# IA - Proyecto 1

MANUEL ALEJANDRO MARTINEZ FLORES

## 1. Resultados

Se realizó la implementación de los algoritmos de búsqueda BFS, DFS, Greedy Best First Search y A* para la resolución de laberintos. Para evaluar cada algoritmo, se calculan el costo final de la solución, la cantidad de nodos visitados, el tiempo de ejecución y el branching factor. A continuación se muestra un ejemplo visual de una solución del Laberinto 3 usando A*. El área sombreada corresponde a los nodos visitados

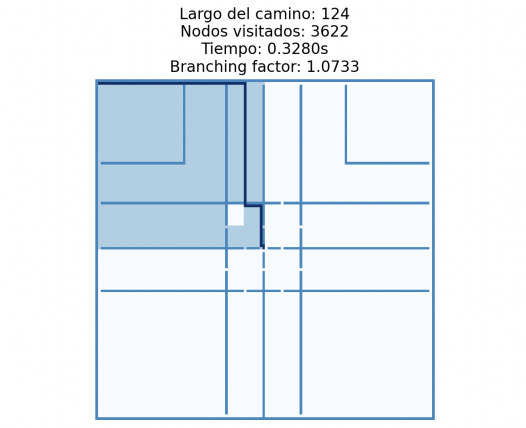


Figura 1: Ejemplo de visualización A*.

Cabe destacar que para el cálculo del branching factor se supuso que el árbol era uniforme, por lo que $N = 1 + \cdots + b^d$ donde $N$ es la cantidad de nodos visitados y $d$ la profundidad del árbol. Primero se tomaron los casos base proporcionados. Estos resultados se muestran en el Cuadro 1.

| Algoritmo | Heurística | Laberinto | Costo | No. visitados | Tiempo (s) | Branching f. |
|---|---|---|---|---|---|---|
| A* | euclidean | 1 | 250 | 15875 | 1.6079 | 1.0238 |
| | | 2 | 250 | 13733 | 1.4927 | 1.0230 |
| | | 3 | 124 | 3655 | 0.1751 | 1.0404 |
| | manhatan | 1 | 250 | 15875 | 3.3201 | 1.0238 |
| | | 2 | 250 | 11652 | 1.6874 | 1.0223 |
| | | 3 | 124 | 3622 | 0.3361 | 1.0403 |
| GBFS | euclidean | 1 | 250 | 250 | 0.0040 | 1.0000 |
| | | 2 | 280 | 430 | 0.0076 | 1.0027 |
| | | 3 | 270 | 1703 | 0.0473 | 1.0110 |
| | manhatan | 1 | 250 | 250 | 0.0029 | 1.0000 |
| | | 2 | 274 | 340 | 0.0048 | 1.0015 |
| | | 3 | 136 | 153 | 0.0017 | 1.0015 |
| BFS | - | 1 | 250 | 15875 | 0.1564 | 1.0238 |
| | | 2 | 250 | 15506 | 0.1204 | 1.0237 |
| | | 3 | 124 | 7343 | 0.0579 | 1.0476 |
| DFS | - | 1 | 250 | 250 | 0.0032 | 1.0000 |
| | | 2 | 6934 | 11929 | 0.1294 | 1.0001 |
| | | 3 | 3744 | 4618 | 0.0442 | 1.0001 |

Cuadro 1: Resultados de casos base

Además, se consideraron 10 puntos de partida aleatorios para cada laberinto y se evaluaron los algoritmos en cada uno de estos casos. Luego se calcularon los promedios de cada métrica y se muestran en el Cuadro 2.

| Algoritmo | Heurística | Laberinto | Costo | No. visitados | Tiempo (s) | Branching f. |
|---|---|---|---|---|---|---|
| A* | euclidean | 1 | 146.3 | 7110.4 | 1.0070 | 1.0338 |
| | | 2 | 125.4 | 3992.2 | 0.4180 | 1.0429 |
| | | 3 | 88.5 | 1930.5 | 0.1206 | 1.0538 |
| | manhatan | 1 | 146.3 | 5047.2 | 0.6926 | 1.0309 |
| | | 2 | 125.4 | 2379.0 | 0.1794 | 1.0370 |
| | | 3 | 88.5 | 1528.6 | 0.1017 | 1.0483 |
| GBFS | euclidean | 1 | 146.3 | 146.3 | 0.0023 | 1.0000 |
| | | 2 | 133.0 | 194.2 | 0.0033 | 1.0034 |
| | | 3 | 117.5 | 936.8 | 0.0369 | 1.0136 |
| | manhatan | 1 | 146.3 | 146.3 | 0.0019 | 1.0000 |
| | | 2 | 145.0 | 225.9 | 0.0035 | 1.0053 |
| | | 3 | 121.3 | 734.2 | 0.0330 | 1.0143 |
| BFS | - | 1 | 146.3 | 13206.8 | 0.1204 | 1.0486 |
| | | 2 | 125.4 | 11512.5 | 0.0970 | 1.0572 |
| | | 3 | 88.5 | 3624.9 | 0.0302 | 1.0670 |
| DFS | - | 1 | 252.7 | 252.7 | 0.0022 | 1.0000 |
| | | 2 | 4422.4 | 11560.3 | 0.1146 | 1.0007 |
| | | 3 | 6246.3 | 10027.0 | 0.1158 | 1.0001 |

Cuadro 2: Resultados casos aleatorios (promedio)

Esta información se puede resumir aún más, promediando los resultados de todos los laberintos. Esto se muestra en el siguiente cuadro.

| Algoritmo | Heurística | Costo | No. visitados | Tiempo (s) | Branching f. |
|---|---|---|---|---|---|
| A* | euclidean | 120.0667 | 4344.3667 | 0.5152 | 1.0435 |
| | manhatan | 120.0667 | 2984.9333 | 0.3246 | 1.0387 |
| GBFS | euclidean | 132.2667 | 425.7667 | 0.0142 | 1.0057 |
| | manhatan | 137.5333 | 368.8000 | 0.0128 | 1.0065 |
| BFS | - | 120.0667 | 9448.0667 | 0.0825 | 1.0576 |
| DFS | - | 3640.4667 | 7280.0000 | 0.0775 | 1.0003 |

Cuadro 3: Resumen de casos aleatorios

# 2. Discusión

La implementación de cada algoritmo es bastante similar. La principal diferencia yace en la forma en la que se maneja la estructura de cola. Para el algoritmo BFS es un FIFO, para DFS es un LIFO, y para A* y GBFS es una cola de prioridad. Esto permitió utilizar un factory de cola con las condiciones anteriormente mencionadas y se pudo reutilizar el código del algorimto de búsqueda principal. De todas las implementaciones de cola, la que presenta el mayor reto es la cola de prioridad, en este caso se implemento con una búsqueda lineal para añadir elementos. Esto puede optimizarse utilizando estructuras óptimas para el mantener el orden de elementos como heaps o BST.

Para este experimento, los únicos algoritmos que encontraron soluciones óptimas en cada iteración fueron A* y BFS. El primero garantiza solución óptima cuando la heuristica es admisible, i.e. no sobreestima el costo de trasladarse de un nodo a otro, lo cual ocurre para ambas heuristicas consideradas aquí. Ahora, en este experimento, el costo de la trayectoria equivalía a la profundidad del árbol solución. Por lo que BFS asegura explorar todos los caminos de cierta profundidad antes de considerar los caminos de profundidad mayor, obteniendo soluciones óptimas en este caso.

En el Cuadro 3, se presentan el resumen de las métricas para cada algoritmo con su heurística. La primer métrica es el costo final de la solución, un menor costo es más favorable. Esta métrica nos permite identificar los algoritmos que encuentran soluciones óptimas y nos permite medir cuanto sacrificamos en el costo para obtener una mayor simplicidad o rapidez. Como se menciono anteriormente, A* y BFS obtienen soluciones óptimas en este caso, mientras que DFS encuentra soluciones sin importarle el costo de la solución, incurriendo en costos bastante elevados. La métrica que cuenta la cantidad de nodos visitados puede interpretarse como el costo del algoritmo

en memoria. Estos nodos deben mantenerse en memoria para poder reconstruir el camino una vez encontrada la solución. El algoritmo que menos nodos visita es el GBFS, debido a su naturaleza greedy, mientras que DFS y BFs visitan una gran cantidad de nodos debido a que no son informados. El tiempo de ejecución también es importante, GBFS es el más rápido, esto relacionado a que es el que menos nodos visita. DFS y BFS presentan tiempos similares a pesar de visitar una gran cantidad de nodos. El más lento es A* debido a que visita una cantidad mediana de nodos, pero el proceso de añadir a la cola es más lento en una cola de prioridad. Posibles soluciones a esto se discutieron anteriormente. El branching factor indica la cantidad de hijos promedio que cada nodo tiene en el árbol solución. Este valor se vuelve 1 cuando la heurística coincide perfectamente con el costo. Se puede ver que en este caso el branching factor más alto es el de BFS, debido a que recorre todos los hijos de los nodos antes de pasar a la siguiente profundidad. Luego se encuentran los algoritmos A*, que visitan una cantidad mediana de nodos. En este caso, se ve una notoria mejora entre la heurística euclideana y la manhatan, debido a que la segunda coincide con la cantidad de movimientos sin restricciones. Esto no ocurre en GBFS, debido a la naturaleza greedy del algoritmo. El menor branching factor corresponde a DFS, debido a que se expande de manera vertical, visitando casi un hijo por nodo.

El algoritmo DFS se desempeña mejor cuando los nodos que se instancian primero son los más probables de ser ótpimos. Esto se puede ver en su desempeño del laberinto 1: en este caso sin restricciones todos los nodos que visitó formaron parte de la solución y esta fue óptima. Sin embargo, cuando enfrenta alguna restricción puede encontrar soluciones con un costo final muy elevado. El algoritmo BFS funciona bien cuando el costo coincide con la longitud de la solución, como fue este caso. Sin embargo, no es recomendable cuando el instanciar nodos sea demasiado costoso. El algoritmo GBFS funciona bien cuando se necesita una solución aceptable (no necesariamente óptima) con considerable rapidez. No obstante, es necesario que tenga una heurística lo suficientemente informativa y que no se le presenten muchos obstáculos. El algoritmo A* funciona bien cuando encontrar solución óptima es escencial y esto compensa el costo de tiempo o computo.

# 3. Código

Link de repositorio GitHub