Trabajo práctico 2: Diseño
PokémonGOArgentina

Normativa

Límite de entrega: Domingo 16 de Octubre *hasta las 22:00 hs.* Enviar PDF a `algo2.dc+TP2@gmail.com` con asunto "Grupo: nombre del grupo"

Normas de entrega: Ver "Información sobre la cursada" en el sitio Web de la materia.
(`http://www.dc.uba.ar/materias/aed2/2016/2c/cursada`)

Versión: 1.1 del 28 de septiembre de 2016 (ver TP2_Changelog.txt)

Enunciado

El objetivo de este TP es diseñar el módulo correspondiente al TAD Juego, junto con todos los módulos que sean necesarios para contar con un diseño **completo**.

Este enunciado contiene la especificación del TAD y demás tipos necesarios tal como deberá ser diseñado. La especificación presentada puede diferir de los criterios presentados para el TP1, por lo que es importante leerla y entenderla nuevamente. En particular se resaltan los siguientes cambios:

- Como parte de las definiciones de diseño era necesario establecer cuál de los entrenadores que podrían capturar un pokémon era el que lo capturaba. Se definió que el jugador que lo captura es aquel que tiene menor cantidad de pokémons capturados.
- Por otra parte, en Mapa, se considera que dos coordenadas son adyacentes si difieren en uno en latitud o en longitud, i.e. es una grilla. Además, la especificación explícita cuando hay un camino entre dos coordenadas.
- También se cambió el criterio con el que se identifican los jugadores. Los mismos se siguen identificando como un Nat pero ahora es el juego quen establece que Nat corresponde a cada jugador. Ver la especificación para mayor detalle.

Se requiere que todas las operaciones exportadas de los TADs tengan una contraparte en los módulos diseñados. Recomendamos modularizar adecuadamente: no necesariamente se aconseja hacer un único módulo por cada TAD.

Contexto de uso y complejidades requeridas

A partir de la especificación definimos las siguientes variables:

- J es la cantidad total de jugadores que fueron agregados al juego.
- $|P|$ es el nombre más largo para un pokémon en el juego.
- EC es la máxima cantidad de jugadores esperando para atrapar un pokémon (ver Figura 1 en Apéndice).
- PS es la cantidad de pokémon salvajes.
- PC es la máxima cantidad de pokémon capturados por un jugador.

Se definen las siguientes restricciones temporales para la implementación de las funciones de los módulos:

Para Juego:

- agregarJugador: $\mathcal{O}(J)$. Aparte, esta función deberá cambiar su aridad respecto del TAD devolviendo el id del jugador agregado.
- agregarPokémon: $\mathcal{O}(|P| + EC * log(EC))$
- posConPokémons: $\mathcal{O}(1)$ devolviendo el conjunto por referencia.
- conectarse: $\mathcal{O}(log(EC))$
- desconectarse: $\mathcal{O}(log(EC))$
- moverse: $\mathcal{O}((PS + PC) * |P| + log(EC))$
- índiceRareza: $\mathcal{O}(|P|)$

- jugadores: $\mathcal{O}(1)$ devolviendo un iterador a los jugadores. El Siguiente del iterador puede no ser $\mathcal{O}(1)$.
- estáConectado, sanciones, posición: $\mathcal{O}(1)$
- pokémons: $\mathcal{O}(1)$ devolviendo un iterador de tupla⟨pokemon, cantidad⟩.
- posPokémonCercano: $\mathcal{O}(1)$

Requisitos y consideraciones

- Cuentan con los siguiente TADs y módulos ya diseñados:
  - Char que representan los posibles caracteres. Siendo un tipo enumerado de 256 valores. con funciones ord y ord^{-1} para la correspondencia de cada $Char$ a Nat.
  - String como sinónimo de $Vector(Char)$.
  - Todos los definidos en el apunte de TADs básicos.
  - Todos los definidos en el apunte de módulos básicos.
- Los módulos definidos en el apunte de módulos básicos pueden utilizarse como están. Esto es, no es necesario hacer el diseño y alcanza con nombrarlos en la sección de servicios utilizados con sus géneros exportados. Si por algún motivo la inferfaz del módulo del apunte no es la que necesitan (ya sea por aridad, complejidad, aliasing, etc.), está en ustedes diseñar el módulo completo que cumpla estos requerimientos. El diseño incluye Rep y Abs, por lo que necesita un TAD que lo acompañe. Pueden utilizar los TADs básicos, nuevamente, si es que cumplen con lo que necesitan.
- Todos los algoritmos (exportados y auxiliares) *deben* tener su desarrollo que justifique los órdenes de complejidad. Si algún paso es no trivial pueden hacer notas a continuación del mismo.
- Respetando los generadores y observadores de los TADs, ciertos cambios de aridad en las funciones son aceptables si hacen más claro el uso del módulo, considerando el cambio de paradigma.
- Todas las operaciones auxiliares deben ser especificadas formalmente según las herramientas vistas en clase (pre y pos, complejidad y justificación, descripción, etc.) y no deberían aparecer en la interfaz del módulo. Agregar comentarios necesarios para entender la forma en la cual deben ser utilizadas para su correcto funcionamiento.
- Cuando se formalicen los invariantes y funciones de abstracción, *deben* identificar cada parte de la fórmula del Rep y comentar en castellano lo que describe.
- El invariante de representación de las estructuras recursivas (aquellas que involucren punteros) puede escribirse en castellano.
- Tener en cuenta que las complejidades son en peor caso. Soluciones más eficientes serán bien recibidas.
- También, tener en cuenta que hay estructuras que pueden servir para más de una finalidad, sobre todo los contenedores.
- Pueden crear módulos adicionales si así lo necesitan.
- Para el resto de los módulos que se expliquen con TADs básicos o TADs que están en la especificación adjunta, solo es obligatorio incluir en el diseño los generadores y observadores básicos. Las otras operaciones pueden incluirlas o no según las precisen.

Apéndice

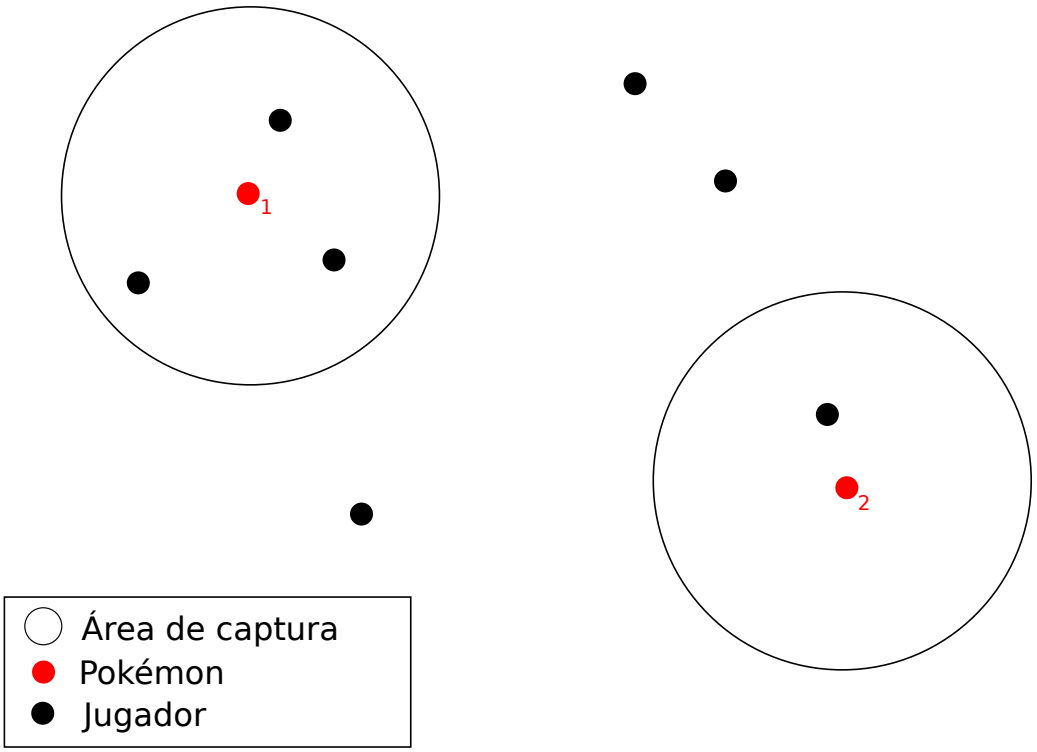

Figura 1: Ejemplo del valor de EC: en este ejemplo de juego el pokémon 1 tiene 3 jugadores a distancia de captura y el pókemon 2 solo uno. Hay otros jugadores en el juego que no están a distancia de captura de ningún pokémon. Para este escenario el valor de EC es 3.