Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2115 – Programación como Herramienta para la Ingeniería (II/2018)

Laboratorio 3

Objetivo

- Aplicar técnicas avanzadas de programación, en conjunto con estructuras de datos adecuadas y programación orientada a objetos.
- Aplicar distintos algoritmos de búsqueda y ordenamiento, dependiendo de las estructuras de datos utilizadas.
- Evaluar la complejidad computacional de los algoritmos utilizados, con el fin de generar una solución eficiente.

Entrega

- **Lenguaje a utilizar:** Python 3.6
- **Lugar:** repositorio GitHub. Recuerde incluir todo en una carpeta de nombre **L03**.
- **Fecha límite de entrega: miércoles 27 de septiembre, a las 23:59 hrs.**
- **Formato de entrega:** dos archivos con la solución propuesta, uno en formato **.py** y el otro en **.ipynb**. Deben ser exactamente iguales (el ayudante revisará cualquiera de ellos). Adicionalmente, incluya un archivo **README.md** con información útil para el ayudante corrector.

Introducción

Decides ir con un grupo de amigos a una fonda para celebrar las Fiestas Patrias. En la fonda, hay abundancia de choripanes, anticuchos, empanadas y terremotos por toda la zona. El grupo se pone de acuerdo para poder consumir lo más posible ese día.

Instrucciones

Todo el grupo se encontraba en la fonda buscando que comer, y encontraron un mapa de todo el recinto con instrucciones que detallaban como era aquella zona.

1. Espacio transitable. Se puede caminar sobre él. Se representa con un "_".
2. Obstáculo. No se puede pasar sobre el. Se representa con una "X".
3. Choripán: Se representa con la letra "C" y tiene 500 calorías.
4. Anticucho: Un anticucho tiene 300 calorías. Se representa con la letra "A".

5. Empanada: Una empanada tiene 600 calorías. Se representa con la letra "E".

6. Terremoto: Esta delicia tiene 200 calorías. Se representa con la letra "T"

Al estar todos bastante cansados, pero con las ganas intactas de comer, deciden que lo mejor es separarse para explorar la fonda de manera más efectiva y eficiente. Como eres el único del grupo que conoce sobre técnicas de programación, tu misión es proponer un algoritmo que permita, dado un mapa[1], generar rutas para cada uno de tus compañeros, de manera que entre ellos recolecten la totalidad de comida en la fonda.

Es importante notar que, por limitaciones humanas (y criterio moral), tus compañeros no pueden consumir más de 4.000 calorías. Por suerte, el mapa les informa sobre el valor total de calorías que están repartidas en la fonda y este resulta ser menor o igual a la capacidad de consumo de los compañeros, por la cantidad de compañeros (*i.e.* si hay m compañeros, el mapa tendrá $4000m$ calorías).

Concretamente, debes crear un programa que reciba una cantidad de compañeros (desde ahora llamados "amigos") y un mapa (como una lista de listas con los símbolos descritos previamente) y que haga lo siguiente:

- **Asigne a cada amigo una posición aleatoria** (pero válida) en el mapa, desde la cual comenzar a explorar. Cabe aclarar que dos amigos pueden estar en la misma posición. Los compañeros se representan con la letra G mayúscula[2] y un número natural positivo (*e.g.* "G1" para el primer amigo). Luego de asignar las posiciones iniciales, se debe imprimir el estado actual del mapa en pantalla. En relación a esto último, cuando hay más de un amigo en una casilla, sus nombres se separan por comas (*e.g.* "G1,G6,G10" para los amigos 1, 6 y 10 en una misma casilla). Es importante que el formato visual del mapa sea claro, entendible y mantenga correctamente el alineamiento de filas y/o columnas.

- **Otorgue un turno a cada fiestero**, de manera tal que en un determinado momento, no pueda haber más de un amigo moviéndose. A la secuencia de turnos de todos los amigos se le denomina *episodios*. El movimiento funciona igual para todos los amigos: en su turno se pueden mover solo una casilla hacia arriba, hacia abajo, hacia la derecha o hacia la izquierda (solo cuando no hayan obstáculos). Además, los que no pueden seguir comiendo, se quedan en la última posición utilizada. Luego de cada episodio, **se debe preguntar si se desea mostrar en pantalla el estado actual del mapa o sólo mostrar el estado final, imprimiéndolo de forma ordenada**[3] (si se solicita la segunda opción, sólo debe imprimir el estado final una vez).

- Genere las rutas necesarias para conseguir la comida de cada amigo, en base a las posiciones iniciales definidas anteriormente. Al finalizar todo el proceso (todos los episodios), se espera que se imprima en pantalla, por cada amigo, una lista de tuplas (x, y), donde la K-ésima tupla representa la posición del mapa en que estuvo el amigo en el episodio K. Además, se deben guardar en una lista la comida obtenida (choripanes, anticuchos, empanadas, y terremotos), manteniendo el orden en que estos fueron encontradas.

[1] Representado por una lista de listas.
[2] Guatón Loyola.
[3] Al final de este enunciado se encuentran ejemplos de mapa

Búsqueda

El procedimiento de búsqueda la comida de cada compañero debe ser óptimo. Para ello, se te solicita que utilices el algortimo de A* con cada uno de tus amigos, los que se moverán una casilla por turno (siguiendo el orden de turnos otorgado anteriormente). Debes priorizar las comidas que se encuentren más cercanas, ya que si se demoran mucho en llegar a una comida, otro de tus amigos podría quitársela (camarón que se duerme, se lo lleva la corriente). En caso de que ocurra esto, el amigo que se quedó sin comida debe volver a trazar una nueva ruta.
Con el fin de comparar rendimiento, deberá utilizar también los algoritmos de BFS y DFS para encontrar la comida, así que antes de comenzar el programa, deberá preguntar al usuario que lo utiliza qué algoritmo desea utilizar (es decir, el proceso de búsqueda se realizará de 3 formas distintas dependiendo del modo que ingrese el usuario)

Ordenamiento

Una vez resuelto el problema de búsqueda de comida, deberán ejecutar un algoritmo que ordene la lista de comidas encontradas **de manera estable**, en orden ascendente (en base al valor calórico de cada comida), mediante el siguiente procedimiento recursivo, popularmente conocido como **HuaSort**:

1. Dividir la lista en sublistas, de la forma más equilibrada posible (tamaño de cada sublista).
2. Ordenar las sublistas usando el mismo algoritmo.
3. Juntar todas las sublistas ordenadas de forma tal de obtener los elementos de la lista inicial ordenados.

Ejemplos de mapa

Si quieres puedes utilizar estos mapas para probar tu código, pero para usarlos debes remover los compañeros y poner los que necesites.

Ejemplo 1

Nótese que el compañero G1 está en la misma celda que un Choripán en la esquina inferior derecha. Cuando un compañero llega a una celda con comida, se separan también por comas.

```
[['E', '_', 'G2', '_', '_', 'X', 'C'    ],
 ['_', '_', '_' , '_', 'E', 'X', '_'    ],
 ['X', 'X', 'X' , 'X', '_', 'X', '_'    ],
 ['_', 'T', '_' , 'X', '_', 'X', '_'    ],
 ['_', '_', '_' , 'X', '_', 'X', '_'    ],
 ['_', 'X', '_' , '_', '_', '_', '_'    ],
 ['_', 'X', '_' , '_', '_', '_', 'C,G1']]
```

Ejemplo 2

```
[['C','_', 'E', 'X' , '_', '_','C', 'X', '_', 'C', 'T' , 'X', '_'],
 ['_','X', '_', 'X' , 'A', 'X','_', 'X', '_', 'X', '_' , 'X', 'C'],
 ['_','X', 'C', 'X' , '_', 'X','C', 'X', '_', 'X', 'G2', 'X', '_'],
 ['C','X', '_', 'G1', '_', 'X','_', '_', 'T', 'X', '_' , '_', 'C']]
```

Política de Integridad Académica

Los alumnos de la Escuela de Ingeniería deben mantener un comportamiento acorde al Código de Honor de la Universidad:

"Como miembro de la comunidad de la Pontificia Universidad Católica de Chile me comprometo a respetar los principios y normativas que la rigen. Asimismo, prometo actuar con rectitud y honestidad en las relaciones con los demás integrantes de la comunidad y en la realización de todo trabajo, particularmente en aquellas actividades vinculadas a la docencia, el aprendizaje y la creación, difusión y transferencia del conocimiento. Además, velaré por la integridad de las personas y cuidaré los bienes de la Universidad."

En particular, se espera que mantengan altos estándares de honestidad académica. Cualquier acto deshonesto o fraude académico está prohibido; los alumnos que incurran en este tipo de acciones se exponen a un procedimiento sumario. Ejemplos de actos deshonestos son la copia, el uso de material o equipos no permitidos en las evaluaciones, el plagio, o la falsificación de identidad, entre otros. Específicamente, para los cursos del Departamento de Ciencia de la Computación, rige obligatoriamente la siguiente política de integridad académica en relación a copia y plagio: Todo trabajo presentado por un alumno (grupo) para los efectos de la evaluación de un curso debe ser hecho individualmente por el alumno (grupo), sin apoyo en material de terceros. Si un alumno (grupo) copia un trabajo, se le calificará con nota 1.0 en dicha evaluación y dependiendo de la gravedad de sus acciones podrá tener un 1.0 en todo ese ítem de evaluaciones o un 1.1 en el curso. Además, los antecedentes serán enviados a la Dirección de Docencia de la Escuela de Ingeniería para evaluar posteriores sanciones en conjunto con la Universidad, las que pueden incluir un procedimiento sumario. Por "copia" o "plagio" se entiende incluir en el trabajo presentado como propio, partes desarrolladas por otra persona. Está permitido usar material disponible públicamente, por ejemplo, libros o contenidos tomados de Internet, siempre y cuando se incluya la cita correspondiente.