DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

TPE - Agricultura con drones

8 de abril de 2016 — Algoritmos y Estructuras de Datos I

Grupo 3

Integrante	LU	Correo electrónico
Apellido, Nombre1	001/01	email1@dominio.com
Apellido, Nombre2	002/01	email2@dominio.com
Apellido, Nombre3	003/01	email3@dominio.com
Apellido, Nombre4	004/01	email4@dominio.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2610 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (++54 +11) 4576-3300
http://www.exactas.uba.ar

1. Tipos

`tipo` Id = $\mathbb{Z}$;
`tipo` Carga = $\mathbb{Z}$;
`tipo` Ancho = $\mathbb{Z}$;
`tipo` Largo = $\mathbb{Z}$;
`tipo` Parcela = Cultivo, Granero, Casa ;
`tipo` Producto = Fertilizante, Plaguicida, PlaguicidaBajoConsumo, Herbicida, HerbicidaLargoAlcance ;
`tipo` EstadoCultivo = ReciénSembrado, EnCrecimiento, ListoParaCosechar, ConMaleza, ConPlaga, NoSensado ;

2. Campo

`tipo Campo` {
`observador dimensiones` (c: Campo) : (Ancho, Largo) ;
`observador contenido` (c: Campo, i, j: $\mathbb{Z}$) : Parcela ;
`requiere enRango` : $0 \leq i < prm(dimensiones(c)) \land 0 \leq j < sgd(dimensiones(c))$;

`invariante dimensionesValidas` : $prm(dimensiones(c)) > 0 \land sgd(dimensiones(c)) > 0$;
`invariante unaSolaCasa` : $|[(i,j)|i \leftarrow [0..prm(dimensiones(c))), j \leftarrow [0..sgd(dimensiones(c))), contenido(c,i,j) == Casa]| == 1$;
`invariante unSoloGranero` : $|[(i,j)|i \leftarrow [0..prm(dimensiones(c))), j \leftarrow [0..sgd(dimensiones(c))), contenido(c,i,j) == Granero]| == 1$;
`invariante algoDeCultivo` : $|[(i,j)|i \leftarrow [0..prm(dimensiones(c))), j \leftarrow [0..sgd(dimensiones(c))), contenido(c,i,j) == Cultivo]| \geq 1$;
`invariante posicionesAlcanzables` : $posicionesAlcanzablesEn100(c)$;
}

`aux posicionesAlcanzablesEn100` (c: Campo) : Bool = $alcanzableEn100(posicionGranero(c), prm(dimensiones(c)), sgd(dimensiones(c)))$;

`problema crearC` (posG, posC: ($\mathbb{Z}$, $\mathbb{Z}$)) = `res` : Campo {
`requiere` $posPositiva(posG) \land posPositiva(posC)$;
`requiere` $posG \neq posC$;
`requiere` $distancia((0,0), posG) \leq 100 \land distancia(posG, posC) \leq 100$;
`asegura` $contenido(res, \mathsf{prm}(posG), \mathsf{sgd}(posG)) == Granero$;
`asegura` $contenido(res, \mathsf{prm}(posC), \mathsf{sgd}(posC)) == Casa$;
}

`problema dimensionesC` (c: Campo) = `res` : (Ancho, Largo) {
`asegura` $res == dimensiones(c)$;
}

`problema contenidoC` (c: Campo, i, j: $\mathbb{Z}$) = `res` : Parcela {
`requiere` $enRango(dimensiones(c), i, j)$;
`asegura` $res == contenido(c, i, j)$;
}

3. Drone

tipo Drone {
observador id (d: Drone) : Id ;
observador bateria (d: Drone) : Carga ;
observador enVuelo (d: Drone) : Bool ;
observador vueloRealizado (d: Drone) : [(ℤ, ℤ)] ;
observador posicionActual (d: Drone) : (ℤ, ℤ) ;
observador productosDisponibles (d: Drone) : [Producto] ;

invariante vuelosOk :
$enVuelo(d) \Rightarrow (|vueloRealizado(d)| > 0 \land posicionActual(d) == vueloRealizado(d)_{|vueloRealizado(d)|-1} \land$
$posicionesPositivas(d) \land movimientosOK(d)) \land \neg enVuelo(d) \Rightarrow |vueloRealizado(d)| == 0$;
invariante bateriaOk : $0 \leq bateria(d) \leq 100$;
}

aux posicionesPositivas (d: Drone) : Bool $= (\forall i \leftarrow [0..|vueloRealizado(d)|))prm(vueloRealizado(d)_i) \geq 0 \land$
$sgd(vueloRealizado(d)_i \geq 0$;
aux movimientosOK (d: Drone) : Bool $= (\forall i \leftarrow [1..|vueloRealizado(d)|))$
$prm(vueloRealizado(d)_i) == prm(vueloRealizado(d)_{i-1}) \land (sgd(vueloRealizado(d)_i) == sgd(vueloRealizado(d)_{i-1}) - 1 \lor$
$sgd(vueloRealizado(d)_i) == sgd(vueloRealizado(d)_{i-1}) + 1) \lor sgd(vueloRealizado(d)_i) == sgd(vueloRealizado(d)_{i-1})$
$\land(prm(vueloRealizado(d)_i) == prm(vueloRealizado(d)_{i-1})-1 \lor prm(vueloRealizado(d)_{i-1}) == prm(vueloRealizado(d)_{i-1})+$
1) ;

problema crearD (id: Id, pd:[Producto]) = res : Drone {
asegura $id(res) == id$;
asegura $bateria(d) == 100$;
asegura $\neg enVuelo(res)$;
asegura $mismos(productosDisponibles(res), pd)$;
}

problema idD (d: Drone) = res : ℤ {
asegura $res == id(d)$;
}

problema bateriaD (d: Drone) = res : ℤ {
asegura $res == bateria(d)$;
}

problema enVueloD (d: Drone) = res : Bool {
asegura $res == enVuelo(d)$;
}

problema vueloRealizadoD (d: Drone) = res : [(ℤ,ℤ)] {
asegura $enVuelo(d) \Rightarrow mismos(res, cola(vueloRealizado(d)))$;
}

problema posicionActualD (d: Drone) = res : (ℤ, ℤ) {
asegura $res == posicionActual(d)$;
}

problema productosDisponiblesD (d: Drone) = res : [Producto] {
asegura $mismos(res, productosDisponibles(d))$;
}

problema vueloEscaleradoD (d: Drone) = res : Bool {
asegura $res == |vueloRealizado(d)| \geq 3 \land esEscalera(vueloRealizado(d))$;
}

aux esEscalera (ps: [(ℤ, ℤ)]) : Bool $= primerosTresEscalera(ps) \land (\forall i \leftarrow [0..|ps|-1))comoLosPrimerosTres(ps, i)$;
aux primerosTresEscalera (ps: [(ℤ, ℤ)]) : Bool $= |\mathsf{prm}(restarPos(ps_2, ps_0))| == 1 \land |\mathsf{sgd}(restarPos(ps_2, ps_0))| == 1$;
aux comoLosPrimerosTres (ps: [(ℤ, ℤ)], i: ℤ) : Bool $= restarPos(ps_{i+1}, ps_i) == restarPos(ps_{(i \text{ mód } 2)+1}, ps_{i \text{ mód } 2})$;

problema vuelosCruzadosD (ds: [Drone]) = res : [((ℤ,ℤ),ℤ)] {
    requiere todosEnVuelo : $(\forall d \leftarrow ds) enVuelo(d)$;
    requiere igualDeLargos : $(\forall d1 \leftarrow ds, d2 \leftarrow ds) |vueloReaizado(d1)| == |vueloRealizado(d2)|$;
    asegura hayCruce : $(\forall j \leftarrow [0..|res|)) sgd(res_j) > 1$;
    asegura cantidadDeCruces : $(\forall j \leftarrow [0..|res|)) sgd(res_j) == colisionesEnPosicion(ds, prm(res_j))$;
    asegura orden : $ordenadoAsc(res) \vee ordenadoDesc(res)$;
}

aux colisionesEnPosicion (ds: [Drone], pos: (ℤ, ℤ)) : ℤ = $|[d|d \leftarrow ds, i|i \leftarrow [0..|vueloRealizado(d)|), vueloRealizado(d)_i == pos \wedge colisionesEnInstanteEnPosicion(ds, pos, i) > 1]|$;

aux colisionesEnInstanteEnPosicion (ds: [Drone], pos: (ℤ, ℤ), i: ℤ) : ℤ = $|[d|d \leftarrow ds, vueloRealizado(d)_i == pos]|$;

aux ordenadoAsc (ps: [((ℤ,ℤ),ℤ)]) : Bool = $(\forall p \leftarrow [0..|ps| - 1)) sgd(res_p) \leq sgd(res_{p+1}))$;

aux ordenadoDesc (ps: [((ℤ,ℤ),ℤ)]) : Bool = $(\forall p \leftarrow [0..|ps| - 1)) sgd(res_p) \geq sgd(res_{p+1}))$;

4. Sistema

```
tipo Sistema {
        observador campo (s: Sistema) : Campo ;
        observador estadoDelCultivo (s: Sistema, i, j: ℤ) : EstadoCultivo ;
             requiere enRango(dimensiones(s), i, j) ∧ contenido(campo(s), i, j) == Cultivo ;
        observador enjambreDrones (s: Sistema) : [Drone] ;

        invariante identificadoresUnicos : sinRepetidos([ id(d) | d ← enjambreDrones(s) ]) ;
        invariante unoPorParcela : (∀d, d' ← dronesEnVuelo(s), id(d) ≠ id(d'))posicionActual(d) ≠ posicionActual(d') ;
        invariante siNoVuelanEstanEnGranero : (∀d ← enjambreDrones(s), ¬enVuelo(d))
            posicionActual(d) == posicionGranero(campo(s)) ;
        invariante siEstanEnVueloElVueloEstaEnRango : (∀d ← dronesEnVuelo(s))(∀v ← vueloRealizado(d))
            enRango(dimensiones(campo(s), prm(v), sgd(v)) ;
}
   aux dronesEnVuelo (s: Sistema) : [Drone] = [ d | d ← enjambreDrones(s), enVuelo(d) ] ;
problema crearS (c: Campo, ds: [Drone]) = res : Sistema {
}

problema campoS (s: Sistema) = res : Campo {
}

problema estadoDelCultivoS (s: Sistema, i, j: ℤ) = res : EstadoCultivo {
}

problema enjambreDronesS (s: Sistema) = res : [Drone] {
}

problema crecerS (s: Sistema) {
}

problema seVinoLaMalezaS (s: Sistema, ps: [(ℤ, ℤ)]) {
}

problema seExpandePlagaS (s: Sistema) {
}

problema despegarS (s: Sistema, d: Drone) {
}

problema listoParaCosecharS (s: Sistema) = res : Bool {
}

problema aterrizarYCargarBateriaS (s: Sistema, b: ℤ) {
}

problema fertilizarPorFilas (s: Sistema) {
}

problema volarYSensarS (s: Sistema, d: Drone) {
}
```

5. Funciones Auxiliares

aux `cuenta` (x: T, a: [T]) : $\mathbb{Z} = |[y|y \leftarrow a, y == x]|$;
aux `distancia` (a, b: (Ancho, Largo)) : $\mathbb{Z} = |prm(a) - prm(b)| + |sgd(a) - sgd(b)|$;
aux `sinRepetidos` (xs:[T]) : Bool $= (\forall x \leftarrow xs) cuenta(x, xs) == 1$;
aux `mismos` (a, b:[T]) : Bool $= (|a| == |b|) \wedge (\forall c \leftarrow a) cuenta(c, a) == cuenta(c, b)$;
aux `cola` (a : [T]) : [T] $= [a_i | i \leftarrow [1..|a|)]$;
aux `restarPos` (a,b: ($\mathbb{Z}$, $\mathbb{Z}$)) : ($\mathbb{Z}$, $\mathbb{Z}$) $= (\mathsf{prm}(a) - \mathsf{prm}(b), \mathsf{sgd}(a) - \mathsf{sgd}(b))$;

5.1. Campo

aux `alcanzableEn100` (posG: ($\mathbb{Z}$, $\mathbb{Z}$), a, l: $\mathbb{Z}$) : Bool $= (\forall i \leftarrow [0..a), j \leftarrow [0..l)) distancia(posG, (i, j)) \leq 100$;
aux `posicionGranero` (c: Campo) : ($\mathbb{Z}$, $\mathbb{Z}$) $= [(i, j) | i \leftarrow [0..prm(dimensiones(c))), j \leftarrow [0..sgd(dimensiones(c))), contenido(c, i, j) == Granero]_0$;
aux `enRango` (dim: (Ancho, Largo), i, j : $\mathbb{Z}$) : Bool $= 0 \leq i < prm(dim) \wedge 0 \leq j < sgd(dim)$;
aux `posPositiva` (p: ($\mathbb{Z}$, $\mathbb{Z}$)) : Bool $= \mathsf{prm}(p) \geq 0 \wedge \mathsf{sgd}(p) \geq 0$;

5.2. Drone

5.3. Sistema

6. Comentarios

6.1. Funciones Auxiliares

Por una cuestión de claridad, las funciones auxiliares genéricas y las que corresponden a cada tipo (Campo, Drone, Sistema) se encuentran en la sección Funciones Auxiliares, mientras que las auxiliares específicas de cada problema se encuentran debajo de las resoluciones de los mismos.

6.2. Uso de *mismos* en *vueloRealizadoD*

Dado que el enunciado pide 'la lista de las posiciones de las parcelas que recorrió el dron', consideramos que el orden no importa.

6.3. Requiere $|vueloRealizado(d)| \geq 3$ en *vueloEscaleradoD*

Consideramos que una escalera sin escalones no es tal, por lo tanto si el recorrido del vuelo no llega a formar ningún escalón, decimos que no se trata de un vuelo escalerado.

6.4. Sobre *vuelosCruzadosD*

La colisión de varios drones en la misma posición en distintos instantes se podría resolver de varias maneras. Debido a la respuesta a nuestra consulta, en esta especificación se suma el número de colisiones. Es decir, si en la posición (2, 3) se cruzaron dos drones en un instante y cuatro drones en otro instante, $((2,3),6) \in res$.

Si un solo dron estuvo en (2, 3) en un instante y en otro instante hubo un cruce de dos drones en (2, 3), $((2,3),2) \in res$ dado que la presencia de un solo dron no se considera cruce.