el nodo padre y B uno de sus sucesores, se quedaría atrapado haciendo un recorrido infinito de A-->B-->A-->B-->A-->B...

Como podemos ver, entre las búsquedas no informadas, el mejor algoritmo es el DFS por lo que hace a número de nodos expandidos, pero no es óptimo, el BFS por su contra si es óptimo, ya que los costes entre nodos son idénticos, aunque tenga que expandir muchos más nodos. Teniendo como contrapunto los algoritmos UCS y BFS. Esto es así debido a que el UCS expande todos los nodos para escoger la mejor ruta mínima hacia el objetivo (no expandirá los nodos que sean superiores del coste del nodo objetivo) y el BFS expandirá todos los nodos en cada nivel, desde los más superficiales e irá pasando por cada nivel de profundidad. Así pues, vemos que las búsquedas en profundidad son las mejores para explorar rutas hacia un objetivo en este tipo de problemas (PAC-MAN) si importa más la complejidad de espacio y de tiempo que encontrar el camino más corto.

En las búsquedas informadas, notamos que el algoritmo voraz (GBFS) es el que menos nodos expande. Eso es así gracias a que todos los costes entre nodos son igual a uno, de esta forma consigue una buena heurística y se convierte en el algoritmo más eficiente en complejidad en espacio y tiempo, por el contrario nunca podrá a llegar a ser eficiente ya que no es óptimo. Si fuera diferente, con unos costes variables, el algoritmo A* podría ser el más adecuado ya que compensa la complejidad con la optimalidad, es decir, la complejidad en el tiempo y espacio no sería excesiva y siempre seria optimal También podemos notar que la heurística Manhattan es la que da las heurísticas más admisibles, de esta forma la complejidad disminuye y decrementa el numero de nodos expandidos.

En el IDS en grafo la expansión de nodos era demasiado superior a la media de los otros algoritmos, por eso hemos hecho otro gráfico donde puede verse mejor. Aunque la complejidad en espacio sea mayor, encuentra la solución óptima.