DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

# Trabajo práctico

## Examen de Residencia Version GPT5 Iroquai

September 22, 2025 — Algoritmos y Estructuras de Datos

**Algoritmia**


| Integrante | LU | Correo electrónico |
|---|---|---|
| Lopez Abraham, Ignacio Matias | 415/25 | ignaciolopezabraham05@gmail.com |
| Ferrario, Victoria | 395/25 | vickyferrario56@gmail.com |
| Ramdan, Matias | 326/21 | matias.ramdan@gmail.com |
| Potenzoni, Mateo | 452/25 | matepotenzoni@gmail.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

$Solucion\ ES$ seq$\langle\mathbb{Z}\rangle$
$Nota\ ES\ \mathbb{Z}$
$Estudiante\ ES$ $\langle$Solucion : Solucion, Posicion : tuple$\langle\mathbb{Z},\mathbb{Z}\rangle$, EnAula : bool$\rangle$

TAD EdR {
obs SolucionCanonica : $Solucion$
obs Estudiantes : conj$\langle Estudiante\rangle$
obs ListaCopiados : conj$\langle Estudiante\rangle$
obs FilaColumnaAula : $\mathbb{Z}$

proc EdR(in $Filas$ : $\mathbb{Z}$, in $SolucionCanonica$ : Solucion, in $CantidadEstudiantes$ : $\mathbb{Z}$) : EdR {
requiere {
$(CantidadEstudiantes > 0 \land Filas > 0) \land_L$
$entranEstudiantes(filas, CantidadEstudiantes) \land$
$solucionCumple(SolucionCanonica)$
}
asegura {
$res.SolucionCanonica = SolucionCanonica \land$
$res.FilaColumnaAulas = Filas \land res.Estudiantes =$ conj$\langle Estudiante\rangle \land$
$EstudiantesCumple(res.Estudiante, CantidadEstudiantes, SolucionCanonica, filas) \land$
$res.ListaCopiados =$ conj$\langle Estudiante\rangle \land |res.ListaCopiados| = 0$
}
}
pred entranEstudiantes($filas : \mathbb{Z}, cantidadEstudiantes : \mathbb{Z}$) {
$((filas/2) * filas) \geq cantidadEstudiantes$
}
pred solucionCumple($solucionCanonica$ : Solucion) {
$(\forall punto : \mathbb{Z})$ (
$(0 \leq punto < |solucionCanonica|) \rightarrow_L ((solucionCanonica[punto]) \geq 0 \land$
$(solucionCanonica[punto]) \leq 9)$
)
}
pred estudiantesCumple($estudiantes$ : seq$\langle$Estudiante$\rangle$, $cantidadEstudiantes : \mathbb{Z}$, $solucion$ : Solucion, $filas : \mathbb{Z}$) {
$|estudiantes| = cantidadEstudiantes \land_L$
$(\forall estudiante$ : Estudiante) (
$(estudiante \in estudiantes) \rightarrow_L (|estudiante.Solucion| = |solucion|) \land$
$(\forall punto : \mathbb{Z})$ (
$(0 \leq punto < |estudiante.Solucion|) \rightarrow estudiante.Solucion[punto] = -1$
) $\land$
$(0 \leq estudiante.Posicion_1 < filas) \land (0 \leq estudiante.Posicion_2 < filas) \land$
$(\forall estudiante'$ : Estudiante) (
$estudiante' \in estudiantes \rightarrow_L (estudiante' \neq estudiante \land$
$estudiante'.Posicion \neq estudiante.Posicion)$
) $\land (estudiante'.Posicion_1 = estudiante.Posicion_1) \rightarrow$
$|estudiante'.Posicion_2 - estudiante.Posicion_2| \geq 2$
)
}

proc igualdad(in $EdR1$ : EdR, in $EdR2$ : EdR) {
  requiere { $True$ }
  asegura {
    $EdR1.SolCanonica = EdR2.SolCanonica \land$
    $EdR1.Estudiantes = EdR2.Estudiantes \land$
    $EdR1.ListaCopiados = EdR2.ListaCopiados \land$
    $EdR1.FilaColumnaAula = EdR2.FilaColumnaAula$
  }
}
proc copiarse(inout $EdR$ : EdR, in $estudianteCopiado$ : Estudiante, in $estudianteCopiador$ : Estudiante, in $puntoCopiado : \mathbb{Z}$) {
  requiere {
    $EdR = EdR_0 \land$
    $(estudianteCopiado \in EdR.Estudiantes \land estudianteCopiador \in EdR.Estudiantes) \land$
    $(estudianteCopiado.enAula = True \land estudianteCopiador.enAula = True) \land_L$
    $CopiadorEnRango(estudianteCopiado, estudianteCopiador) \land$
    $puntoValidoASerCopiado(EdR, puntoCopiado, estudianteCopiado, estudianteCopiador)$
  }
  asegura {
    $copiadorCambiaSoloPuntoCopiado(EdR, puntoCopiado,$
    $estudianteCopiador, estudianteCopiado) \land$
    $EdR.SolCanonica = EdR_0.SolCanonica \land$
    $EdR.ListaCopiados = EdR_0.ListaCopiados \land$
    $restoEstudiantesNoCambia(EdR, EdR_0, estudianteCopiador)$
  }
}
pred CopiadorEnRango($estudianteCopiado$ : Estudiante, $estudianteCopiador$ : Estudiante, $fila : \mathbb{Z}$) {
  $\Big( (estudianteCopiador.posicion_1 \neq 0) \land$
  $\big(estudianteCopiador.posicion_2 = estudianteCopiado.posicion_2 \land$
  $estudianteCopiador.posicion_1 = estudianteCopiado.posicion_1 + 1\big)$
  $\Big) \lor \Big($
  $(fila > 2) \land$
  $\big((estudianteCopiador.posicion_2 = 0 \land$
  $estudianteCopiador.posicion_2 + 2 = estudianteCopiado.posicion_2)$
  $\lor$
  $(estudianteCopiador.posicion_2 = fila - 1 \land$
  $estudianteCopiador.posicion_2 - 2 = estudianteCopiado.posicion_2)$
  $\lor$
  $(|estudianteCopiador.posicion_2 - estudianteCopiado.posicion_2| = 2 \land \,)\Big)$
}
pred puntoValidoASerCopiado($EdR : EdR, p : \mathbb{Z}, estudianteCopiado : Estudiante,$ $estudianteCopiador : Estudiante$) {
  $(0 < p \leq |EdR.SolCanonica|) \land_L$
  $(\exists estudiante : Estudiante)\ (estudiante \in EdR.Estudiantes \land_L$
  $(estudiante = estudianteCopiado \land (0 \leq estudiante.Solucion[p] \leq 9)))$
  $\land$
  $(\exists estudiante' : Estudiante)\ (estudiante' \in EdR.Estudiantes \land_L$

$(estudiante' = estudianteCopiador \land estudiante'.Solucion = -1))$
}
pred copiadorCambiaSoloPuntoCopiado($EdR$ : EdR, $p : \mathbb{Z}$, $estudianteCopiado$ : Estudiante, $estudianteCopiador$ : Estudiante) {
$(\exists estudiante$ : Estudiante) $(estudiante \in EdR.Estudiantes \land_L$
$((estudiante.Posicion = estudianteCopiador.Posicion) \land$
$(estudiante.enAula = estudianteCopiado.enAula) \land_L$
$(\exists estudiante'$ : Estudiante) $(estudiante' \in EdR.Estudiantes \land_L$
$(estudiante' = estudianteCopiado \land$
$estudiante.Solucion = setAt(estudianteCopiador.Solucion, (p),$
$estudiante'.Solucion[p])))))$
}
pred restoEstudiantesNoCambia($Edr_0$ : Edr, $Edr$ : Edr, $estudianteQueNoCambia$ : Estudiante) {
$|Edr_0| = |Edr| \land_L$
$(\forall estudiante$ : Estudiante) (
$(estudiante \in Edr_0.estudiantes \land estudiante \neq estudianteQueNoCambia) \rightarrow_L$
$(estudiante \in Edr_0.estudiantes)$
)
}
proc consultarDarkWeb(in $solucionDarkWeb$ : Solucion, inout $EdR$ : EdR) {
requiere {
$SolucionCumple(solucionDarkWeb) \land |solucionDarkWeb| = |EdR.SolucionCanonica| \land$
$EdR = EdR_0$
}
asegura {
$consultaronDarkWeb(EdR, solucionDarkWeb) \land$
$|EdR.Estudiantes| = |EdR_0.Estudiantes| \land$
$EdR.SolucionCanonica = EdR_0.SolucionCanonica \land$
$EdR.ListaCopiados = EdR_0.ListaCopiados \land$
$EdR.FilaColumnaAula = EdR_0.FilaColumnaAula \land$
$(\forall estudiante$ : Estudiantes) (
$estudiante \in EdR_0.Estudiantes \rightarrow estudiante \in EdR.Estudiantes \lor$
$cambioElEstudiante(EdR, solucionDarkWeb, estudiante)$
)
}
}
pred consultaronDarkWeb($EdR$ : EdR, $solucionDarkWeb$ : Solucion) {
$(\exists conjuntoEstudiantes$ : conj⟨Estudiante⟩) (
$(conjuntoEstudiantes \in EdR.Estudiantes) \land$
$(0 < |conjuntoEstudiantes| < |EdR.Estudiantes|) \land$
$(\forall estudiante$ : Estudiante) (
$estudiante \in conjuntoEstudiantes \rightarrow estudiante.Solucion = solucionDarkWeb$
)
)
}

pred cambioElEstudiante($Edr$ : Edr, $nuevaSolucion$ : Solucion, $estudiante$ : Estudiante)
{
$(\exists estudiante : Edr.Estudiantes)\ ((estudiante.EnAula = estudiante.EnAula) \wedge_L$
$(estudiante.Posicion = estudiante.Posicion) \wedge_L (estudiante.Solucion = nuevaSolucion))$
}
proc resolver(inout $EdR$ : EdR, in $pasos$ : seq$\langle Solucion \rangle$, in $puntoAResolver$ : $tuple\langle \mathbb{Z}, \mathbb{Z} \rangle$,
in $estudiante$ : Estudiante) : seq$\langle Solucion \rangle$ {
requiere {
$EdR = Edr_0 \wedge |pasos| \geq 0 \wedge estudiante.enAula = True \wedge_L$
$EstudianteEnEdr(EdR.Estudiantes, estudiante) \wedge$
$examenVacio(estudiante.solucion) \wedge$
$pasosValidos(pasos, EdR.SolucionCanonica) \wedge$
$\neg puntoResuelto(pasos[|pasos| - 1], puntoAResolver)$
}
asegura {
$todoIgualMenosEstudiantes(EdR, EdR_0) \wedge$
$res = pasos + +setAt(pasos[|pasos| - 1], puntoAResolver_1, puntoAResolver_2) \wedge$
$cambioRespuestaEstudiante(EdR_0, EdR, res[|res| - 1], estudiante) \wedge$
$restoEstudiantesNoCambia(EdR_0, EdR, estudiante)$
}
}
pred examenVacio($examen$ : Solucion) {
$(\forall paso : \mathbb{Z})\ (0 \leq paso < |examen| \rightarrow_L examen[paso] = -1)$
}
pred pasosValidos($pasos$ : seq$\langle$Solucion$\rangle$, $solucion$ : Solucion) {
$(\forall i : \mathbb{Z})\ (0 \leq i < |pasos| \rightarrow_L (|pasos| = |solucion| \wedge cantidadPuntosHechos(pasos[i]) =$
$i))$
}
pred cambioRespuestaEstudiante($EdR$ : EdR, $nuevaSolucion$ : Solucion, $estudiante$ :
Estudiante) {
$(\exists estudiante'$ : Estudiante$)\ (estudiante' \in EdR.Estudiantes \wedge$
$estudiante'.EnAula = estudiante.EnAula \wedge$
$estudiante'.Posicion = estudiante.Posicion \wedge$
$estudiante'.Solucion = nuevaSolucion)$
}
pred todoIgualMenosEstudiantes($EdR1$ : Edr, $EdR2$ : EdR) {
$EdR1.FilaColumnaAula = EdR2.FilaColumnaAula \wedge$
$EdR1.ListaCopiados = EdR2.ListaCopiados \wedge$
$EdR1.SolucionCanonica = EdR2.SolucionCanonica$
}
aux puntoResuelto($examen$ : Solucion, $punto$ : $\mathbb{Z}$) : bool $= 0 \leq examen[punto] \leq 9$
aux cantidadPuntosHechos($solucion$ : Solucion) : $\mathbb{Z}$ =
$\sum_{i=0}^{|solucion|-1}$ IfThenElse$(0 \leq solucion[i] \leq 9, 1, 0)$

```
proc entregar(inout EdR : EdR, in estudianteEntregar : Estudiante) {
   requiere {
        estudianteEntregar ∈ EdR.Estudiantes ∧ estudianteEntregar.EnAula = True
   }
   asegura {
        SaleEstudiante(EdR, EdR_0, estudianteEntregar) ∧
        restoEstudiantesNoCambia(EdR_0, EdR, estudianteEntregar) ∧
        EdR.SolucionCanonica = EdR_0.SolucionCanonica ∧
        EdR.ListaCopiados = EdR_0.ListaCopiados ∧
        EdR.FilaColumnaAula = EdR_0.FilaColumnaAula
   }
}
pred SaleEstudiante(EdR : EdR, estudianteEntregar : Estudiante) {
    (∃est : Estudiante) (est ∈ EdR.Estudiantes∧(est.Posicion = estudianteEntregar.Posicion)∧
    (est.Solucion = estudianteEntregar.Solucion) ∧ (est.EnAula = False))
}
proc chequearCopias(inout EdR : EdR) : seq⟨Estudiante⟩ {
   requiere {
        EdR = EdR_0 ∧ terminoExamen(EdR.Estudiantes)
   }
   asegura {
        noHayRepetidos(res) ∧
        (∀estudiante : Estudiante) (
             estudiante ∈ EdR.Estudiantes →
             (seCopioEnRango(estudiante, EdR.Estudiantes) ∨
             seCopioDarkWeb(estudiante, EdR.Estudiantes)) ↔ estudiante ∈ res
        ) ∧ todosIgualesMenosListaCopiados(EdR_0, EdR) ∧ EdR.ListaCopiados = res
   }
}
pred terminoExamen(estudiantes : conj⟨Estudiante⟩) {
    (∀estudiante : Estudiante) (
         estudiante ∈ estudiantes → estudiante.EnAula = False
    )
}
pred seCopioEnRango(estudiante : Estudiante, estudiantes : conj⟨Estudiante⟩) {
    (∃estudianteCopiado : Estudiante) (
         estudianteCopiado ∈ estudiantes ∧
         CopiadorEnRango(estudianteCopiado, estudiante) ∧
         cantidadDeIgualesEnSeq(estudiante.Solucion, estudianteCopiado.Solucion) ≥
         ((|estudianteCopiado|/10) ∗ 6)
    )
}
pred seCopioDarkWeb(estudiante : Estudiante, estudiantes : conj⟨Estudiante⟩) {
    (∃estudiantesCopiados : conj⟨Estudiante⟩) (
         estudiantesCopiados ⊂ estudiantes ∧ estudiante ∈ estudiantesCopiados →_L
         (∀estudianteCopiado : Estudiante) (
              estudianteCopiado ∈ estudiantesCopiados →
              mismaSolucion(estudianteCopiado, estudiante)
         ) ∧ |estudiantesCopiados| ≥ |estudiantes|/4
```

```
        )
    }
    pred todosIgualesMenosListaCopiados(EdR1 : EdR, EdR2 : EdR)  {
        EdR1.SolucionCanonica = EdR2.SolucionCanonica ∧
        EdR1.Estudiantes = EdR2.Estudiantes ∧
        EdR1.FilaColumnaAula = EdR2.FilaColumnaAula
    }
    pred noHayRepetidos(secuencia : seq⟨T⟩)  {
        (∀i : ℤ) ((∀j : ℤ) (((0 ≤ i < |secuencia|) ∧ (0 ≤ j < |secuencia|) ∧ i ≠ j) →L
        secuencia[i]1 ≠ secuencia[j]1))
    }
    aux mismaSolucion(estudiante1 : Estudiante, estudiante2 : Estudiante) : bool =
        estudiante1.Solucion = estudiante2.Solucion
    aux cantidadDeIgualesEnSeq(elemento : T, secuencia : seq⟨T⟩) : ℤ =
        ∑_{i=0}^{|secuencia|−1} IfThenElse(elemento = secuencia[i], 1, 0)
    proc corregir(in EdR : EdR) : seq⟨tuple⟨Estudiante, ℝ⟩⟩ {
       requiere { terminoExamen(EdR.Estudiantes) }
       asegura {
           listaSolucionNoContieneNombresDeMas(res, EdR) ∧
           noHayRepetidos(res) ∧
           (∀estudiante : Estudiante) (
               (estudiante ∈ EdR.Estudiantes ∧ ¬(estudiante ∈ EdR.ListaCopiados))
               → (∃i : ℤ) (
                   (0 ≤ i < |res|) ∧L (res[i]1 = estudiante) ∧
                   (res[i]2 = notaDelEstudiante(estudiante.Solucion, EdR.SolCanonica))
               )
           )
       }
    }
    pred listaSolucionNoContieneNombresDeMas(res : seq⟨Estudiante, ℝ⟩, EdR : EdR)  {
        (∀i : ℤ) ((0 ≤ i < |res|) →L
        ((res[i]1 ∈ EdR.Estudiantes)∧¬(res[i]1 ∈ EdR.ListaCopiados)∧¬(estaEnElAula(res[i]1))))
    }
    aux notaDelEstudiante(solucion : Solucion, solucionCanonica : Solucion) : ℝ =
        ∑_{j=0}^{|solucion|} IfThenElse(solucion[j] = solucionCanonica[j], 1, 0) · 10/|solucion|
}
```

---

**Aclaraciones:** Consideramos los puntos de Solucion enumerados, en orden, desde el *0*. El valor de un punto todavia no resuelto es *-1*.