UBA – Facultad de Ciencias Exactas y Naturales – Departamento de Computación

Algoritmos y Estructura de Datos I

Segundo cuatrimestre de 2019

Versión: 21 de octubre de 2019

TPI - "Recuperación de Información Música"

Entrega: 6 de Noviembre (hasta las 16:30)

1. Antes de empezar

1. Bajar del campus de la materia Archivos TPI.
2. Descomprimir el ZIP.
3. Dentro del ZIP van a encontrar una carpeta llamada *MIR*, cargarla como proyecto en CLion.

2. Ejercicios

1. Implementar las funciones especificadas en la sección **Especificación**. Utilizar los tests provistos por la cátedra para verificar sus soluciones (los cuales **No pueden modificar**). Respetar los tiempos de ejecución de peor caso para las funciones que se enumeran a continuación. Justificar.
   - *revertirAudio*: $O(n)$ donde n representa la longitud del audio.
   - *limpiarAudio*: $O(n^2)$ donde n representa la longitud del audio.
   - *maximosTemporales*: $O(m \times n)$ donde m representa la cantidad de tiempos, n la longitud del audio.
2. Calcular los tiempos de ejercución en el peor caso para las siguientes funciones. Justificar.
   - $magnitudAbsolutaMaxima$
   - $audiosSoftYHard$
   - $reemplazarSubAudio$
3. Gráficar el tiempo de ejecución de revertirAudio y limpiarAudio. Utilizando el graficador.py dado por la cátedra
4. Implementar las funciones descriptas en la sección **Entrada/Salida**.
5. (*Opcional*) Utilizar la interfaz gráfica provista para probar las funciones de Entrada/Salida y escuchar el resultado de aplicarle las diferentes funciones a audios. Ver sección **Interfaz gráfica**.
6. Completar (agregando) los tests estructurales necesarios para cubrir todas las líneas del archivo *solucion.cpp*. Utilizar la herramienta **lcov** para dicha tarea. Ver sección **Análisis de cobertura**.

3. Especificación

```
proc formatoVálido (in s: seq⟨ℤ⟩, in c : ℤ, in p: ℤ, out esValido : Bool ) {
    Pre {c > 0 ∧ p > 0}
    Post {esValido = true ↔ esFormatoValido(s, p, c)}

    pred esFormatoValido (s: seq⟨ℤ⟩, c : ℤ, p: ℤ) {
        |s| > 0 ∧ |s|  mód c = 0 ∧ (∀e : ℤ)(e ∈ s → −2^(p−1) ≤ e ≤ 2^(p−1) − 1)
    }
}

proc replicar (in a: audio, in c : ℤ, in p: ℤ, out result : audio ) {
    Pre {c > 0 ∧ p > 0 ∧ esFormatoValido(a, c, p)}
    Post {|result| = c ∗ |a| ∧_L (∀i : ℤ)(0 ≤ i < |a| →_L elementoDePosicionReplicado(a, i, c, result))}

    pred elementoDePosicionReplicado (a: seq⟨ℤ⟩, i : ℤ, c : ℤ, result: seq⟨ℤ⟩) {
        (∀j : ℤ)(c ∗ i ≤ j < c ∗ (i + 1) →_L result[j] = a[i]))
```

```
    }
}

proc revertirAudio (in a: audio, in c : ℤ, in p: ℤ, out result : audio ) {
    Pre {c > 0 ∧ p > 0 ∧ esFormatoValido(a, c, p)}
    Post {|invertido| = |a| ∧ (∀i : ℤ)(0 ≤ i < |a|/c →_L bloqueRevertido(a, i, c, result)}

    pred bloqueRevertido (a: seq⟨ℤ⟩, i : ℤ, c : ℤ, result: seq⟨ℤ⟩) {
        (∀j : ℤ)(0 ≤ j < c →_L result[|a| − c ∗ (i + 1) + j] = a[(c ∗ i) + j]))
    }
}

proc magnitudAbsolutaMáxima (in a: audio, in c : ℤ, in p: ℤ, out maximos : seq⟨ℤ⟩, out posicionesMaximos : seq⟨ℤ⟩ ) {
    Pre {c > 0 ∧ p > 0 ∧_L esFormatoValido(a, c, p)}
    Post {|maximos| = c ∧ |posicionesMaximo| = c ∧_L
    (∀canal : ℤ)(1 ≤ canal ≤ c →_L esMagnitudAbsolutaMaximaDelCanal(a, canal, c))}

    pred esMagnitudAbsolutaMaximaDelCanal (a: seq⟨ℤ⟩, canal : ℤ, cantCanales: ℤ, maximos : seq⟨ℤ⟩, posicionesMaximos : seq⟨ℤ⟩) {
        (∃maxCanal : ℤ)esMaximoDelCanal(a, maxCanal, canal, cantCanales) ∧
        maximos[canal − 1] = maxCanal ∧
        a[posicionesMaximo[canal − 1]] = maxCanal ∧
        esPosicionDelCanal(posicionesMaximo[canal − 1], canal, cantCanales)
    }
    pred esMaximoDelCanal (a: seq⟨ℤ⟩, maxCanal: ℤ, canal: ℤ, cantCanales: ℤ) {
        (∀i : ℤ)(0 ≤ i < |a| ∧ esPosicionDelCanal(i, canal, cantCanales) →_L abs(a[i]) ≤ abs(maxCanal))
    }
    pred esPosicionDelCanal (posicion ℤ, canal: ℤ, cantCanales: ℤ) {
        posicion  mód cantCanales = canal − 1
    }
}

proc redirigir (in a: audio, in c : ℤ, in p: ℤ, out result : audio ) {
    Pre {(c = 1 ∨ c = 2) ∧ p > 0 ∧ esFormatoValido(a, 2, p)}
    Post {|result| = |a| ∧
    (c = 1∧(∀i : ℤ)(0 ≤ i < |a|∧esPosicionDelCanal(i, 2, 2) →_L clip(a[i], a[i−1], result[i], p))∧result[i−1] = a[i−1])∨
    (c = 2∧(∀i : ℤ)(0 ≤ i < |a|∧esPosicionDelCanal(i, 1, 2) →_L clip(a[i+1], a[i], result[i], p))∧result[i+1] = a[i+1])}

    pred clip (v1: ℤ, v2: ℤ, res: ℤ, p: ℤ) {
        (res = v1 − v2 ∧ −2^(p−1) ≤ res ≤ 2^(p−1) − 1) ∨
        (res = −2^(p−1) ∧ v1 − v2 < −2^(p−1)) ∨
        (res = 2^(p−1) − 1 ∧ 2^(p−1) ≤ v1 − v2)
    }
}

proc bajarCalidad (inout as: seq⟨audio⟩, in p: ℤ, in p2: ℤ) {
    Pre {as = as_0 ∧ p > 1 ∧ p2 > 0 ∧ p2 < p ∧ audiosValidos(as, 1, p)}
    Post {|as| = |as_0| ∧_L
    (∀i : ℤ)(0 ≤ i < |as| →_L |as[i]| = |as_0[i]| ∧_L bajaCalidadAudio(as[i], as_0[i], p, p2))}

    pred audiosValidos (as: seq⟨audio⟩, c ℤ, p ℤ) {
        (∀i : ℤ)(0 ≤ i < |as| →_L esFormatoValido(as[i], c, p))
    }
    pred bajaCalidadAudio (a: audio, a0: audio, p ℤ, p2 ℤ) {
        (∀i : ℤ)(0 ≤ i < |a| →_L a[i] = ⌊a_0[i] / 2^(p−p2)⌋)
    }
}

proc audiosSoftYHard (in sa: seq⟨audio⟩, in p: ℤ, in long: ℤ, in umbral: ℤ, out soft: seq⟨audio⟩, out hard: seq⟨audio⟩ ) {
    Pre {p > 0 ∧ long > 0 ∧ audiosValidos(as, 1, p)}
    Post {losAudiosSoftEstanEnSoft(sa, long, umbral, soft) ∧
    losAudiosHardEstanEnHard(sa, long, umbral, hard)}

    pred losAudiosHardEstanEnHard (sa: audio, long: ℤ, umbral: ℤ, hard: seq⟨audio⟩) {
```

$(\forall a : audio)(a \in sa \rightarrow_L (esHard(a, long, umbral) \leftrightarrow a \in hard))$

}

pred losAudiosSoftEstanEnSoft (sa: *audio*, long: $\mathbb{Z}$, umbral: $\mathbb{Z}$, soft: $seq\langle audio\rangle$) {

$(\forall a : audio)(a \in sa \rightarrow_L (esSoft(a, long, umbral) \leftrightarrow a \in soft))$

}

pred esSoft (a: *audio*, long: $\mathbb{Z}$, umbral; $\mathbb{Z}$) {

$\neg((\exists subAudio : audio)(esSubAudio(subAudio, a) \wedge |subAudio| > long \wedge$
$todosMayoresA(subAudio, umbral)))$

}

pred esSubAudio (subAudio: *audio*, a: *audio*) {

$(\exists i : \mathbb{Z})(\exists j : \mathbb{Z})(subsec(a, i, j) = subAudio)$

}

pred todosMayoresA (a: *audio*, umbral: $\mathbb{Z}$) {

$(\forall i : \mathbb{Z})(0 \leq i < |a| \rightarrow_L a[i] > umbral)$

}

pred esHard (a: *audio*, long: $\mathbb{Z}$, umbral: $\mathbb{Z}$) {

$\neg esSoft(a, long, umbral)$

}

}

proc reemplazarSubAudio (inout a: *audio*, in a1: *audio*, in a2: *audio*, in p: $\mathbb{Z}$) {

Pre $\{a = a_0 \wedge p > 0 \wedge esFormatoValido(a, 1, p) \wedge esFormatoValido(a2, 1, p) \wedge estaALoSumoUnaVez(a1, a)\}$

Post $\{(\exists a_i : audio)(\exists a_f : audio)(a_0 = a_i + +a2 + +a_f \wedge a = a_i + +a1 + +a_f)\}$

pred estaALoSumoUnaVez (subAudio: *audio*, a: *audio*) {

$\neg esSubAudio(subAudio, a) \vee (esSubaAudio(subAudio, a) \wedge noMasDeUnaAparicion(subAudio, a))$

}

pred noMasDeUnaAparicion (subAudio: *audio*, a: *audio*) {

$(\exists i : \mathbb{Z})(\exists j : \mathbb{Z})(0 \leq i < |a| \wedge 0 \leq j < |a| \wedge i \leq j \wedge_L subsec(a, i, j) = subAudio \wedge (\forall i_1 : \mathbb{Z})(\forall j_1 : \mathbb{Z})(0 \leq i_1 <$
$|a| \wedge 0 \leq j_1 < |a| \wedge i_1 \leq j_1 \wedge i \neq i_1 \wedge j \neq j_1 \rightarrow_L subsec(a, i_1, j_1) \neq subAudio)$

}

}

proc maximosTemporales (in a: *audio*, in p: $\mathbb{Z}$, in tiempos: $seq\langle\mathbb{Z}\rangle$, out maximos: $seq\langle\mathbb{Z}\rangle$, out intervalos: $seq\langle\mathbb{Z} \times \mathbb{Z}\rangle$) {

Pre $\{p > 0 \wedge esFormatoValido(a, 1, p) \wedge todosDistintos(tiempos) \wedge todosPositivos(tiempos)\}$

Post $\{intervalosValidos(a, tiempos, intervalos) \wedge maximosValidos(a, maximos, intervalos)\}$

pred todosDistintos (s: $seq\langle\mathbb{Z}\rangle$) {

$(\forall i : \mathbb{Z})(\forall j : \mathbb{Z})(0 \leq i < |s| \wedge 0 \leq j < |s| \wedge i \neq j \rightarrow_L s[i] \neq s[j])$

}

pred todosPositivos (s: $seq\langle\mathbb{Z}\rangle$) {

$(\forall x : \mathbb{Z})(x \in s \rightarrow_L 0 < x)$

}

pred intervalosValidos (a: *audio*, tiempos: $seq\langle\mathbb{Z}\rangle$, intervalos: $seq\langle\mathbb{Z} \times \mathbb{Z}\rangle$) {

$|intervalos| = (\sum_{i=0}^{|tiempos|-1} \left\lceil \frac{|a|}{tiempos[i]} \right\rceil) \wedge (\forall t : \mathbb{Z})(t \in tiempos \rightarrow_L (\exists is : seq\langle\mathbb{Z} \times \mathbb{Z}\rangle)(|is| = \left\lceil \frac{|a|}{t} \right\rceil) \wedge$
$incluido(is, intervalos) \wedge intervalosCorrectos(is, t, a))$

}

pred intervalosCorrectos (a: *audio*, t:$\mathbb{Z}$, is: $seq\langle\mathbb{Z} \times \mathbb{Z}\rangle$) {

$(\forall k : \mathbb{Z})(0 \leq k < \left\lceil \frac{|a|}{t} \right\rceil \rightarrow_L (\exists in : seq\langle\mathbb{Z} \times \mathbb{Z}\rangle)(in \in is \wedge (in_0 = k * t) \wedge (in_1 = ((k + 1) * t) - 1)))$

}

pred incluido (xs: $seq\langle\mathbb{Z} \times \mathbb{Z}\rangle$, ys: $seq\langle\mathbb{Z} \times \mathbb{Z}\rangle$) {

$(\forall x : \mathbb{Z})(x \in xs \rightarrow_L x \in ys)$

}

pred maximosValidos (a: *audio*, maximos: $seq\langle\mathbb{Z}\rangle$, intervalos: $seq\langle\mathbb{Z} \times \mathbb{Z}\rangle$) {

$|intervalos| = |maximos| \wedge_L (\forall i : \mathbb{Z})(0 \leq i < |maximos| \rightarrow_L esMaximo(a, maximo[i], intervalos[i]))$

}

pred esMaximo (a: *audio*, max: $\mathbb{Z}$, intervalo: $\mathbb{Z} \times \mathbb{Z}$) {

$(\forall i : \mathbb{Z})(intervalo_0 \leq i \leq intervalo_1 \wedge i < |a| \rightarrow_L a[i] \leq max)$

}

}

proc limpiarAudio (inout a: *audio*, in p: $\mathbb{Z}$, out outliers: $seq\langle\mathbb{Z}\rangle$) {

Pre $\{p > 0 \wedge esFormatoValido(a, 1, p) \wedge a = a_0\}$

`Post` $\{|a| = |a_0| \land enOutliersEstanTodosLosOutliers(a_0, outliers) \land aLimpia(a_0, a, outliers)\}$

`pred enOutliersEstanTodosLosOutliers` (a: *audio*, outliers: $seq\langle\mathbb{Z}\rangle$) {
$((\forall i : \mathbb{Z})(0 \leq i < |a| \land (esOutlier(a[i], a) \rightarrow_L i \in outliers) \land$
$((\forall outlier : \mathbb{Z})(outlier \in outliers \rightarrow_L esOutlier(a[outlier], a))$
}
`pred aLimpia` (a0: *audio*, a: *audio*, outliers: $seq\langle\mathbb{Z}\rangle$) {
$(\forall i : \mathbb{Z})(0 \leq i < |a0| \rightarrow_L (esOutlier(a0[i], a0) \land$
$(\exists valor : \mathbb{Z})(esValorEnPosicion(a0, valor, i)) \land a[i] = a0[i]) \lor (\neg esOutlier(a0[i], a0) \land a[i] = a0[i]))$
}
`pred esOutlier` (valor:$\mathbb{Z}$, a: *audio*) {
$(\exists a' : audio)(esPermutacion(a', a) \land ordenada(a') \land_L valor > a'[\lfloor 0, 95 * |a'| \rfloor])$
}
`pred esValorEnPosicion` (a: *audio*, valor:$\mathbb{Z}$, i: $\mathbb{Z}$) {
$hayNoOutlierSoloADer(a, valor, i) \lor$
$hayNoOutlierSoloAIzq(a, valor, i) \lor$
$hayNoOutlierAIzqYDer(a, valor, i)$
}
`pred hayNoOutlierSoloADer` (a: *audio*, valor:$\mathbb{Z}$, i: $\mathbb{Z}$) {
$(\exists der : \mathbb{Z})esPosNoOutlierADer(a, der, i) \land esElDerechoMasCercano(a, der, i) \land$
$\neg hayNoOutlierAIzquierda(a, i) \land valor = a[der]$
}
`pred esPosNoOutlierADer` (a: *audio*, pos:$\mathbb{Z}$, i: $\mathbb{Z}$) {
$i < pos < |a| \land \neg esOutlier(a[pos], a)$
}
`pred esElDerechoMasCercano` (a: *audio*, pos:$\mathbb{Z}$, i: $\mathbb{Z}$) {
$(\forall k : \mathbb{Z})i < k < pos \rightarrow_L esOutlier(a[k], a)$
}
`pred hayNoOutlierAIzquierda` (a: *audio*, i: $\mathbb{Z}$) {
$(\exists k : \mathbb{Z})0 < k < i \rightarrow_L \neg esOutlier(a[k], a)$
}
`pred hayNoOutlierSoloAIzq` (a: *audio*, valor:$\mathbb{Z}$, i: $\mathbb{Z}$) {
$(\exists izq : \mathbb{Z})esPosNoOutlierAIzq(a, izq, i) \land esElIzquierdoMasCercano(a, izq, i) \land$
$\neg hayNoOutlierADerecha(a, i) \land valor = a[izq]$
}
`pred esPosNoOutlierAIzq` (a: *audio*, pos:$\mathbb{Z}$, i: $\mathbb{Z}$) {
$0 \leq pos < i \land \neg esOutlier(a[pos], a)$
}
`pred esElIzquierdoMasCercano` (a: *audio*, pos:$\mathbb{Z}$, i: $\mathbb{Z}$) {
$(\forall k : \mathbb{Z})pos < k < i \rightarrow_L esOutlier(a[k], a)$
}
`pred hayNoOutlierADerecha` (a: *audio*, i: $\mathbb{Z}$) {
$(\exists k : \mathbb{Z})i < k < |a| \rightarrow_L \neg esOutlier(a[k], a)$
}
`pred hayNoOutlierAIzqYDer` (a: *audio*, valor:$\mathbb{Z}$, i: $\mathbb{Z}$) {
$(\exists izq : \mathbb{Z})esPosNoOutlierAIzq(a, izq, i) \land esElIzquierdoMasCercano(a, izq, i) \land$
$(\exists der : \mathbb{Z})esPosNoOutlierDer(a, der, i) \land esElDerechoMasCercano(a, der, i) \land valor = \lfloor \frac{a[izq]+a[der]}{2} \rfloor$
}
}

4. Entrada/Salida

Implementar las siguientes funciones.

1. `void escribirAudio(audio a, string nombreArchivo)`

   Que escribe un audio en un archivo con una única línea siguiendo el siguiente formato:

   $C\ a_0\ a_1\ \ldots\ a_{n-1}$. En donde C indica la cantidad de canales del audio y $a_0\ a_1\ \ldots\ a_{n-1}$ el contenido del audio.

   Por ejemplo $a =< 1, 5, 3, 5 >$ y la cantidad de canales es 2, el archivo debe contener:
   2 1 5 3 5

2. `tuple<int, audio> leerAudio(string nombreArchivo)`

   Que dada un nombre de archivo `nombreArchivo`, deberá devolver el audio correspondiente. El formato del archivo debe ser el mismo que en el item anterior.

5. Interfaz

Los archivos del TP incluyen el código de un programa que les va a permitir pasar un archivo wav a una secuencia de enteros (*desde_wav.py*) y otro que dada una secuencia de enteros los va a convertir en un archivo .wav (*a_wav.py*). Ambos programas fueron escritos en python.

Las máquinas de los laboratorios ya tienen casi todas las dependencias necesarias (en Linux), pero en caso de querer utilizar una máquina personal, se necesita tener instalado python3 (recomendamos a través de anaconda que cuenta con versiones para Linux, Windows y Mac. Por último, necesitamos la librería click. Para instalarlo pueden utilizar el comando (`pip3 install click --user`).

Los scripts *desde_wav.py* y *a_wav.py* deben ejecutarse desde una terminal en donde reciben los parámetros necesarios.

Ejemplo de uso *desde_wav.py* :

```
⟶  python3 desde_wav.py --wavfile ~/Downloads/4f01_milhouse.wav --output_file prueba.txt
```

- `--wavfile`: nombre del archivo wav a convertir en archivo de texto. El wav debe estar en formato wav (PCM 44.1 khz). Pueden encontrar audios con ese formato en la carpeta *audios*.
- `--output_file`: nombre del archivo en donde se exportarán los datos del audio.

Para ver como queda el archivo:

```
⟶  head -c N prueba.txt
```

- `N`: cantidad de datos que se quieren ver.

Ejemplo de uso *a_wav.py* :

```
⟶  python3 a_wav.py --input_file prueba.txt --output_file prueba.wav --profundidad 16
```

- `--input_file`: nombre del archivo que contiene los datos del audio. El formato es el explicado en Entrada/Salida - escribirAudio
- `--output_file`: nombre del archivo wav que se genera.

Para escuchar el audio ejecutar:

```
⟶  play prueba.wav
```

También pueden hacer doble click en el audio.

6. Análisis de cobertura

Para realizar el análisis de cobertura de código utilizaremos la herramienta Gcov, que es parte del compilador GCC. El target *recuperacionInformacionMusical* ya está configurado para generar información de cobertura de código en tiempo de compilación. Esta información estará en archivos con el mismo nombre que los códigos fuente del TP, pero con extensión `*.gcno`. Al ejecutar los casos de test, se generarán en el mismo lugar que los `*.gcno` otra serie de archivos con extensión `*.gcda`. Una vez que tenemos ambos conjuntos de archivos, ejecutar el siguiente comando:

```
⟶  lcov --capture --directory recuperacionInformacionMusical --output-file coverage.info
```

Se generará el archivo `coverage.info` que luego podremos convertir a HTML para su visualización con el siguiente comando:

```
⟶  genhtml coverage.info --output-directory cobertura
```

Finalmente, se generará un archivo `index.html` dentro de `salida/cobertura` con el reporte correspondiente. Utilizar cualquier navegador para verlo.

Para mayor información, visitar:
`https://medium.com/@naveen.maltesh/generating-code-coverage-report-using-gnu-gcov-lcov-ee54a4de3f11`.

Términos y condiciones

El trabajo práctico se realiza de manera grupal con grupos de exactamente 2 personas. Para aprobar el trabajo se necesita:

- Que todos los ejercicios estén resueltos.
- Que las soluciones sean correctas.
- Que todos los tests provistos por la cátedra funcionen.
- Que las soluciones sean prolijas: evitar repetir implementaciones innecesariamente y usar adecuadamente funciones auxiliares.
- Que los test cubran todas las lineas de las funciones.

Pautas de Entrega

Fecha de entrega: Miercoles 6 de Noviembre (hasta las 16:30hs)
Devolución: Miercoles 13 de Noviembre
Recuperatorio: Viernes 6 de Diciembre