**PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ**



**Tarea Académica de Aplicaciones de Ciencias de la Computación**

**Profesores:**

* Oncevay Marcos, César
* Villanueva Talavera, Edwin
* Beltrán Castañón, César

**Horario: 0781**

**Integrantes:**

* ESPEZÚA ZAPANA, Bruno Víctor 20151258
* ALFARO RAMOS, Jorge Gabriel 20151521
* RAMIREZ BENITES, Martín Enrique 20150178

ÍNDICE

[**Descripción de la representación de estados**](#_dsv9t9uv9fk1) **4**

[**Descripción de la heurística**](#_2ln3my5odymq) **4**

[**Tabla comparativa**](#_wowset8lgu8c) **5**

[**Análisis de los resultados**](#_qr4xive3c72m) **6**

[**Conclusiones**](#_6vv2x6pg8r13) **6**

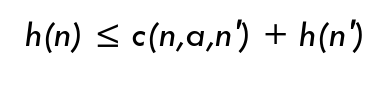
### Descripción de la representación de estados

Para representar los estados se definió una lista que contiene dos elementos, el primero, la tupla (x, y) de la posición inicial, y el segundo, una lista de las posiciones de las cuatro esquinas por visitar, que solo se irá modificando a medida que sean visitados las esquinas, para poder controlar que el pacman ya haya visitado todas al final. En el caso de las expansiones, se usará una lista auxiliar que obtenga la las tuplas de las posiciones de las esquinas restantes, sacadas del segundo elemento de la lista de estados, en cada expansión.

### Descripción de la heurística

Para este problema se escogió una heurística que encuentra la distancia Manhattan desde la posición del nodo hasta una esquina no visitada, luego, la distancia mínima Manhattan desde esta esquina hacia otra no visitada y así sucesivamente hasta visitar las 4 esquinas desde la posición del sucesor, todas estas distancias mínimas se acumulan y el cornerHeuristic finalmente devuelve este resultado; luego regresará a la posición inicial y devolviendo valor de la heurística 0 durante este retorno.

Esta heurística es consistente pues siempre se cumple la ecuación para todos los nodos y sus hijos generados:



Esto se demuestra en la función de verificación para cada sucesor obtenido dentro de AstarSearch. (nodos explorados)

Además, el algoritmo es consistente ya que cada movimiento que realiza, lo acerca más a su objetivo, siempre hay sensación de progreso ya que solo influye el espacio por recorrer. Por ende, si la heurística es consistente entonces es admisible.

### Tabla comparativa

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | DFS | BFS | IDS | BS | A\* |
| MediumCorners | #Nodos en Memoria | 266 | 152 | 194 | 152 | 152 |
| #Nodos Visitados | 514 | 3031 | 241284 | 2013 | 2494 |
| Costo de la Solución | 266 | 152 | 194 | 152 | 152 |
| BigCorners | #Nodos en Memoria | 470 | 214 | 438 | 214 | 214 |
| #Nodos Visitados | 997 | 10581 | 1455084 | 6629 | 6286 |
| Costo de la Solución | 426 | 214 | 438 | 214 | 214 |

Análisis de resultados:

Como se puede apreciar en la tabla comparativa, se obtuvieron dos algoritmos con resultados muy similares, que son el de búsqueda bidireccional y el A\*, ya que el primero presenta el menor costo de solución, con un número relativamente bajo (en promedio) de nodos en memoria para el mediumCorners, esto quiere decir que no demorará tanto en encontrar la solución, al no emplear tanta memoria expandiendo nodos, pues se busca desde el punto de inicio y desde el final simultáneamente, buscando la intersección en un nodo. Sin embargo, podemos ver que a medida que el mapa se hace más grande, como en el big corners, va perdiendo su eficiencia y lo reemplaza el A\*, pues la heurística empleada, escala bien con la cantidad de datos, otorgando mayor rapidez al no expandir tantos nodos. Con esto, finalmente podemos concluir que el A\* es el algoritmo de búsqueda más eficiente para resolver estos problemas.

### Análisis de los resultados

A diferencia de las búsquedas anteriores, donde se buscaban todos los puntos y se elegía la primera coincidencia que botara el algoritmo, con esta búsqueda voraz nos enfocaremos en un solo punto (el más cercano) y a partir de eso se buscará el camino más corto a ese punto, haciéndolo más eficiente y menos costosa (si los caminos tuvieran diferentes costos)

- **Vale la pena hacer dicha búsqueda codiciosa en los layouts testados?**

No, porque como cada accion tiene costo uniforme (1) con otras busquedas también se podrá lograr el menor costo, sin tanta complejidad, por ejemplo con el BFS, el BS y el A star como se aprecia en la tabla.

- **La solución entregada será siempre la de menor costo?**

Como se mencionó anteriormente, si sería siempre la de menor costo, al buscar siempre lo más cercano, y considerando que cada desplazamiento tiene el mismo costo.

### Conclusiones

Habiendo desarrollado diversos algoritmos de búsqueda y heurísticas para este proyecto de Pacman, podemos concluir que:

1. La búsqueda iterativa no es la solución indicada, definitivamente. Esto ya que la cantidad de nodos expandidos escala rápidamente y tiene que volver a recorrer los niveles.
2. Los algoritmos más eficientes son: la búsqueda bidireccional y el A\* con la heurística corners, aunque a mayor tamaño del “mapa” se opta por un A\*, debido a la heurística empleada.
3. Para poder visualizar los resultados gráficamente, se modificó el pacman.py para que termine cuando el pacman regrese a la posición inicial, luego de recorrer las 4 esquinas, esto ayudó a comprender la naturaleza de cada algoritmo de búsqueda desarrollado, como la preferencia por el bfs e ids a dirigirse a una ruta, no siendo esta la de menor costo de camino.
4. En problemas como este, en los que hay que recorrer un tablero y existiera la posibilidad de poder retornar, se hace importante e indispensable trabajar con una estructura que almacene los puntos ya visitados.