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

IIC2115 – Programación como Herramienta para la Ingeniería (II/2017)

Actividad 6

Objetivo

- Aplicar técnicas avanzadas de programación, en conjunto con estructuras de datos adecuadas y programación orientada a objetos.
- Utilizar la estrategia de *dividir y conquistar* para resolver problemas de manera eficiente.
- 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
- **Lugar:** GitHub
- **Hora:** 16:55

Introducción

Te encuentras, junto a un grupo de excursionistas, en una zona que posee preciosos tesoros. Rubíes, zafiros y esmeraldas les rodean, así que convences al resto de tus acompañantes para explorar el terreno y llevarse a casa el botín.

Instrucciones

El grupo y tú llevaban varias horas caminando sin rumbo, hasta que encontraron lo que parecía un mapa con su respectiva leyenda:

1. Bloque libre. Se puede caminar sobre él. Se representa con un guión bajo ("_").
2. Roca corriente. No se puede pasar sobre ella. Se representa con una *equis* mayúscula ("X").
3. Rubí. Se representa con una *erre* mayúscula ("R").
4. Esmeralda. En este mundo, una esmeralda equivale a 0.5 rubíes. Se representa con una *e* mayúscula ("E").

5. Zafiro. Cada uno equivale a 0.25 rubíes. Se representa con una *zeta* mayúscula ("Z").

Al estar todos bastante cansados, pero con las ganas intactas de recolectar tesoros, deciden que lo mejor es separarse para explorar la zona 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 los cazafortunas, de manera que entre ellos recolecten la totalidad de los tesoros del mapa.

Es importante notar que, por limitaciones humanas, los excursionistas no pueden cargar más del equivalente a 11 rubíes. Por suerte (o infortunio), el mapa les informa sobre el valor total (en rubíes) del tesoro que esconde y este resulta ser menor o igual a la capacidad de carga de los excursionistas, menos la cantidad de excursionistas (*i.e.* si hay m excursionistas, el mapa tendría el equivalente en gemas a $10m$ rubíes).

Concretamente, debes crear un programa que reciba una cantidad de excursionistas y un mapa (como una lista de listas con los símbolos descritos previamente) y que haga lo siguiente:

- **Asigne a cada excursionista una posición aleatoria** (pero válida) en el mapa, desde la cual comenzar a explorar. Cabe aclarar que un excursionista **sí** puede estar en la misma casilla que otro. Los excursionistas se representan con la letra i mayúscula[2] y un número natural positivo (*e.g.* "I1" para el primer excursionista). 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 excursionista en una casilla, sus nombres se separan por comas (*e.g.* "I1,I6,I10" para los excursionistas 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 cazafortunas**, de manera tal que en un determinado momento, no pueda haber más de un excursionista moviéndose. A la secuencia de turnos de todos los excursionistas se le denomina *episodio*. El movimiento funciona igual para todos los exploradores: 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 recogiendo gemas, se quedan en la última posición utilizada. Luego de cada episodio, **se debe mostrar en pantalla el estado actual del mapa**[3].

- Genere las rutas necesarias para conseguir el botín para cada excursionista, 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 excursionista, una lista de tuplas (x, y), donde la K-ésima tupla representa la posición del mapa en que estuvo el excursionista en el episodio K. Además, se deben guardar en una lista las piedras obtenidas (rubíes, esmeraldas y zafiros), manteniendo el orden en que estas fueron encontradas.

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

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.

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

Ejemplos de mapa

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

Ejemplo 1

Nótese que el exucursionista I1 está en la misma celda que un rubí en la esquina inferior derecha. Cuando un excursionista llega a una celda con una gema, se separan también por comas.

```
[['E', '_', 'I2', '_', '_', 'X', 'R'    ],
 ['_', '_', '_' , '_', 'E', 'X', '_'    ],
 ['X', 'X', 'X' , 'X', '_', 'X', '_'    ],
 ['_', 'Z', '_' , 'X', '_', 'X', '_'    ],
 ['_', '_', '_' , 'X', '_', 'X', '_'    ],
 ['_', 'X', '_' , '_', '_', '_', '_'    ],
 ['_', 'X', '_' , '_', '_', '_', 'R,I1']]
```

Ejemplo 2

```
[['R','_', 'E', 'X' , '_', '_','R', 'X', '_', 'R', 'Z' , 'X', '_'],
 ['_','X', '_', 'X' , 'E', 'X','_', 'X', '_', 'X', '_' , 'X', 'R'],
 ['_','X', 'R', 'X' , '_', 'X','R', 'X', '_', 'X', 'I2', 'X', '_'],
 ['R','X', '_', 'I1', '_', 'X','_', '_', 'Z', 'X', '_' , '_', 'R']]
```

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.