

**UNIVERSIDAD AUTÓNOMA**

**DE BAJA CALIFORNIA**

**FIAD**

**Inteligencia Artificial**

**Practica #1**

**“PACMAN DFS,**

**PACMAN BFS y PACMAN A\*”**

**Integrantes:**

* Ricardo Villegas López

**Catedrático:**

* Hugo Armando Guillen Ramírez

Ensenada Baja California a 08 de junio de 2018

**Introducción**

En esta práctica se implementará una aplicación simple de la teoría de juegos y heurísticas de búsqueda. Se desarrollo el juego de pacman con 3 diferentes búsquedas (BFS, DFS, A\*), las cuales se mostrarán a continuación.

**Metodología**

La **búsqueda de primer orden (BFS)** es un algoritmo para atravesar o buscar estructuras de datos de árbol o gráfico. Comienza en la raíz del árbol (o en algún nodo arbitrario de un gráfico, a veces denominado 'clave de búsqueda'), y explora todos los nodos vecinos en la profundidad actual antes de pasar a los nodos en el siguiente nivel de profundidad

Utiliza la estrategia opuesta a la búsqueda en profundidad, que en su lugar explora los nodos de mayor profundidad antes de forzarse a retroceder y expandir los nodos más profundos.

Una **Búsqueda en profundidad** (en inglés **DFS o Depth First Search**) es un algoritmo de búsqueda no informada utilizado para recorrer todos los nodos de un grafo o árbol de manera ordenada, pero no uniforme. Su funcionamiento consiste en ir expandiendo todos y cada uno de los nodos que va localizando, de forma recurrente, en un camino concreto. Cuando ya no quedan más nodos que visitar en dicho camino, regresa (retrocede), de modo que repite el mismo proceso con cada uno de los hermanos del nodo ya procesado.

**Algoritmo de Búsqueda A\*.** Conocido también como A asterisco o A estrella, su función es encontrar siempre y cuando se cumplan determinadas condiciones, el camino de menor costo entre un nodo origen y uno objetivo, es la forma más ampliamente conocida de la búsqueda primero el mejor, siendo la búsqueda A\* tanto completa como óptima.

**Desarrollo de pacman con búsqueda BFS**

La finalidad de Pacman es que tiene que encontrar la comida usando la amplitud de primera búsqueda (BFS), siempre y cuando la cuadrícula es completamente observable, para saber dónde se encuentra su comida se desarrolló un BFS en la cuadrícula y luego se imprime la ruta obtenida por BFS desde el pacman a la comida.

**Desarrollo de pacman con búsqueda DFS**

En este juego, Pacman se coloca en una cuadrícula. Pacman tiene que encontrar el alimento usando la primera búsqueda de la profundidad (DFS). Asuma que la cuadrícula es completamente observable, realice un DFS en la cuadrícula y, a continuación, imprima la ruta obtenida por DFS desde el pacman a la comida.

**Desarrollo de pacman con búsqueda A\***

En el juego anterior, realizó UCS en la cuadrícula PacMan. En este juego utilizamos el algoritmo AStar para reducir los nodos expandidos utilizando la búsqueda utilizando una heurística simple pero eficiente.

AStar en gráficos usa la siguiente función

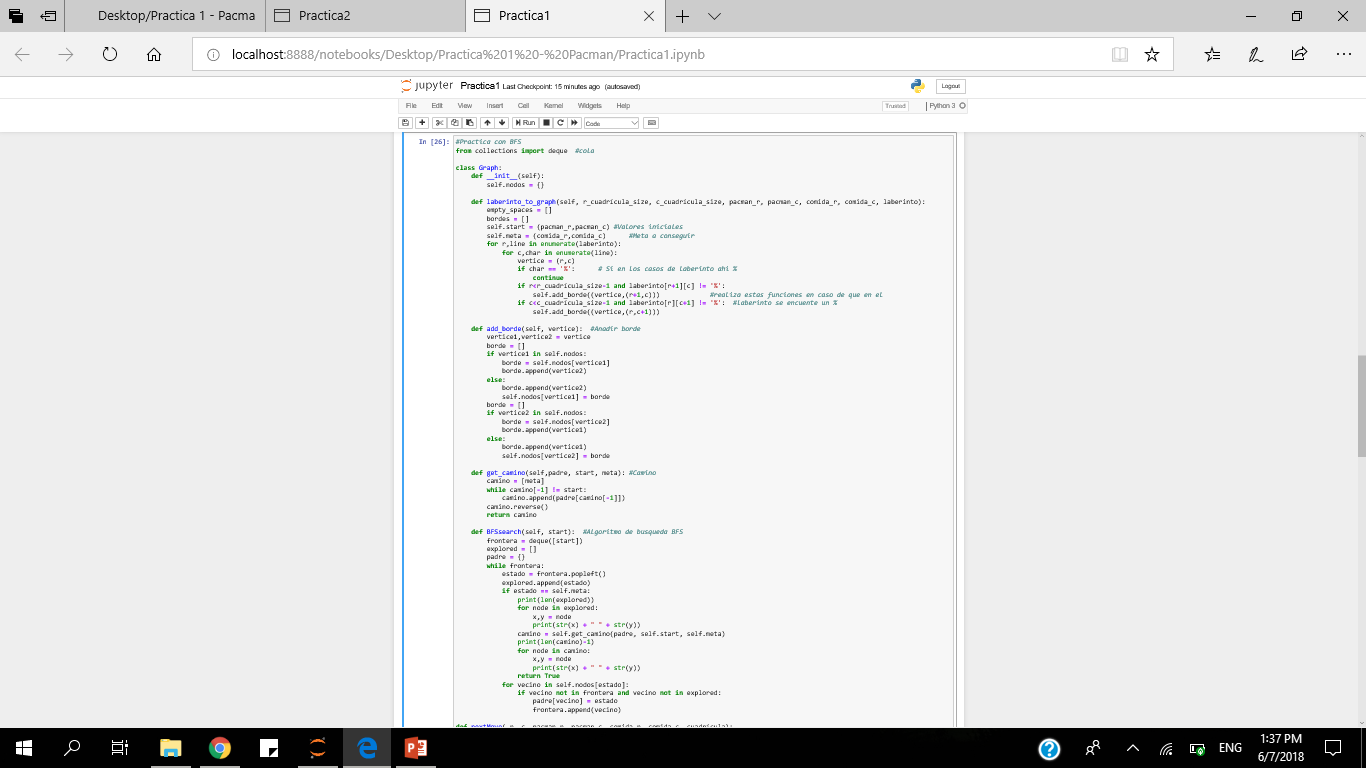
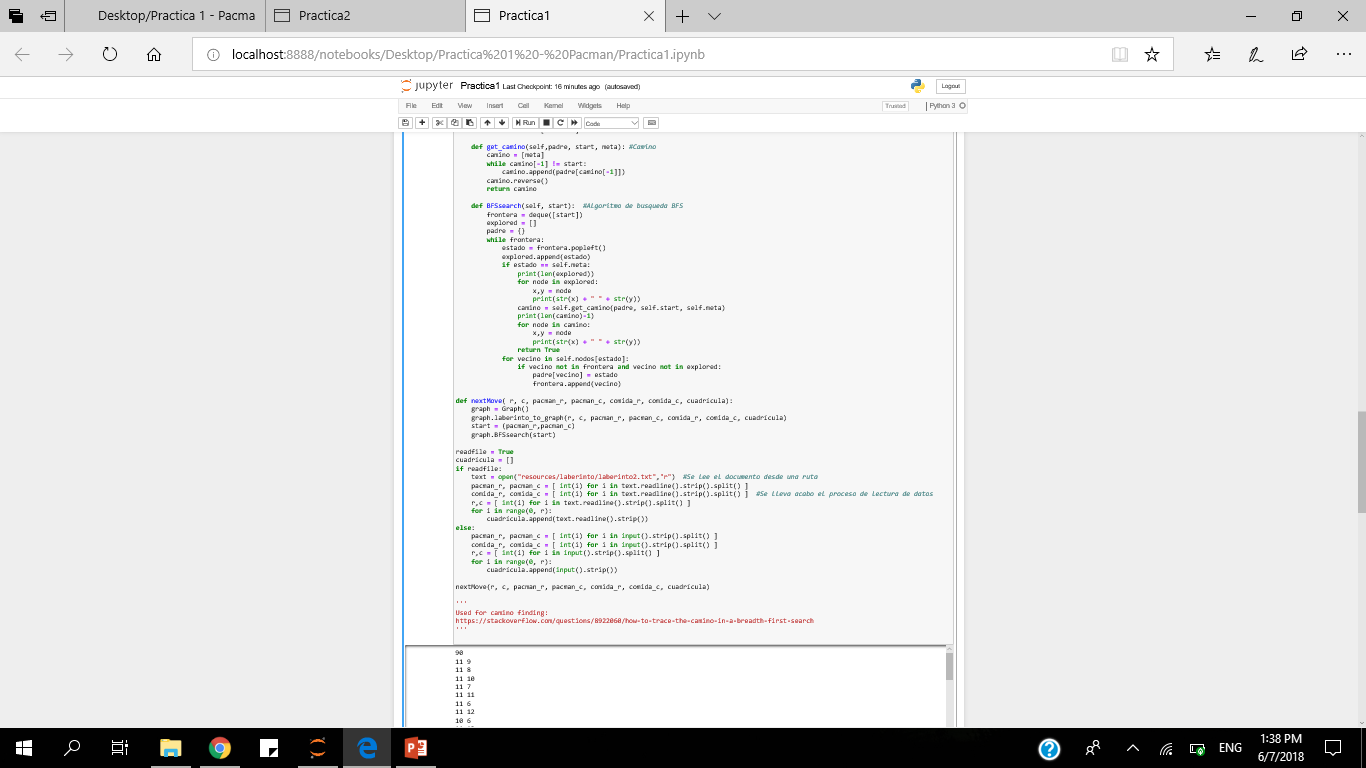
**cost = d(s,c) + h(c)**

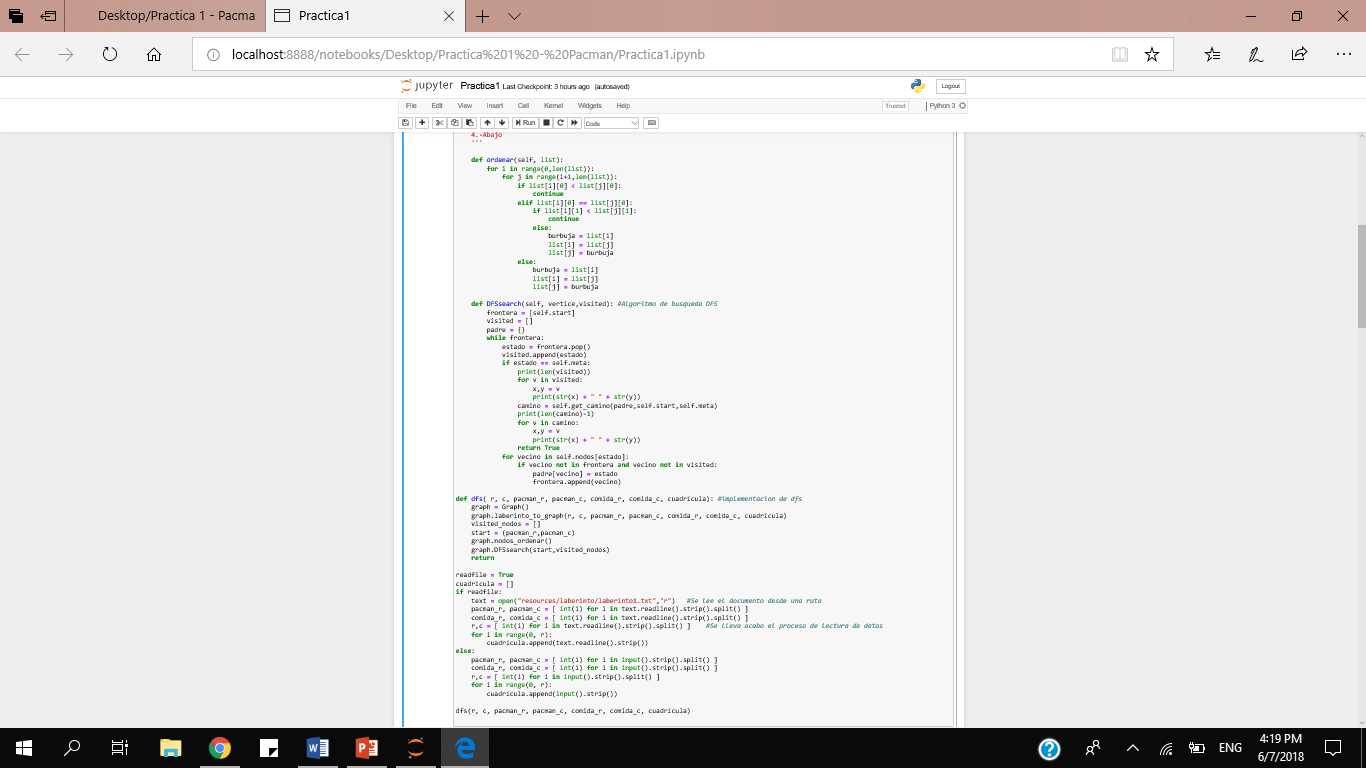
donde s es el nodo fuente, c es el nodo actualmente expandido yh (c) es la estimación del costo para llegar desde c hasta el destino (comida).

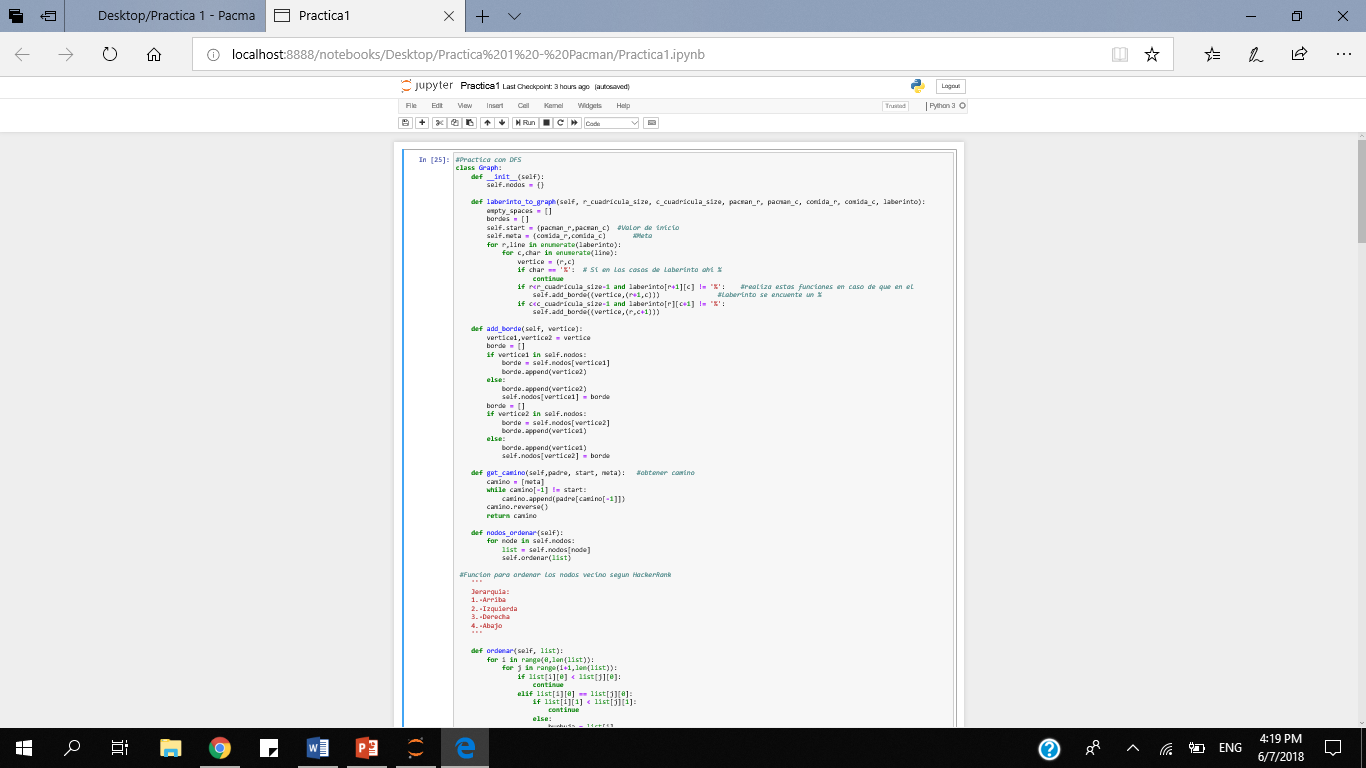
En este juego, usamos la heurística de Manhattan como estimación. Dado dos nodos (r, c) y (r1, c1). La heurística de Manhattan es la distancia de Manhattan entre los dos nodos y está dada por

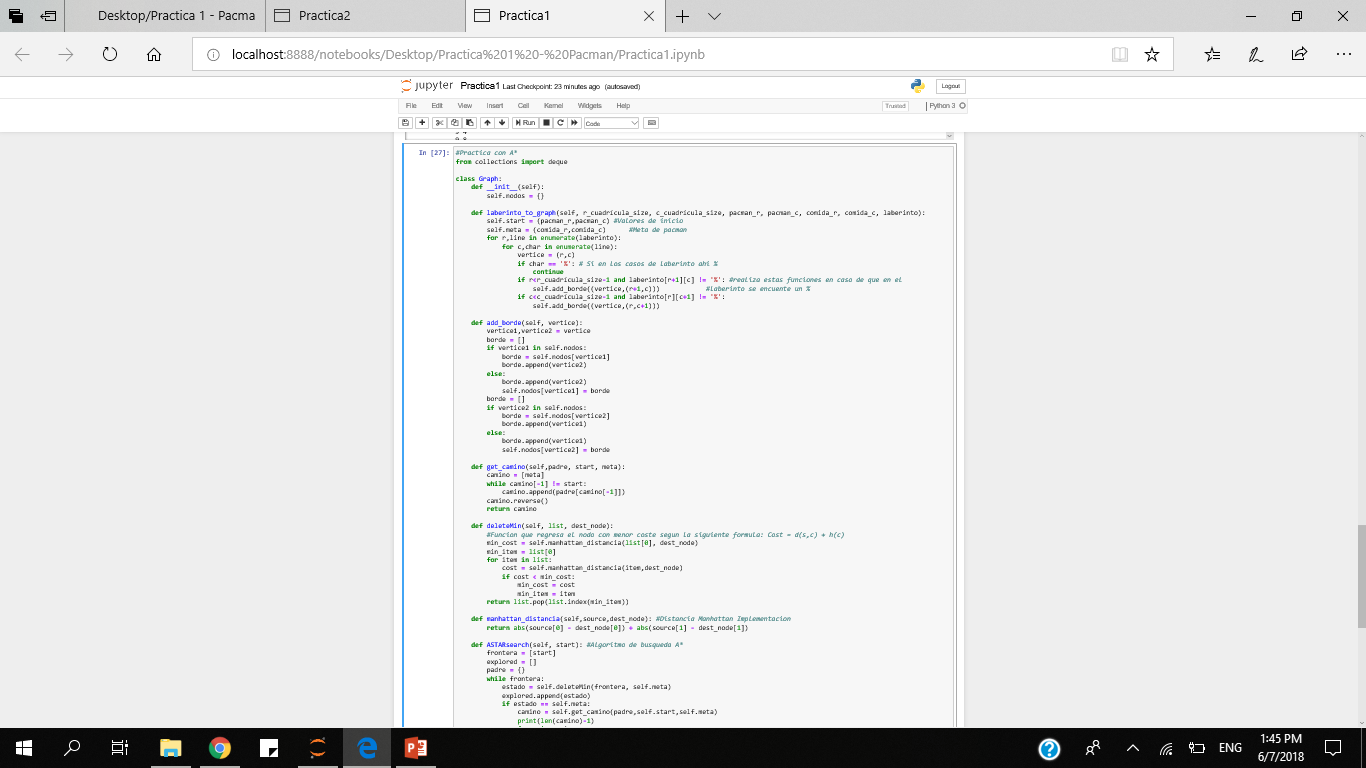
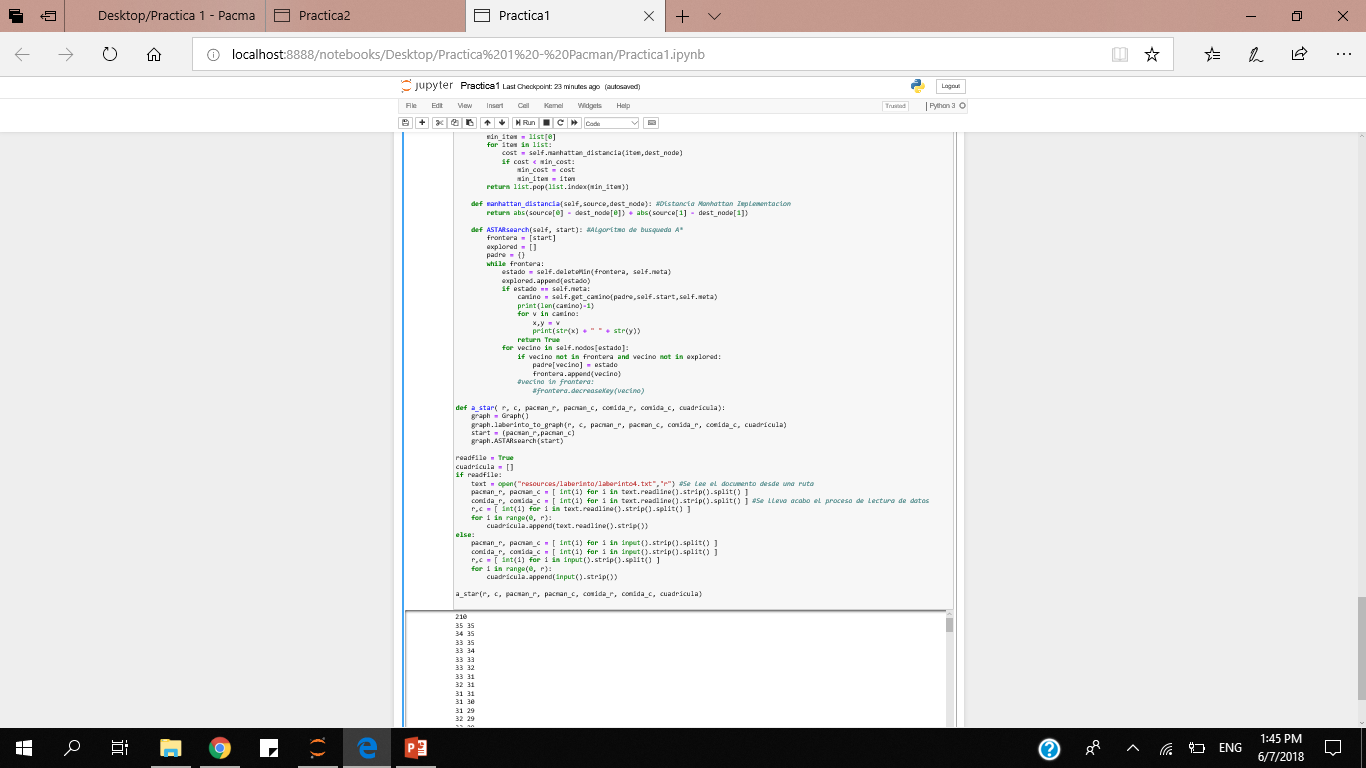
**|r1 - r | + |c1 - c|**

**Código**

**Búsqueda BFS**

**Búsqueda DFS**

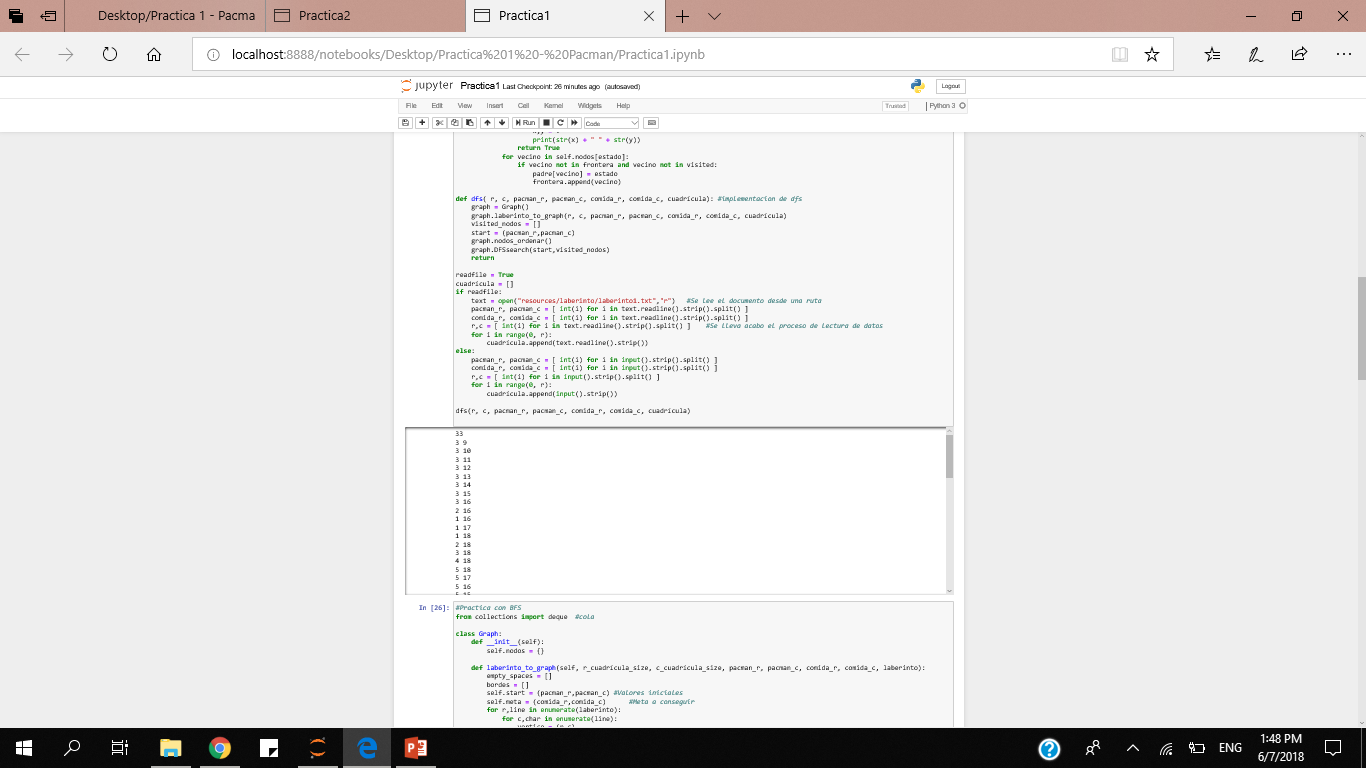
****

**Búsqueda A\***

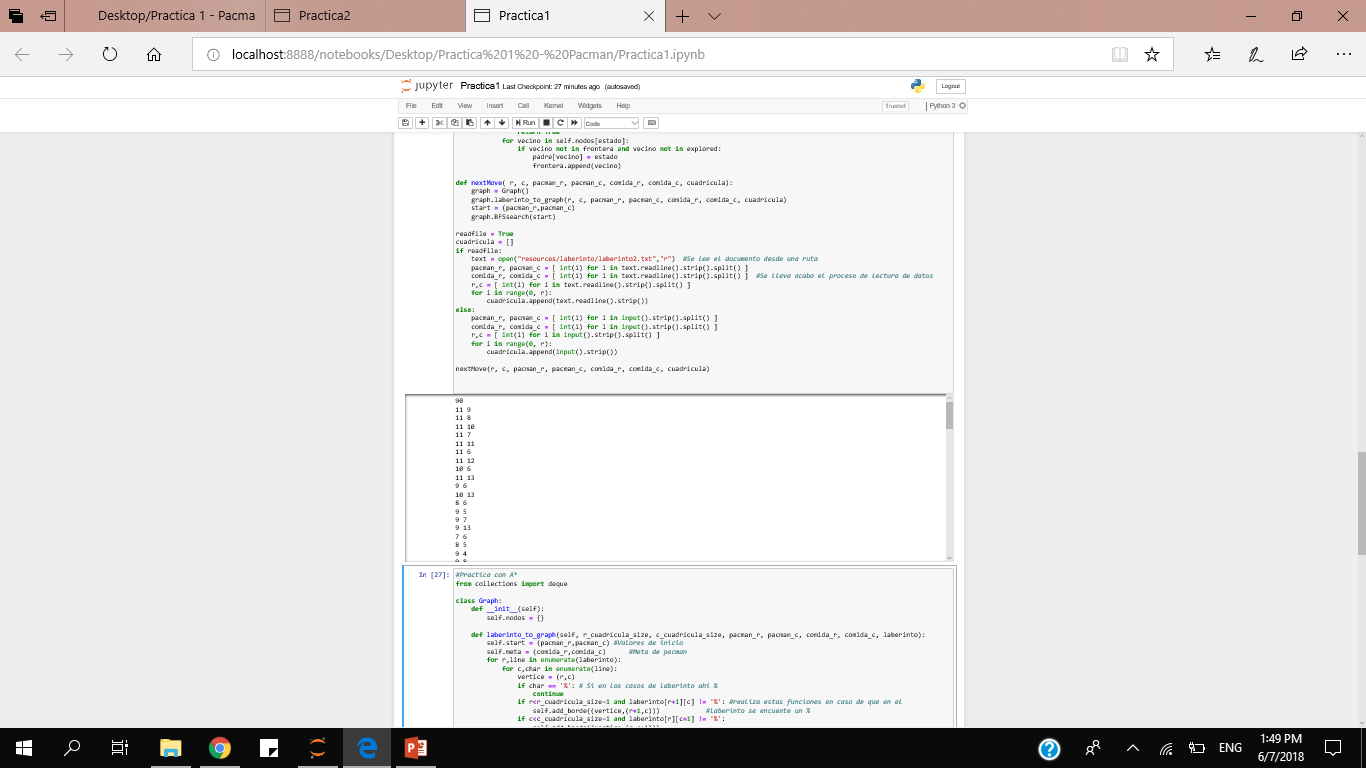
**Resultados**

Se ingresaron laberintos para resolver en cada una de las búsquedas los resultados son los siguientes:

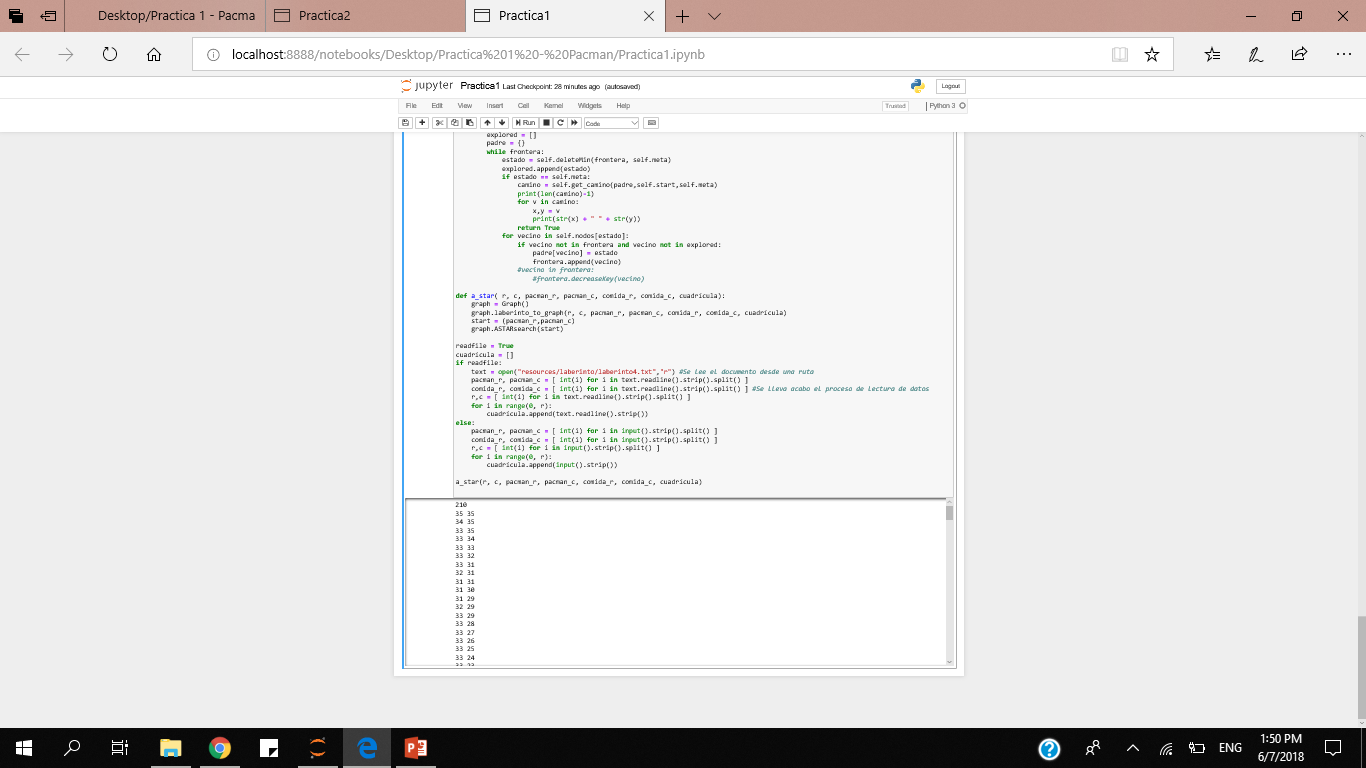
**BFS**

****

**DFS**

****

**A\***

****

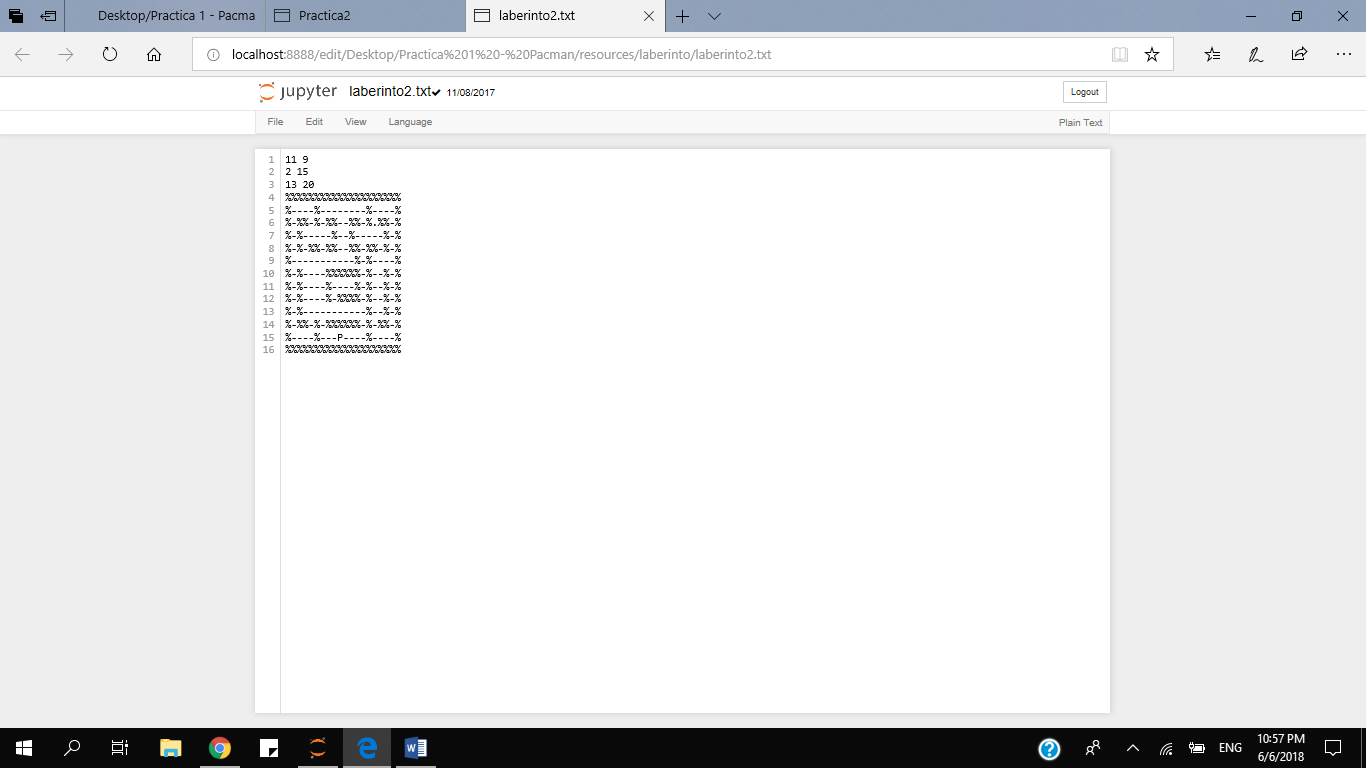
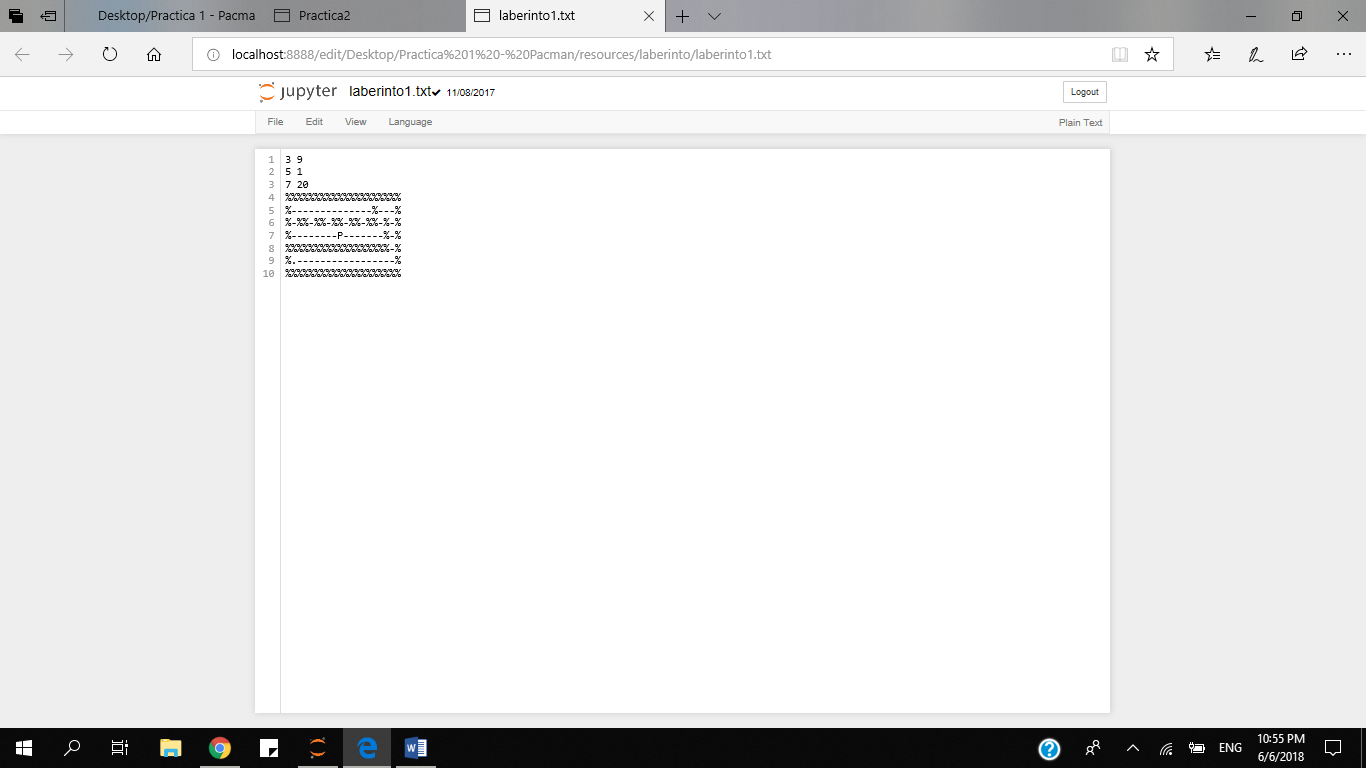
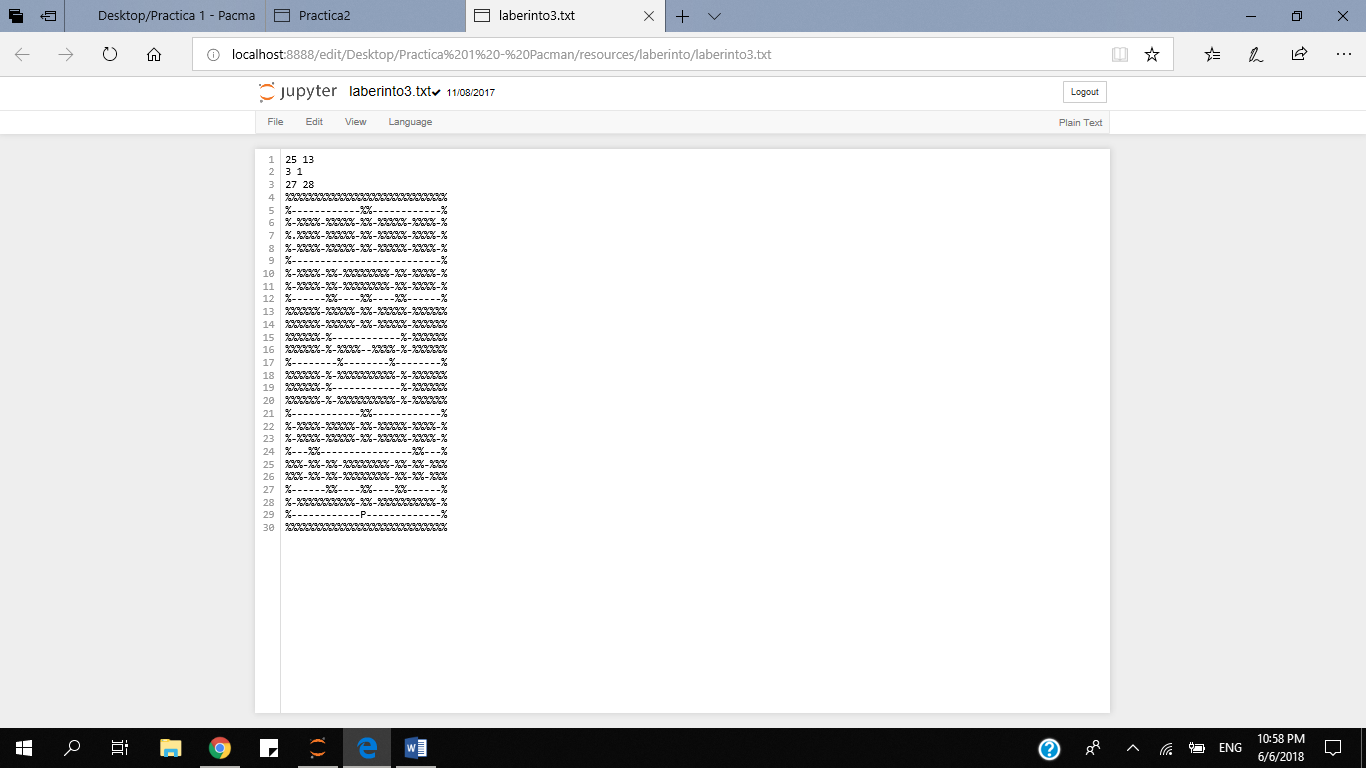
**Conclusiones**

Gracias a los distintos algoritmos de búsqueda que existen es que hoy en día se pueden optimizar una gran variedad de métodos, pues, lo más importante de un programa es poder encontrar las soluciones o los datos lo más rápido posible ya que el tiempo juega un papel muy importante dentro de este ámbito. A sí mismo, la implementación de estos, te pueden ayudar a terminar más rápido cualquier tipo de programa, aunque las complejidades de estos son un poco elevadas y a veces resulta un poco difícil comprenderlo.

**Referencias**

* HackerRank. (2018). Solve PacMan - BFS. [online] Available at: https://www.hackerrank.com/challenges/pacman-bfs [Accessed 8 Jun. 2018].
* HackerRank. (2018). Solve PacMan - DFS. [online] Available at: https://www.hackerrank.com/challenges/pacman-dfs [Accessed 8 Jun. 2018].
* HackerRank. (2018). Solve Pacman A\*. [online] Available at: https://www.hackerrank.com/challenges/pacman-astar [Accessed 8 Jun. 2018].
* Search?, H. (2018). How to trace the path in a Breadth-First Search?. [online] Stack Overflow. Available at: https://stackoverflow.com/questions/8922060/how-to-trace-the-camino-in-a-breadth-first-search [Accessed 8 Jun. 2018].
* GeeksforGeeks. (2018). Depth First Search or DFS for a Graph - GeeksforGeeks. [online] Available at: https://www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph/ [Accessed 8 Jun. 2018].
* GeeksforGeeks. (2018). A\* Search Algorithm - GeeksforGeeks. [online] Available at: https://www.geeksforgeeks.org/a-search-algorithm/ [Accessed 8 Jun. 2018].

**Apéndices**

 Laberinto 1 Laberinto 2 Laberinto 3