Trabajo práctico 1: Diseño

v1.0.1

SimCity

Normativa

Límite de entrega:

Normas de entrega: Ver "Información sobre la cursada" en el sitio Web de la materia. (`http://campus.exactas.uba.ar`)

Enunciado

Este trabajo práctico plantea modelar una lógica simplificada del juego SimCity de 1989. `https://www.youtube.com/watch?v=A54blk-ojA4`.

El SimCity es un juego de simulador de la vida y crecimiento de las ciudades. En el recorte para el TP, solo vamos a manejar el crecimiento de casas y comercios.

- En nuestra versión del juego, las partidas comienzan con un mapa ya definido. Un mapa puede considerarse una grilla infinita en cual corren ríos. Los ríos corren en líneas infinitas horizontales o verticales. En la partida, en cada turno pueden agregarse casas o comercios. Cada uno ocupa una única casilla, que no puede estar en el mismo lugar que un río ni encima de otra construcción.
- Los comercios y casas además tienen un nivel de avance. En las casas, el nivel se define como la cantidad de turnos que sucedieron desde que fue agregada. En los comercios, el nivel se define de igual manera, con la excepción de que si una casa a 3 casilleros de distancia manhattan (`https://es.wikipedia.org/wiki/Geometría_del_taxista`) tiene un nivel más alto, el comercio adopta ese nivel.
- Como en el juego original, no se pueden agregar construcciones en turnos anteriores, sin embargo, se debe agregar al menos una construcción en cada turno.
- Partidas colaborativas: dos partidas de SimCity deben poder unirse en una nueva. Esta acción deberá hacer concordar los ríos y construcciones de ambas ciudades. Para que el mecanismo de unión sea constructivo y no destructivo, se deberá restringir la posibilidad de unir dos SimCitys a los casos donde no se solapen ríos con construcciones. Además, para evitar la depresión de los jugadores, tampoco deberán solaparse las construcciones de nivel máximo de cualquiera de las partidas con otras construcciones.

  Con estas consideraciones, deberán tomar decisiones respecto de cómo resolver conflictos entre construcciones de ambos SimCitys que ocupen el mismo casillero (casa vs. casa, comercio vs. comercio o casa vs. comercio).
- La semantica de la union, por ejemplo en sc1.union(sc2) es que sc1 observacionalmente tiene que mostrar las construcciones y los rios de sc2 como propios.
- Por simplicidad se debe asumir que: No hay "uniones cicurlares", esto significa que si si sc1.unir(sc2) y sc2.unir(sc3) no puede pasar que sc3.unir(sc1). Una vez que un SimCity es utilizado en una union, no se puede modifcar (agregarle construcciones o unirle otro SimCity. Esto significa que sc2 y sc3 no pueden modificarse. No obstante sc1, si.

  Las decisiones deberán estar documentadas debidamente especificando las decisiones tomadas, así como en la axiomatización. No son aceptables especificaciones que ignoren el conflicto y, por ejemplo, liberen el casillero.
- Para conocer la popularidad de la partida, es necesario poder conocer la cantidad de uniones que la conforman. Además nos interesa conocer la antigüedad de una partida como la cantidad de turnos que sucedieron en la misma. En el caso de unirse dos partidas, se considera la mayor antiguedad como la antiguedad de la unión.
- Por ultimo, se desea mantener un servidor de SimCitys en el cual se registrara un nombre único para cada juego. El nombre no esta acotado.
- Todas las operaciones se disparan desde el servidor.

1. Diseño

En el trabajo práctico de diseño se busca escribir los módulos que comprenderán la futura implementación del `SimCity`. La cátedra suministrará la interfaz de los TADs Mapa y SimCity. Tendrian que axiomatizarlos explicando el criterio elegido y además, especificar el TAD Servidor.

El entregable consiste en la presentación de los módulos de diseño, lo que incluye Interfaz, Estructura de Representación, Invariante de Representación, Función de Abstracción y Algoritmos. En el caso de utilizar un Trie, no hace falta implementar los algoritmos, solo se debe aclarar la interfaz con las complejidades asumidas. La interfaz de cada módulo debe declarar las funciones que la componen, con aridad completa (valores de entrada y salida) y contrato (precondición, postcondición y complejidad).

Además, la elección de estructuras de representación y algoritmos deberá ser de forma que se cumplan las restricciones de complejidad presentadas a continuación. Para ello, primero repasamos las operaciones que producen cambios en un `SimCity`:

- **`agregar casa`**. Un SimCity puede recibir una posición donde debe agregarse una casa.
- **`agregar comercio`**. Un SimCity puede recibir una posición donde debe agregarse un comercio.
- **`unir`**. Un SimCity pueden recibir otro SimCity e incorporarlo mediante la unión de la información de ambos.

Luego, el diseño a realizar debe cumplir con las siguientes restricciones de complejidad:

1. Conocer el turno actual de una partida debe ser $O(|Nombre|)$
2. Agregar una casa a una partida debe ser $O(|Nombre|)$
3. Agregar un comercio a una partida debe ser $O(|Nombre|)$
4. Unir dos partidas debe ser $O(|Nombre|)$
5. Conocer la popularidad de un SimCity debe ser $O(|Nombre|)$
6. Donde Nombre es el nombre mas largo de los SimCitys

Tener en cuenta que:

- Dado que es un mapa infinito, no se puede asumir que un hash sería $O(1)$
- El resto de las operaciones no tienen restricciones de complejidad

2. Entrega

La entrega deberá ser un único documento en formato `pdf` con los nombres de los integrantes y la descripción de los módulos, que incluye todo lo mencionado anteriormente. Este documento debe contener todos los módulos necesarios para una solución satisfactoria del `SimCity`.

Si bien no es necesario realizar la solución en LaTeX, junto con el enunciado proveemos un módulo `Mapa` que pueden utilizar y tener de referencia, junto con una versión del módulo `SimCity` para completar. Estos módulos están tanto en `pdf` como en LaTeX, de forma que puedan trabajar sobre estos archivos directamente.

La fecha de entrega es el 1/6/2022 hasta las 23:59 en el repo privado de cada grupo en la carpeta `tp1`, no aceptando entregas posteriores. Dado los feriados del 18/5 y 25/5, daremos una consulta virtual del TP por Discord el lunes 16/5 a las 11 y a las 18hs. En caso de no realizar una entrega válida, se podrán presentar al recuperatorio. El TP2 es la implementación del TP1 en C++; daremos mas detalles mas adelante. Para comenzar el TP2 tienen que aplicar las correcciones del docente asignado al TP1.

3. Especificación

TAD MAPA

igualdad observacional

$$(\forall m, m' : \text{Mapa}) \left(m =_{\text{obs}} m' \iff \begin{pmatrix} \text{horizontales}(m) =_{\text{obs}} \text{horizontales}(m') \wedge_{\text{L}} \\ \text{verticales}(m) =_{\text{obs}} \text{verticales}(m') \end{pmatrix} \right)$$

géneros Mapa

exporta completar

usa completar

observadores básicos

horizontales : Mapa $\longrightarrow$ conj(Nat)
verticales : Mapa $\longrightarrow$ conj(Nat)

Mapa

generadores

crear : conj(Nat) $\times$ conj(Nat) $\longrightarrow$ Mapa

axiomas $\forall hs, vs$: conj(Nat)

horizontales(crear(hs, vs)) $\equiv$ hs
verticales(crear(hs, vs)) $\equiv$ vs

Fin TAD

TAD SIMCITY

igualdad observacional

$$(\forall s, s' : \text{SimCity}) \left(s =_{\text{obs}} s' \iff \begin{pmatrix} \text{mapa}(s) =_{\text{obs}} \text{mapa}(s') \wedge_{\text{L}} \\ \text{completar} \end{pmatrix} \right)$$

géneros SimCity

exporta completar

usa completar

observadores básicos

mapa : SimCity $\longrightarrow$ Mapa
casas : SimCity $\longrightarrow$ dicc(Pos, Nivel)
comercios : SimCity $\longrightarrow$ dicc(Pos, Nivel)
popularidad : SimCity $\longrightarrow$ Nat

generadores

iniciar : Mapa $\longrightarrow$ SimCity
avanzarTurno : SimCity s $\times$ dicc(Pos $\times$ Construccion) cs $\longrightarrow$ SimCity
{completar: posiciones no estan ocupadas y no son ríos}
unir : SimCity a $\times$ SimCity b $\longrightarrow$ SimCity
{completar: ríos no elimnan construcciones y no se pisan construcciones de nivel máximo}

otras operaciones

turnos : SimCity $\longrightarrow$ Nat

completar

axiomas $\forall s, s'$: simcity, $\forall cs$: dicc(Pos, Construccion)

mapa(iniciar(m)) $\equiv$ m

completar

Fin TAD

4. Módulos de referencia

4.1. Módulo Mapa

Interfaz

se explica con: MAPA

géneros: `mapa`

Operaciones básicas de mapa

CREAR(**in** hs: `conj(Nat)`,**in** vs: `conj(Nat)`) $\rightarrow$ res : `mapa`
Pre $\equiv$ {true}
Post $\equiv$ $\{res =_{\text{obs}} mapa(hs, vs)\}$
Complejidad: $O(copy(hs), copy(vs))$
Descripción: crea un mapa

completar

Representación

Representación de mapa

Un mapa contiene rios infinitos horizontales y verticales. Los rios se representan como conjuntos lineales de naturales que indican la posición en los ejes de los ríos.

`mapa` **se representa con** `estr`

donde `estr` es `tupla`(*horizontales*: `conj(Nat)`, *verticales*: `conj(Nat)`)

Rep : estr $\longrightarrow$ bool
Rep(e) $\equiv$ true $\iff$ true

Abs : estr m $\longrightarrow$ mapa $\{\text{Rep}(m)\}$
Abs(m) $\equiv$ horizontales(m) = $estr$.horizontales $\wedge$ verticales(m) = $estr$.verticales

Algoritmos

---

crear(**in** hs: `conj(Nat)`, **in** vs: `conj(Nat)`) $\rightarrow$ res : estr

1: $estr.horizontales \leftarrow hs$
2: $estr.verticales \leftarrow vs$ **return** estr

Complejidad: $O(copy(hs) + copy(vs))$

---

completar