TP 3: Camino mínimo a maestro Pokemon

Fecha de entrega: viernes 11 de noviembre, hasta las 18:00 horas.

Al comienzo del cuatrimestre fue el lanzamiento en Argentina de un juego conocido como *Pokemon Go*. Durante las primeras semanas fue noticia en varios medios de comunicación, se veia a muchas personas persiguiendo a los bichos (conocidos como pokemones) por la calle.

En este problema, vamos a ayudar a Brian a convertirse en un maestro Pokemon. En el juego, **existen gimnasios** que estan controlados por uno de tres equipos. Un jugador es de algún equipo que elige inicialmente. Donde vive Brian, todos los gimnasios estan controlados por equipos enemigos. Cada gimnasio tiene un conjunto de pokemones que deben ser vencidos para tomar el control. Por su parte, los jugadores también tienen un conjunto de pokemones que los puede usar para realizar batallas en los gimnasios. Una batalla exitosa debilita al gimnasio. Luego de algunas batallas, el gimnasio se debilita lo suficiente y es vencido.

Para realizar multiples batallas en un gimnasio, un jugador puede curar a sus pokemones con pociones para realizar una nueva batalla. **Las pociones se pueden buscar en estaciones conocidas como poke paradas**. Cuando un jugador va a una poke parada, esta le entrega algunas pociones y el jugador no puede solicitar más pociones por cierto tiempo. Las pociones son guardadas en **una mochila que tiene un límite de peso**. Si no hay lugar en la mochila, las pociones son descartadas y no se puede volver por ellas. Para este problema vamos a suponer que todas las poke paradas entregan 3 pociones cuando son visitadas y no se pueden visitar dos veces.

Un jugador es capaz de ver la fuerza de un gimnasio antes de visitarlo. **Para vencer a un gimnasio es necesario llevar una cierta cantidad de pociones**. Cuando un gimnasio es atacado, se pierden esa cantidad de pociones. Si no se tiene suficiente pociones, no va a visitar el gimnasio porque estos pueden recuperar fuerza, es clave que cuando se visita al gimansio sea seguro de poder vencerlo.

Brian esta muy ocupado corrigiendo TPs, por lo que quiere saber **cuanto tiempo le lleva vencer todos los gimnasios**. El objetivo, para convertirse en maestro pokemon, es vencer a todos los gimnasios recorriendo la mínima distancia, sin pasar por un gimnasio, o una poke parada dos veces.

En el presente trabajo práctico se pide:

1. Diseñar e implementar un **algoritmo exacto** para el problema y desarrollar los siguientes puntos:
   a) Explicar detalladamente el algoritmo implementado. Elaborar podas y estrategias que permitan mejorar los tiempos de resolución.
   b) Calcular el orden de complejidad temporal de peor caso del algoritmo.
   c) Realizar una experimentación que permita observar los tiempos de ejecución del algoritmo en función del tamaño de entrada y de las podas y/o estrategias implementadas.
2. Diseñar e implementar una **heurística constructiva golosa** para el problema y desarrollar los siguientes puntos:
   a) Explicar detalladamente el algoritmo implementado.
   b) Calcular el orden de complejidad temporal de peor caso del algoritmo.
   c) Describir instancias de el problema para las cuales la heurística no proporciona una solución óptima. Indicar qué tan mala puede ser la solución obtenida respecto de la solución óptima.
   d) Realizar una experimentación que permita observar la performance del algoritmo en términos de tiempo de ejecución en función del tamaño de entrada.
3. Diseñar e implementar una **heurística de búsqueda local** para el problema y desarrollar los siguientes puntos:
   a) Explicar detalladamente el algoritmo implementado. Plantear al menos dos vecindades distintas para la búsqueda.
   b) Calcular el orden de complejidad temporal de peor caso de una iteración del algoritmo de búsqueda local (para las vecindades planteadas). Si es posible, dar una cota superior para la cantidad de iteraciones de la heurística.

   c) Realizar una experimentación que permita observar la performance del algoritmo comparando los tiempos de ejecución y la calidad de las soluciones obtenidas, en función de las vecindades utilizadas y elegir, si es posible, la configuración que mejores resultados provea para el grupo de instancias utilizado.

4. Utilizando las heurísticas implementadas en los puntos anteriores, diseñar e implementar un algoritmo para el problema que use alguna **metaheurística** (Tabu [1, 2] o GRASP [3]) y desarrollar los siguientes puntos:

   a) Explicar detalladamente el algoritmo implementado. Plantear distintos criterios de parada y de selección.

   b) Realizar una experimentación que permita observar los tiempos de ejecución y la calidad de las soluciones obtenidas. Se debe experimentar variando los valores de los parámetros de la metaheurística (lista de candidatos, criterios de parada, etc.) y las vecindades utilizadas en la búsqueda local. Elegir, si es posible, la configuración que mejores resultados provea para el grupo de instancias utilizado.

5. Una vez elegidos los mejores valores de configuración para cada heurística implementada (si fue posible), realizar una **experimentación sobre un conjunto nuevo de instancias** para observar la performance de los métodos comparando nuevamente la calidad de las soluciones obtenidas y los tiempos de ejecución en función del tamaño de entrada. Para los casos que sea posible, comparar también los resultados del algoritmo exacto implementado. Presentar todos los resultados obtenidos mediante gráficos adecuados y discutir al respecto de los mismos.

Condiciones de entrega y términos de aprobación

Este trabajo práctico consta de varias partes y para aprobar el trabajo se requiere aprobar todas las partes del mismo. La nota final del trabajo será un promedio ponderado de las notas finales de las partes y el trabajo práctico se aprobará con una nota de 5 (*cinco*) o superior. De ser necesario (o si el grupo lo desea) el trabajo podrá reentregarse una vez corregido por los docentes y en ese caso la reentrega deberá estar acompañada por un *informe de modificaciones*. Este informe deberá detallar brevemente las diferencias entre las dos entregas, especificando los cambios, agregados y/o partes eliminadas del trabajo. Cualquier cambio que no se encuentre en dicho informe podrá no ser tenido en cuenta en la corrección de la reentrega.

Respecto de las implementaciones, se acepta cualquier lenguaje que permita el cálculo de complejidades según la forma vista en la materia. Además, debe poder compilarse y ejecutarse correctamente en las máquinas de los laboratorios del Departamento de Computación. La cátedra recomienda el uso de C++ o Java, y se sugiere consultar con los docentes la elección de otros lenguajes para la implementación.

Deberá entregarse un informe impreso que desarrolle los puntos mencionados. Por otro lado, deberá entregarse el mismo informe en formato digital acompañado de los códigos fuentes desarrollados e instrucciones de compilación, de ser necesarias. Estos archivos deberán enviarse a la dirección `algo3.dc@gmail.com` con el asunto *"TP 3: Apellido_1, ..., Apellido_n"*, donde n es la cantidad de integrantes del grupo y *Apellido_i* es el apellido del i-ésimo integrante.

La entrada y salida de los programas **deberá hacerse por medio de la entrada y salida estándar del sistema**. No se deben considerar los tiempos de lectura/escritura al medir los tiempos de ejecución de los programas implementados.

Formato de entrada: La entrada comienza con una línea con tres valores enteros positivos, n, m y k, separados por espacios; los valores n y m indican la cantidad gimnasios y paradas, respectivamente, mientras que k indica el tamaño de la mochila. A continuación, siguen n líneas, representando los gimnasios. Cada una contiene 3 enteros, x, y y p, donde x e y indican la posición, y p la cantidad de pociones necesarias para vencer el gimnasio. Las siguentes m lineas indican la posición de las pokeparadas, cada una contiene dos enteros x e y.

Ejemplo el formato:

```
n m k
xg1 yg1 pg1
xg2 yg2 pg2
...
xgn ygn pgn
xp1 yp1
xp2 yp2
...
xpm ypm
```

Formato de salida: La salida debe contener una línea con el siguiente formato:

```
D k i1 i2 ... ik
```

donde `D` y `k` son la distancia total recorrida y la cantidad de vértices de la solución, respectivamente, y `i1`, ..., `ik` son los indices de 1 a $n + m$ de acuerdo a su linea en el archivo de entrada (es decir de 1 a n son gimnasios, y el resto son poke paradas). En caso de no existir solución simplemente devolver -1.

Referencias

[1] Fred Glover. *Tabu Search - Part 1*. ORSA Journal on Computing **1** (3), pp: 190–206, 1989.

[2] Fred Glover. *Tabu Search - Part 2*. ORSA Journal on Computing **2** (1), pp: 4–32, 1990.

[3] Thomas A. Feo and Mauricio G. C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization* 6, pp 109–134, 1995.