

Algoritmos y Estructura de Datos I

Primer cuatrimestre 2012

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

TP OJOTA (Organización de Juegos Olímpicos) v1.0

Grupo 4

Integrante	LU	Correo electrónico
Nicolas Lasso	763/10	lasso.nico@gmail.com
Guido Tripodi	843/10	guido.tripodi@hotmail.com
Tomas Agustin Shaurli	671/10	zeratulzero@hotmail.com
Patricio Inzaghi	255/11	pinzaghi@dc.uba.ar

1. Aclaraciones

1. En el ejercicio 4 requerimos que el ranking sea mayor a uno ya que sino no quedaria sin efecto la funcion auxiliar ya que de esta manera no se podrian comparar los atletas porque habria uno solo.
2. En el ejercicio 7, en caso de empate en medallas de oro, se compara las medallas de plata, si siguen siendo las mismas, se compara las medallas de bronce. En caso de tener las 3 medallas iguales se considera mejor al primero que se comparo. Si no hay ningun pais con medallas la lista resultante sera vacia.
3. En el ejercicio 12, consideramos a un patron, como la secuencia de paises hasta que aparece un elemento repetido sin incluirlo. Por Ej: [Argentina, USA, Canada, Argentina, Chile] el patron seria [Argentina, USA, Canada], en cambio en [Argentina, USA, USA, Argentina, Chile] el patron seria [Argentina, USA].

2. Resolucion

1. problema **entrenarNuevoDeporte** (a: Atleta, d: Deporte, c: \mathbb{Z}) {
 requiere $d \notin \text{deportes}(\text{pre}(a))$;
 requiere $0 \leq c \leq 100$;
 modifica a ;
 asegura $\text{nombre}(a) == \text{nombre}(\text{pre}(a))$;
 asegura $\text{sexo}(a) == \text{sexo}(\text{pre}(a))$;
 asegura $\text{añoNacimiento}(a) == \text{añoNacimiento}(\text{pre}(a))$;
 asegura $\text{naionalidad}(a) == \text{nacionalidad}(\text{pre}(a))$;
 asegura $\text{ciaNumber}(a) == \text{ciaNumber}(\text{pre}(a))$;
 asegura $\text{mismos}(\text{deportes}(a), \text{deportes}(\text{pre}(a)) : d)$;
 asegura $\text{ordenada}(\text{deportes}(a))$;
 asegura $\text{capacidad}(a, d) == c$;
}
2. problema **finalizarCompetencia** (c: Competencia, posiciones: [Atleta], control: [(Atleta, Bool)]) {
 requiere $|\text{participantes}(c)| > 0$;
 requiere $(\forall a \in \text{control}) \text{prm}(a) \in \text{participantes}(c)$;
 requiere $(\forall a \in \text{posiciones}) a \in \text{participantes}(c)$;
 requiere $\text{finalizada}(c) == \text{False}$;
 modifica c ;
 asegura $\text{categoria}(c) == \text{categoria}(\text{pre}(c))$;
 asegura $\text{mismos}(\text{participantes}(c), \text{participantes}(\text{pre}(c)))$;
 asegura $\text{finalizada}(c)$;
 asegura $(\forall a \in \text{ranking}(c)) a \in \text{participantes}(c)$;
 asegura $\text{ranking}(c) == \text{posiciones}$;
 asegura $(\forall a \in \text{control}) \text{prm}(a) \in \text{lesTocoControlAntiDoping}(c)$;
 asegura $(\forall a \in \text{lesTocoControlAntiDoping}(c)) a \in \text{participantes}(c)$;
 asegura $\text{mismos}([\text{prm}(d) | d \leftarrow \text{control}], \text{lesTocoControlAntiDoping}(c))$;
 asegura $(\forall x \leftarrow \text{control}), \text{lesDioPositivo}(c, \text{prm}(x)) == \text{sgd}(x)$;
}
3. problema **linfordChristie** (c: Competencia, a: Atleta) {
 requiere $\text{finalizada}(c) == \text{False}$;
 requiere $\text{atletaPertenezca} : a \in \text{participantes}(c)$;
 modifica c ;
 asegura $\text{categoria}(c) == \text{categoria}(\text{pre}(c))$;
 asegura $\text{atletaNoPertenece} : a \notin \text{participantes}(c)$;
 asegura $\text{long}(\text{participantes}(c)) == \text{long}(\text{participantes}(\text{pre}(c)) - 1$;
 asegura $\text{mismos}(a : \text{participantes}(c), \text{participantes}(\text{pre}(c)))$;
}

4. problema `gananLosMasCapaces` (c: Competencia) = result : Bool {
 requiere *finalizada*(c) == True;
 requiere | *ranking*(c) | > 1;
 requiere | *participantes*(c) | > 1;
 asegura ($\forall a \in \text{ranking}(c) a \in \text{participantes}(c)$);
 asegura *Result* == *masCapaces*(c);
 }
5. problema `sancionarTramposos` (c: Competencia) {
 requiere *finalizada*(c) == true;
 modifica c;
 asegura *categoria*(c) == *categoria*(pre(c));
 asegura *finalizada*(c) == *finalizada*(pre(c));
 asegura *participantes*(c) == *participantes*(pre(c));
 asegura *lesTocoControlAntiDoping*(c) == *lesTocoControlAntiDoping*(pre(c));
 asegura ($\forall a \in \text{participantes}(c) \text{leDioPositivo}(c, a) == \text{leDioPositivo}(\text{pre}(c), a)$);
 asegura *ranking*(c) == *borrarPositivos*(pre(c));
 aux *borrarPositivos* (c:competencia) : [Atleta] = $[x | x \leftarrow \text{ranking}(c), (\forall y \leftarrow \text{listaPositivos}(c)) y \neq x]$;
 }
6. problema `dePaseo` (j: JJOO) = result : [Atleta] {
 requiere | *atletas*(j) | > 0;
 asegura *result* == $[x | y \leftarrow \text{cantDias}(j), k \leftarrow \text{cronograma}(j, y), x \leftarrow \text{atletas}(j), x \notin \text{participantes}(k)]$;
 }
7. problema `medallero` (j: JJOO) = result : [(País, [Z])] {
 requiere | *atletas*(j) | > 0;
 asegura *paísesConMedallas* : ($\forall p \leftarrow \text{result}, \text{algunaMedalla}(\text{sgd}(p))$);
 asegura *soloTresTiposDeMedallas* : ($\forall p \leftarrow \text{result}, |\text{sgd}(p)| == 3$);
 asegura *mejoresMedallasQueSiguiente* :
 ($\forall x \leftarrow [0..|\text{result}| - 1], \text{mejoresOIgualesMedallas}(\text{sgd}(\text{result}_x), \text{sgd}(\text{result}_{x+1}))$);
 aux *mejoresOIgualesMedallas* (m: [Z], p: [Z]) : Bool = *superaPorOro*(m, p);
 aux *superaPorOro* (m: [Z], p: [Z]) : Bool =
 ifThenElse($m_0 > p_0$, True, *ifThenElse*($m_0 == p_0$, *superaPorPlata*(m, p), False));
 aux *superaPorPlata* (m: [Z], p: [Z]) : Bool =
 ifThenElse($m_1 > p_1$, True, *ifThenElse*($m_1 == p_1$, *superaPorBronce*(m, p), False));
 aux *superaPorBronce* (m: [Z], p: [Z]) : Bool =
 ifThenElse($m_2 > p_2$, True, *ifThenElse*($m_2 == p_2$, True, False));
 aux *algunaMedalla* (m: [Z]) : Bool = ($m_0 \neq 0 \vee m_1 \neq 0 \vee m_2 \neq 0$);
 }
8. problema `boicotPorDisciplina` (j: JJOO, cat: (Deporte, Sexo), p: País) = result : Z {
 requiere *long*(*atletas*(j)) > 0;
 requiere ($\forall a \in \text{atletasEnCategoria}(j, \text{cat}, p) \text{nacionalidad}(a) == p$);
 requiere ($\exists c \in \text{competencias}(j) \text{categoria}(c) == \text{cat}$);
 modifica j;
 asegura *año*(j) == *año*(pre(j));
 asegura *cantDias*(j) == *cantDias*(pre(j));
 asegura *misimos*(*cronograma*(j), *cronograma*(pre(j)));
 asegura *jornadaactual*(j) == *jornadaactual*(pre(j));
 asegura ($\forall a \in \text{atletasEnCategoria}(j, \text{cat}, p) a \notin \text{atletas}(j)$);
 asegura *result* == | *atletasEnCategoria*(j, cat, p) |;
 }

9. problema losMasFracasados (j: JJOO, p: País) = result : [Atleta] {
 requiere $(\exists a \in atletas(j))nacionalidad(a) == p$;
 aux atletasDelPais (j:JJOO,p: País) : [Atleta] = $[x|x \leftarrow atletas(j), nacionalidad(x) == p]$;
 aux competenciasAtleta (j:JJOO, p:País, a:Atleta) : [Competencia] = $[x|x \leftarrow competencias(j), alguno(participantes(x) == a) \wedge en(prm(categoria(x)), deportes(a))]$;
 aux atletasFracasados (j:JJOO, p:País) : [(atleta, \mathbb{Z})] = $[(y, |competenciasAtleta(j, p, y)|)| y \leftarrow atletasDelPais(j, p), c \leftarrow competenciasAtleta(j, y, p), en(y, sub(ranking(c), 3, |ranking(c) - 1|))]$;
 aux masFracasados (j:JJOO, p:País) : [Atleta] = $[prm(y)|y \leftarrow atletasFracasados(j, p), k \leftarrow atletasFracasados(j, p), sgd(y) \geq sgd(k)]$;
 asegura result == masFracasados(j, p);
 }
10. problema liuSong (j: JJOO, a: Atleta, p: País) {
 requiere $a \in atletas(j)$;
 modifica a, j;
 asegura nombre(a) == nombre(pre(a));
 asegura sexo(a) == sexo(pre(a));
 asegura añoNacimiento(a) == añoNacimiento(pre(a));
 asegura ciaNumber(a) == ciaNumber(pre(a));
 asegura mismos(deportes(a), deportes(pre(a)));
 asegura cambioNacionalidad : nacionalidad(a) == p;
 asegura año(j) == año(pre(j));
 asegura cantDias(j) == cantDias(pre(j));
 asegura mismos(cronograma(j), cronograma(pre(j)));
 asegura jornadaActual(j) == jornadaActual(pre(j));
 asegura $|atletas(j)| == |atletas(pre(j))|$;
 asegura mismosAtletas : $(\forall x \in atletas(pre(j)))ciaNumber(x) \in ciaNumbers(j)$;
 }
11. problema stevenBradbury (j: JJOO) = result : Atleta {
 requiere algunaCompetenciaFinalizada(j);
 asegura esElMenosCapaz :
 $(\forall a \in atletasConOro(j), a \neq result)sumaDeCapacidades(result) \leq sumaDeCapacidades(a)$;
 aux atletasConOro (j:JJOO) : [Atleta] = $[ranking(c)_0 | c \in competencias(j)]$;
 aux sumaDeCapacidades (a:Atleta) : $\mathbb{Z} = \sum[capacidad(d) | d \in deportes(a)]$;
 }
12. problema uyOrdenadoAsíHayUnPatrón (j: JJOO) = result : Bool {
 asegura result == verificaPaíses(j);
 aux competenciasPorDia (j:JJOO) : $[(\mathbb{Z}, [Competencia])]$ = $[(dia, c) | dia \leftarrow [1..jornadaActual(j)], c == mismos(cronograma(j, dia))]$;
 aux oroPorDia (j:JJOO) : $[(\mathbb{Z}, [Atleta])]$ = $[(dia, a) | h \leftarrow competenciasPorDia(j), dia \leftarrow prm(h), x \leftarrow sgd(h), i \leftarrow [0..|x| - 1], a \leftarrow ranking(x_i)_0]$;
 aux paisesPorDia (j:JJOO) : $[(\mathbb{Z}, [País])]$ = $[(dia, p) | x \leftarrow oroPorDia(j), dia \leftarrow prm(x), h \leftarrow sgd(x), i \leftarrow [0..|h| - 1], p \leftarrow nacionalidad(h_i)]$;
 aux mejoresPaíses (j:JJOO) : $[(\mathbb{Z}, [País])]$ = $[(dia, p) | x \leftarrow paisesPorDia(j), dia \leftarrow prm(x), p \leftarrow sgd(x), i \leftarrow [0..|sgd(x)| - 2], (cuenta(p, sgd(x)) \geq cuenta(sgd(x)_{i+1}, sgd(x)))]$;
 aux paisesMasGrosos (j:JJOO) : [País] = $[p | x \leftarrow mejoresPaíses(j), p \leftarrow sgd(x), i \leftarrow [0..|sgd(x)| - 2], p \leq sgd(x)_{i+1}]$;
 aux patronDePaíses (j:JJOO) : [País] = $[sub(paisesMasGrosos(j), 0, i - 2) | i \leftarrow [0..|paisesMasGrosos| - 1], \neg sinRepetidos(sub(paisesMasGrosos(j), 0, i))_0]$;
 aux verificaPaíses (j:JJOO) : Bool = $\forall i \in [0..|paisesMasGrosos(j)| - 1], paisesMasGrosos(j)_i == paisesMasGrosos(j)_{i+|patronDePaíses(j)|} \wedge ((i + |patronDePaíses(j)|) \leq |paisesMasGrosos(j)|)$;
 }

```

13. problema sequíaOlimpica (j: JJOO) = result : [País] {
  aux atletasDelPais (j:JJOO,p: País) : [Atleta] = [x|x ← atletas(j), nacionalidad(x) == p];
  aux todosLosPaises (J:JJOO) : [(País,[Bool])] = [(nacionalidad(x), [alguno(h ← (ranking(y)[0, 1, 2])
    k ← (atletasDelPais(j, nacionalidad(x))), k == h))]|x ← atletas(j), i ← [0..cantDias(j)], y ← cronograma(j, i)];
  aux paisesConDiasSep (j:JJOO) : [(Pais,[[Bool]])] = [(prm(x), [ifThenElse(sgd(x)i ≠ sgd(x)k,
    sub(todosLosPaises(j), i, k-1) ∧ i == k))]|x ← todosLosPaises(j), i = 0, k ← [0..|sgd(todosLosPaises(j))|-1]];
  aux elijoLosFracasados (j:JJOO) : [(pais,[[Bool]])] = [(prm(x), [y])
    |x ← paisesConDiasSep(j), (∀h ← y ← sgd(x))(h == False)];
  aux mayorCantidadDeDias (j:JJOO) : [(pais, ℤ)] = [(prm(x), long(y))
    |x ← elijoLosFracasados(j), y ← sgd(x), h ← sgd(x), long(y) ≥ long(h)];
  aux elPeorPais (j:JJOO) : [pais] = [prm(x)|x ← mayorCantidadDeDias(j),
    y ← mayorCantidadDeDias(j), sgd(x) ≥ sgd(y)];
  asegura result == elPeorPais(j);
}

problema sequíaOlimpica (j: JJOO) = result : [País] {
}

14. problema transcurrirDia (j: JJOO) {
  requiere jornadaActual(j) < cantDias(j);
  modifica j;
  asegura cantDias(j) == cantDias(pre(j));
  asegura año(j) == año(pre(j));
  asegura mismos(atletas(j), atletas(pre(j)));
  asegura (∀d ∈ [1...cantDias(pre(j))])cronograma(pre(j), d) == cronograma(j, d));
  asegura cambiaElDia : jornadaActual(j) == jornadaActual(pre(j)) + 1;
  asegura lasPasadasFinalizaron(pre(j));
  asegura rankingSegunMasCapaces : (∀c ∈ cronograma(pre(j), jornadaActual(pre(j))))masCapaces(c);
  asegura unControlPorCompetencia : (∀c ∈ cronograma(pre(j), jornadaActual(pre(j)))
    |lesTocaControlAntiDoping(c)| == 1);
  asegura dopingPositivoMax5Porciento : (∃a ∈ atletasLesTocoAntiDoping(j, jornadaActual(pre(j))))
    0 ≤ porcentajePositivos(j) ≤ 5);
  aux porcentajePositivos (j:JJOO) : ℝ = |atletasLesDioPositivo(j, jornadaActual(pre(j)))| * 100
    /|atletasLesTocoAntiDoping(j, jornadaActual(pre(j)))|;
  aux atletasLesTocoAntiDoping (j:JJOO,d:ℤ) : [Atleta] = [lesTocoControlAntiDoping(c)0|c ∈ cronograma(j, d)];
  aux atletasLesDioPositivo (j:JJOO,d:ℤ) : [Atleta] = [lesTocoControlAntiDoping(c)0|c ∈ cronograma(j, d),
    lesTocoControlAntiDoping(c)0 ∈ leDioPositivo(c)];
}

```

3. Tipos

```
tipo Deporte = String ;
tipo Pais = String ;
tipo Sexo = Femenino, Masculino ;
```

4. Atleta

```
tipo Atleta {
  observador nombre (a: Atleta) : String ;
  observador sexo (a: Atleta) : Sexo ;
  observador añoNacimiento (a: Atleta) :  $\mathbb{Z}$  ;
  observador nacionalidad (a: Atleta) : Pais ;
  observador ciaNumber (a: Atleta) :  $\mathbb{Z}$  ;
  observador deportes (a: Atleta) : [Deporte] ;
  observador capacidad (a: Atleta, d: Deporte) :  $\mathbb{Z}$  ;
    requiere  $d \in \text{deportes}(a)$  ;

  invariante sinRepetidos(deportes(a)) ;
  invariante ordenada(deportes(a)) ;
  invariante capacidadEnRango :  $(\forall d \leftarrow \text{deportes}(a)) 0 \leq \text{capacidad}(a, d) \leq 100$  ;
}
```

5. Competencia

```
tipo Competencia {
  observador categoria (c: Competencia) : (Deporte, Sexo) ;
  observador participantes (c: Competencia) : [Atleta] ;
  observador finalizada (c: Competencia) : Bool ;
  observador ranking (c: Competencia) : [Atleta] ;
    requiere finalizada(c) ;
  observador lesTocoControlAntiDoping (c: Competencia) : [Atleta] ;
    requiere finalizada(c) ;
  observador leDioPositivo (c: Competencia, a: Atleta) : Bool ;
    requiere finalizada(c)  $\wedge a \in \text{lesTocoControlAntiDoping}(c)$  ;

  invariante participaUnaSolaVez : sinRepetidos(ciaNumbers(participantes(c))) ;
  invariante participantesPerteneceenACat :
     $(\forall p \leftarrow \text{participantes}(c)) \text{prm}(\text{categoria}(c)) \in \text{deportes}(p) \wedge \text{sgd}(\text{categoria}(c)) == \text{sexo}(p)$  ;
  invariante elRankingEsDeParticipantesYNoHayRepetidos :
    finalizada(c)  $\Rightarrow \text{incluida}(\text{ranking}(c), \text{participantes}(c))$  ;
  invariante seControlanParticipantesYNoHayRepetidos :
    finalizada(c)  $\Rightarrow \text{incluida}(\text{lesTocoControlAntiDoping}(c), \text{participantes}(c))$  ;
}
```

6. JJOO

```
tipo JJOO {
  observador año (j: JJOO) :  $\mathbb{Z}$  ;
  observador atletas (j: JJOO) : [Atleta] ;
  observador cantDias (j: JJOO) :  $\mathbb{Z}$  ;
  observador cronograma (j: JJOO, dia:  $\mathbb{Z}$ ) : [Competencia] ;
    requiere  $1 \leq \text{dia} \leq \text{cantDias}(j)$  ;
  observador jornadaActual (j: JJOO) :  $\mathbb{Z}$  ;

  invariante atletasUnicos : sinRepetidos(ciaNumbers(atletas(j))) ;
  invariante unaDeCadaCategoria :  $(\forall i, k \leftarrow [0..|\text{competencias}(j)|], i \neq k)$ 
    categoria(competencias(j)i)  $\neq$  categoria(competencias(j)k) ;
  invariante competidoresInscriptos :  $(\forall c \leftarrow \text{competencias}(j)) \text{incluida}(\text{participantes}(c), \text{atletas}(j))$  ;
  invariante jornadaValida :  $1 \leq \text{jornadaActual}(j) \leq \text{cantDias}(j)$  ;
  invariante finalizadasSiiYaPasoElDia : lasPasadasFinalizaron(j)  $\wedge$  lasQueNoPasaronNoFinalizaron(j) ;
}
```

}

7. Auxiliares

```
aux atletasEnCategoria (j:JJOO, cat: Categoria, p: País ) : [Atleta] = [a | a ∈ atletas(j), nacionalidad(a) == p ∧
    sexo(a) == sgd(cat) ∧ prm(cat) ∈ deportes(a)];
aux algunaCompetenciaFinalizada (j:JJOO) : Bool = (∃ dia ∈ [1..cantDias(j)], cron ∈ cronograma(j, dia), comp ∈
    cron) finalizada(comp);
aux ciaNumbers (as: [Atleta]) : [ℤ] = [ciaNumber(a) | a ← as];
aux competencias (j: JJOO) : [Competencia] = [c | d ← [1..cantDias(j)], c ← cronograma(j, d)];
aux competenciasFinalizadas (j: JJOO) : [Competencia] = [c | C ← [1..cantDias(j)], c ← cronograma(j, d), finalizada(c)];
aux incluida (l1, l2: [T]) : Bool = (∀ x ← l1) cuenta(x, l1) ≤ cuenta(x, l2);
aux lasPasadasFinalizaron (j: JJOO) : Bool = (∀ d ← [1..jornadaActual(j)]) (∀ c ← cronograma(j, d)) finalizada(c);
aux lasQueNoPasaronNoFinalizaron (j: JJOO) : Bool =
    (∀ d ← (jornadaActual(j)..cantDias(j))) (∀ c ← cronograma(j, d)) ¬ finalizada(c);
aux listaPositivos (c: competencia) : [Atleta] = [lesTocoControlAntiDoping(c), lesDioPositivo(c, x) == True];
aux ordenada (l: [T]) : Bool = (∀ i ← [0..|l| - 1]) li ≤ li+1;
aux sinRepetidos (l: [T]) : Bool = (∀ i, j ← [0..|l|], i ≠ j) li ≠ lj;
aux masCapaces (c: competencia) : Bool = (∀ x ← [0..|ranking(c)| - 1], capacidad(ranking(c)x) > capacidad(ranking(c)x+1));
aux atletasdelpaisout (cat: (Deporte, Sexo), p: Pais, j: JJOO) : Bool = (x ← [0..|atletas(j)| - 1], p(cat(atletas(j)x)) ≠
    atletas(j)x);
```