



UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

Estructura de Datos y Análisis de Algoritmos

Lab N°3 Similitud de Genes

Roberto Orellana T.

Docente: Javiera Torres M.
Ayudante: Nicolás Romero F.

Santiago - Chile
07-7-2019

TABLA DE CONTENIDOS

CAPÍTULO 1. INTRODUCCIÓN	4
CAPÍTULO 2. DESCRIPCIÓN DE LA SOLUCIÓN	5
2.1 Marco teórico	5
2.1 Herramientas y técnicas	10
2.2 Algoritmos y estructuras de datos.....	11
CAPÍTULO 3. ANÁLISIS DE LOS RESULTADOS.....	13
3.1 Resultados, Falencias y Propuestas de Mejora	13
3.2 Tiempo de ejecución y orden de complejidad	14
3.2.1 Distancia	14
3.2.2 AncestroComun	15
CAPÍTULO 4. CONCLUSIÓN.....	16
CAPÍTULO 5. REFERENCIAS.....	17
CAPÍTULO 6. MANUAL DE USUARIO	18
6.1 Introducción	18
6.2 Cómo compilar y ejecutar.....	18
6.3 Funcionalidades del programa	20
6.4 Posibles Errores	22

INDICE DE FIGURAS

Figura 2.1: Grafo acíclico dirigido	6
Figura 2.2: Distancia al ancestro común y distancia a la raíz.....	7
Figura 2.3: Ecuación para la similitud Wu-Palmer	8
Figura 2.4: Ejemplo de la ecuación Wu-Palmer.....	8
Figura 2.5: Ecuación para la similitud Leacock-Chorodow.....	8
Figura 2.6: Ejemplo de la ecuación Leacock-Chorodow	8
Figura 2.7: Similitud entre dos genes	9
Figura 2.8: Similitud entre dos genes por el método de Wu-Palmer.....	9
Figura 2.9: Similitud entre dos genes por el método de Leacock-Chodorow	9
Figura 3.1: Resultado de la ejecución.....	13
Figura 3.2: Resultado de hasta 9 tipos de gen.	13
Figura 6.1: Tipos de archivos	18
Figura 6.2: Instrucción para generar el ejecutable.....	19
Figura 6.3: Instrucción para ejecutar el programa.....	19
Figura 6.4: Resultado por Consola	20
Figura 6.5: Estructura del archivo de entrada (procesos.in)	20
Figura 6.6: Estructura del archivo de entrada (genes.in)	21
Figura 6.7: Ejemplo de archivo erróneo 1	22
Figura 6.8: Ejemplo de nombre erróneo	22

CAPÍTULO 1. INTRODUCCIÓN

Los Alumnos de Algebre tras una desesperada solución a sus problemas se contactaron en secreto con un Informático del Departamento de Ingeniería en Informática de la Universidad de Santiago de Chile para solucionar sus problemas.

Un informante de la Coordinación vio el momento justo cuando el Informático entrego un pendrive a los alumnos de algebra, audazmente el informante logro obtener un cabello del informático del cual se podrá identificar los genes y sus respectivos procesos biológicos para identificar a los individuos involucrados.

Para eso se ha solicitado en el siguiente laboratorio elaborar un programa que pueda determinar qué genes tienen comportamientos similares, los procesos biológicos y los genes asociados a cada proceso.

La estructura con la que se deberá trabajar será la de un grafo a cíclico dirigido, en la cual existe un nodo raíz y cada descendiente puede tener uno o más ancestros. Cada vértice del grafo representa un proceso biológico, y cada proceso biológico podrá tener 0 o más genes asociados.

Para llegar a la solución se utilizarán distintos métodos de similitud como “*Wu-Palmer*” y “*Leacock-Chodorow*”. Para el algoritmo se analizará su tiempo de ejecución y su orden de complejidad, al igual que sus resultados, falencias y propuestas de mejoras.

CAPÍTULO 2. DESCRIPCIÓN DE LA SOLUCIÓN

2.1 MARCO TEÓRICO

Para el diseño de la solución, es necesario entender las siguientes definiciones:

- a. Wu-Palmer: Calcula la similitud entre dos conceptos $c1$ y $c2$ teniendo en cuenta la profundidad del concepto $c3$ más cercano que conecta a ambos nodos.
- b. Leacock-Chodorow: La medida de Leacock & Chodorow utilizaba la distancia mínima entre dos conceptos, $\text{length}(c1, c2)$, y la profundidad máxima de la jerarquía.
- c. Arreglo Dinámico: Es un array de elementos que crece o mengua dinámicamente conforme los elementos que se agregan o se eliminan, Este tipo de Array permite reservar memoria en tiempo de ejecución y para esto es necesario la cantidad de memoria a reservar como parámetro de entrada.

Existen 3 variantes de arreglo dinámico

- Malloc: La función reserva un bloque de memoria y devuelve un puntero void al inicio de la misma.
 - Calloc: La función funciona de modo similar a malloc, pero además de reservar memoria, inicializa a 0 la memoria reservada.
 - Realloc: La función redimensiona el espacio asignado de forma dinámica anteriormente a un puntero.
- d. Recursividad: La recursividad (recursión) es la propiedad por la cual una función se llama a sí misma directa o indirectamente. La recursión indirecta implica utilizar más de una función. Se puede considerar la recursividad como una alternativa a la iteración. La recursión permite especificar soluciones naturales, sencillas, que serían, en caso contrario, difíciles de resolver. Toda función recursiva debe contemplar un caso base o condición de salida, para terminar, o la recursividad no podrá terminar nunca.

- e. Paso por Valor: significa que la función (o subrutina) recibe sólo una copia del valor que tiene la variable, o sea que no la puede modificar.
- f. Paso por Referencia: significa que se pasa la posición de memoria donde esta guardada la variable, por lo que la función puede saber cuánto vale, pero además puede modificarla de cualquier manera.
- g. Estructura: Una estructura es un tipo de dato compuesto que permite almacenar un conjunto de datos de diferente tipo. Los datos que contiene una estructura pueden ser de tipo simple (caracteres, números enteros o de coma flotante etc.) o a su vez de tipo compuesto (vectores, estructuras, listas, etc.).

Con las instrucciones antes definidas, son utilizadas para la solución que creara los nodos, almacenara los procesos y genes en matrices, luego utilizar los algoritmos adecuados para calcular la similitud entre genes y procesos como muestra en la siguiente figura.

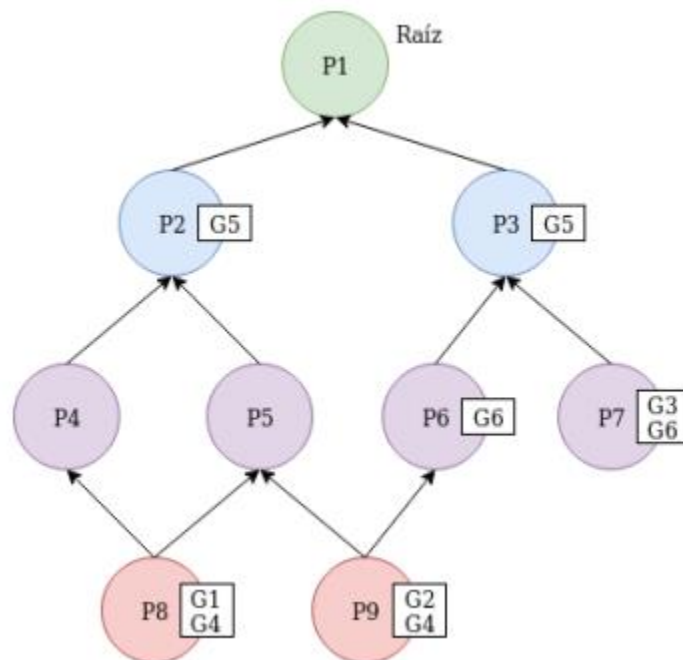


Figura 2.1: Grafo acíclico dirigido

Para ello de manera general se identifican los siguientes algoritmos.

- Obtener y validar la información desde el archivo de datos.
- Crean Matriz
- Encontrar el Ancestro en común
- Calcular la Distancia
- Similitud wu-palmer
- Similitud Leacock-Chorodow
- Similitud de Genes

Similitud de Wu-Palmer

Para la solución se estudian el método Wu-Palmer que calcula la similitud entre dos procesos N1 y N2 teniendo en cuenta la profundidad del proceso N3 más cercano que conecta a ambos nodos.

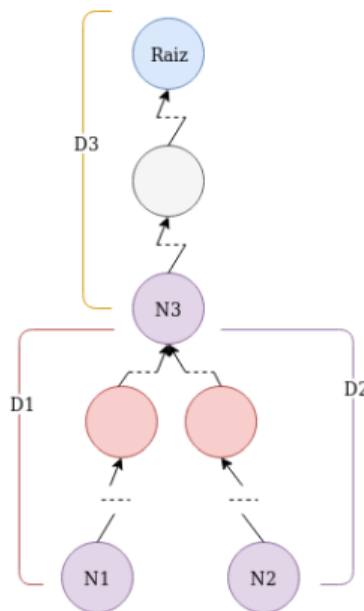


Figura 2.2: Distancia al ancestro común y distancia a la raíz.

Para realizar el cálculo de la similitud entre (N1, N2) de Wu-Palmer se debe conocer la distancia Donde:

- $D1$ = distancia de N1 a N3.
- $D2$ = distancia de N2 a N3.
- $D3$ = distancia de N3 a la raíz.

$$Sim_{wp}(N_A, N_B) = \frac{2 * D3}{D1 + D2 + (2 * D3)}$$

Figura 2.3: Ecuación para la similitud Wu-Palmer

Por ejemplo, para el proceso 9 y el proceso 6 de la figura 2.1 la similitud sería:

$$Sim_{wp}(P9, P6) = \frac{2 * 2}{1 + 0 + 2 * 2} = 0.800000$$

Figura 2.4: Ejemplo de la ecuación Wu-Palmer

Similitud de Leacock-Chodorow

También se utilizó el método Leacock-Chodorow que calcula la distancia mínima entre dos procesos N1 y N2 y la profundidad máxima del grafo.

Para realizar el cálculo de la similitud entre (N1, N2) de Leacock-Chodorow se debe conocer la distancia Donde:

- D1 = distancia de N1 a N3.
- D2 = distancia de N2 a N3.
- D = Es la profundidad máxima del grafo.

$$Sim_{lc}(N_A, N_B) = -\log\left(\frac{D1 + D2 + 1}{2 * D}\right)$$

Figura 2.5: Ecuación para la similitud Leacock-Chodorow

Por ejemplo, para el proceso 9 y el proceso 7, la similitud sería:

$$Sim_{lc}(P9, P7) = -\log\left(\frac{2 + 1 + 1}{2 * 4}\right) = 0.301030$$

Figura 2.6: Ejemplo de la ecuación Leacock-Chodorow

Similitud entre dos genes

La similitud entre dos genes x , y corresponde al promedio de las similitudes entre los pares de procesos biológicos en los que se encuentran involucrados los genes.

$$Simg(g_x, g_y) = \frac{\sum_{i \in x} \sum_{j \in y} Sim(i, j)}{|P_x| |P_y|}$$

Figura 2.7: Similitud entre dos genes

Dónde $Sim(i, j)$ corresponde a la similitud de Wu-Palmer o de Leacock-Chodorow según corresponda entre los procesos involucrados de los genes y $|P_x||P_y|$ corresponde al total de pares de procesos involucrados.

Por ejemplo la similitud de Wu-Palmer entre los genes 4 y 6 de la figura 2.1 sería la siguiente:

$$Simg_{wp}(G4, G6) = \frac{Sim_{wp}(P8, P6) + Sim_{wp}(P8, P7) + Sim_{wp}(P9, P6) + Sim_{wp}(P9, P7)}{4}$$

$$Simg_{wp}(G4, G6) = \frac{0.000000 + 0.000000 + 0.800000 + 0.400000}{4}$$

$$Simg_{wp}(G4, G6) = 0.300000$$

Figura 2.8: Similitud entre dos genes por el método de Wu-Palmer

Y la similitud de Leacock-Chodorow de los genes 4 y 6 sería: Entradas

$$Simg_{lc}(G4, G6) = \frac{Sim_{lc}(P8, P6) + Sim_{lc}(P8, P7) + Sim_{lc}(P9, P6) + Sim_{lc}(P9, P7)}{4}$$

$$Simg_{lc}(G4, G6) = \frac{0.124939 + 0.124939 + 0.602060 + 0.301030}{4}$$

$$Simg_{lc}(G4, G6) = 0.288242$$

Figura 2.9: Similitud entre dos genes por el método de Leacock-Chodorow

2.1 HERRAMIENTAS Y TÉCNICAS

Para la creación de este software se optó por el lenguaje de programación C, debido a su acceso de bajo nivel mediante el uso de punteros, el paso por valor, el paso por referencia, punteros a funciones, variables estáticas y estructura de datos, que permiten una forma rudimentaria de encapsulado y polimorfismo, además de ser un requerimiento no funcional del Laboratorio

En la compilación del código se utilizó MinGW, es una implementación de los compiladores GCC para plataformas Win32, permitiendo un desarrollo de aplicaciones nativas para esa plataforma, pudiendo generar ejecutables y biblioteca usando la API de Windows.

La representación utilizada para este proyecto fue la de una estructura con matrices dinámica dentro de la estructura, debido a que es una estructura compuesta de elementos se hace más fácil el acceso a los datos y el arreglo dinámico para el almacenamiento de las matrices debido a que se desconoce el tamaño de estas.

Para la representación de la estructura definida, se compondrá de los siguientes elementos.

- Genes: es de tipo “*entero*” que corresponde al número de genes que existe en el grafo, los genes pueden repetirse.
- Vértices: es de tipo “*entero*” que representa al número de procesos que existe dentro del grafo.
- Matrizgen: es de tipo “*array[entero][entero]*” que almacena la verificación de los genes correspondiente a cada proceso dentro de la matriz (0 o 1)..
- Matriz: es de tipo “*array[entero][entero]*” que almacena la verificación de cada proceso dentro de la matriz (0 o 1).

Creación de funciones que trabajan sobre la Matriz tales como ancestroComun, distancia y simg además de otras funciones.

Las funciones que permiten conocer la Similitud entre genes son:

- Obtener los datos desde un archivo y agregarlos a la matriz
- Crear matriz para almacenar la información de procesos y gen

- Leer el grafo desde un archivo externo que contiene los procesos y genes.
- Ancestro común entre dos procesos
- Distancia entre dos Procesos
- Similitud Wu-palmer
- Similitud Leacock-Chorodow
- Similitud entre genes

2.2 ALGORITMOS Y ESTRUCTURAS DE DATOS

Para la solución basada en estructura de datos y matrices dinámicas los algoritmos identificados en el marco teórico se desglosan como sigue.

a. Leer grafo

Obtiene la matriz de procesos, la matriz de genes, la cantidad de procesos y la cantidad de genes desde un archivo externo, además valida si el archivo existe y si es una matriz con datos válidos para el programa.

- Datos de entrada (string, string).
- Datos de salida (Struct).

b. Ancestro Comun

Función que encuentra el ancestro común mas próximo entre ambos procesos.

- Datos de entrada (Struct, entero, entero).
- Datos de salida (entero).

c. Distancia

Permite saber la distancia que existe entre dos procesos.

- Datos de entrada (Struct, entero, entero, entero)
- Datos de salida (entero).

d. Similitud Wu-Palmer

Esta función permite calcular la similitud entre dos procesos mediante el método Wu-Palmer

- Datos de entrada (Struct, entero, entero).
- Datos de salida (double).

e. Similitud Leacock-Chorodow

Esta función permite calcular la similitud entre dos procesos mediante el método Leacock-Chorodow.

- Datos de entrada (Struct, entero, entero).
- Datos de salida (double).

f. Similitud genes

Esta función calcula la similitud entre todos los procesos donde existan los genes ingresados por el usuario.

- Datos de entrada (Struct, entero, entero).
- Datos de salida (vacío).

g. Crear Matriz

Crea las matrices para almacenar los procesos y genes, en su estado existente o no con 0 y 1.

- Datos de entrada (entero, entero).
- Datos de salida (entero).

h. Liberar Memoria

Función que permite liberar la memoria dinámica utilizada en las matrices.

- Datos de entrada (Lista, entero).
- Datos de salida (vacío).

CAPÍTULO 3. ANÁLISIS DE LOS RESULTADOS

Para obtener una visión de los resultados obtenidos, se aplica el cálculo del orden de complejidad de cada algoritmo relevante en la solución. Estos corresponden a aquellos que se utilizan al obtener y validar datos, encontrar el ancestro común, la distancia entre procesos, calcular la similitud mediante ambos métodos y la similitud entre genes.

3.1 RESULTADOS, FALENCIAS Y PROPUESTAS DE MEJORA

El resultado obtenido es óptimo con respecto a los objetivos establecidos por el trabajo, para la propuesta de mejora, se puede aumentar la cantidad de nodos para aumentar los procesos, ya que con el tiempo acotado que se tuvo para este laboratorio solo se logró que se analizaran hasta 9 procesos a la vez, ya que con 10 proceso hacia arriba la aplicación caerá, además los datos de entrada no todos fueron validados 100%, podría tener problemas si se agregan datos aleatorios, por eso existe un manual de usuario para que puedan ingresar los datos válidos.

```
ingrese primer gen: 4
ingrese segundo gen: 6
Similitud de genes Wu-Palmer: 0.300000
Similitud de genes Leacock-Chorodow: 0.288242
Si desea ingresar otro par de genes presione 1, en caso contrario presione cualquier tecla
```

Figura 3.1: Resultado de la ejecución.

Hasta el momento se han hecho pruebas con un grafo de hasta 9 procesos y con genes también de hasta 9 tipos de gen llegando al resultado sin problemas.

```
ingrese primer gen: 9
ingrese segundo gen: 2
Similitud de genes Wu-Palmer: 0.600000
Similitud de genes Leacock-Chorodow: 0.451545
Si desea ingresar otro par de genes presione 1, en caso contrario presione cualquier tecla
```

Figura 3.2: Resultado de hasta 9 tipos de gen.

3.2 TIEMPO DE EJECUCIÓN Y ORDEN DE COMPLEJIDAD

Para sacar el tiempo de ejecución y posteriormente el orden de complejidad se utiliza la función que lleva todo el peso en cuanto a instrucciones de iteración

3.2.1 Distancia

Pseudocodigo

```

Proceso distancia
  Si NA == NB Entonces
    Retorna dist
  SiNo
    Para i<-0 Hasta g.vertice Con Paso 1 Hacer
      secuencia_de_acciones
      Si g.matriz != 0 Y dist < g.vertice Y aux == 0 Entonces
        aux<-distancia(g, i, NB, dist +1)
      Fin Si
    Fin Para
  Fin Si
  retorna aux
FinProceso

```

$$T_{max}(n) = \sum_0^n n * c * n^k$$

Quitando las constantes y los valores significativos quedaría de orden (n^k) que tendría complejidad polinómica

3.2.2 AncestroComun

Pseudocodigo

```

Proceso AncestroComun
  Para i<-0 Hasta g.vertice Con Paso i Hacer
    Si g.matriz != 0 Entonces
      Para j<-0 Hasta g.vertice Con Paso j Hacer
        Si g.matriz != 0 Entonces
          Si i == j Entonces
            Si comun z i + 1 Entonces
              comun<-i+1
            Fin Si
          Fin Si
          Si PB != j Entonces
            aux = AncestroComun(g, PB, j)
            Si comun < aux Entonces
              comun<-aux
            Fin Si
          Fin Si
        Fin Si
      Fin Para
    Si PA != i Entonces
      aux = AncestroComun(g, i, PB)
      Si comun < aux Entonces
        comun<-aux
      Fin Si
    Fin Si
  Fin Para
  devuelve comun
FinProceso

```

$$T_{max}(n) = \left(\sum_0^n n * \left(\sum_0^n n * (n^k * C) \right) \right)$$

$$T_{max}(n) = \left(\sum_0^n n * (n^k) \right)$$

Quitando las constantes y los valores significativos quedaría de orden (n^k) que tendría complejidad polinómica

El programa debido a sus funciones recursivas es de $O(n^k)$ complejidad polinómica ($k > 3$), si k crece, la complejidad del programa es bastante mala.

CAPÍTULO 4. CONCLUSIÓN

En conclusión, podemos decir que este laboratorio fue el más complicado en cuanto al análisis de la solución ya que encontrar el ancestro en común y la distancia hacia estos, además con la representación escogida fue bastante complejo.

Respecto al laboratorio, la inexperiencia en herramientas de Debbug en este entorno dificultó la tarea de encontrar errores. por lo que es posible que existan algunos que aún no han sido detectados, a pesar de todo tiene un buen tiempo de respuesta en cuanto a la ejecución.

En cuanto a los errores detectados y no mitigados, se encuentra que no se puede ejecutar archivos con procesos mayores a 9 vértices ya que por el tiempo acotado no se pudo realizar las lecturas pertinentes y la verificación del archivo para que pudiera obtener más de 9 procesos, se espera en la evolución del programa completar los errores detectados.

Con este Laboratorio se consigue una gran experiencia como programador al entender que programar con cualquier tipo de instrucciones o métodos no es lo mismo que saber y entender cuáles son las más eficientes y menos complejas

CAPÍTULO 5. **REFERENCIAS**

Köhler, J. (s.f.). *Análisis De Algoritmos Y Estructura De Datos*. Informe académico, Universidad de Santiago de Chile, Departamento de Ingeniería en Informática, Santiago.

CAPÍTULO 6. MANUAL DE USUARIO

6.1 INTRODUCCIÓN

Este programa fue Elaborado principalmente para detectar la similitud entre distintos procesos utilizando los métodos de similitud de Wu-Palmer y Leacock-Chorodow.

Para verificar la similitud entre procesos se debe agregar dos tipos de genes los cuales ya se encuentran en algunos de los procesos.

Científicamente está comprobado que mientras menor sea la distancia entre dos genes, mayor será la relación entre los individuos, por lo que de esa manera pueden encontrar a todas las personas.

El siguiente manual explica cómo utilizar la herramienta para sacar la similitud entre genes, la forma en que se compila en diferentes Ambiente como Windows y Linux, además de los posibles errores.

6.2 CÓMO COMPILAR Y EJECUTAR

El programa contiene cuatro archivos (*Main.c*, *Add.h*, *procesos.in* y *genes.in*), dos archivos que pertenecen al código (.c y .h) y dos archivo (.in) que contienen los procesos, la cantidad de procesos, los genes y la cantidad de genes.

Los cuatro archivos deben estar en el mismo directorio.





Nombre	Tipo
 main.c	C Source File
 add.h	C Header File
 procesos.in	Archivo IN
 genes.in	Archivo IN

Figura 6.1: Tipos de archivos

El código está distribuido en dos archivos:

- *Main.c* que contiene la Función principal del programa (main)
- *Add.h* que contienen todas las funciones que permiten conseguir la Potencia de una Matriz.

Para la compilación del código se necesita la Instalación de MinGW en ambientes Windows y Linux.

Se inicia la consola y se busca el directorio donde se encuentran los códigos .c y .h.

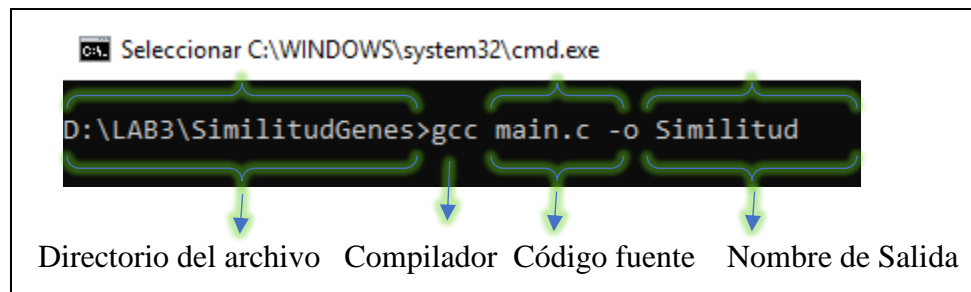


Figura 6.2: Instrucción para generar el ejecutable

Se ejecuta la instrucción donde “**gcc**” es el compilador que se está utilizando, seguido del nombre del código fuente y después el “**-o**” que es la instrucción para definir el nombre de salida del ejecutable. En ambientes de trabajo como Windows y Linux se ejecuta la Instrucción antes mencionada de la misma manera con la única condición que en Linux le debe agregar el “**-lm**” para que ejecute la librería *<math.h>*.

Con el programa ya compilado, ahora se procede a ejecutar.

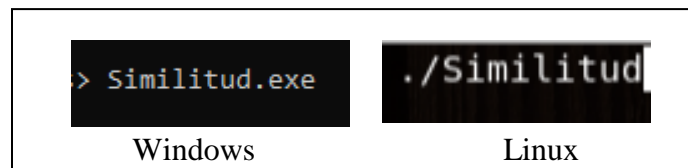


Figura 6.3: Instrucción para ejecutar el programa

Para ejecutar el programa en consolas Windows se debe agregar la extensión “.exe” (PotenciaMatriz.exe), y para consolas Linux se ejecuta agregando un “./” (./PotenciaMatriz)

```

root@Robert:~/Documentos/SimilitudGenes# ./Similitud
ingrese primer gen: 4 an -
ingrese segundo gen: 6 ip4
Similitud de genes Wu-Palmer: 0.300000
Similitud de genes Leacock-Chorodow: 0.288242
Si desea ingresar otro par de genes presione 1, en caso contrario presione cualquier tecla

```

Figura 6.4: Resultado por Consola

Finalmente nos entrega el resultado de la similitud de genes por el método de WP y LC.

6.3 FUNCIONALIDADES DEL PROGRAMA

El programa puede calcular la similitud de cualquier gen, solo debe cumplir un tipo de formato en el archivo de entrada.

Para comenzar el archivo debe tener como nombre y extensión “procesos.in”

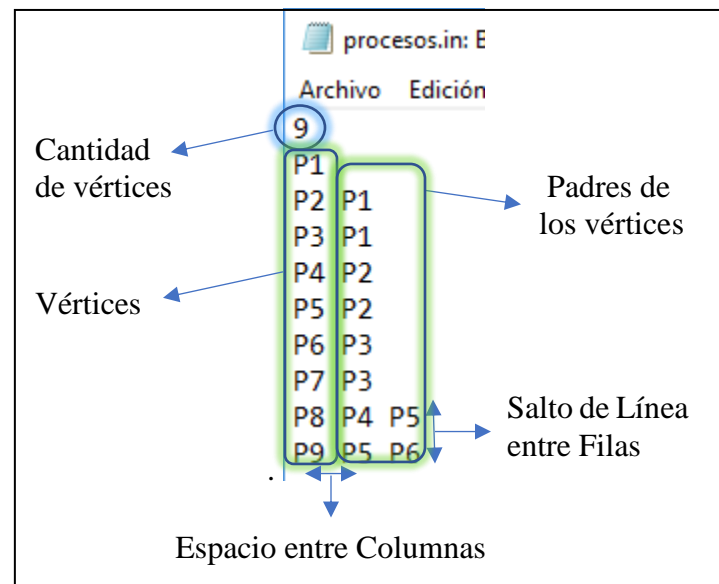


Figura 6.5: Estructura del archivo de entrada (procesos.in)

El archivo “procesos.in” contiene la siguiente estructura para evitar errores de lectura:

- La cantidad de procesos es el primer elemento del archivo.
- Los datos dentro del archivo el primer elemento de cada fila corresponde al proceso hijo y los siguientes elementos corresponden a los procesos padres que es representado por “Pn” donde n corresponde a números del 1 al 9
- El largo de la columna debe ser igual al largo introducido al principio del archivo.

- Cada columna debe ir separado por un espacio.
- Cada fila debe ir separada por un salto de línea [tecla Enter].
- La última fila no lleva salto de línea (algún espacio o salto de línea de más, puede provocar error de lectura en el archivo).

Para el archivo de genes debe tener como nombre y extensión “genes.in”

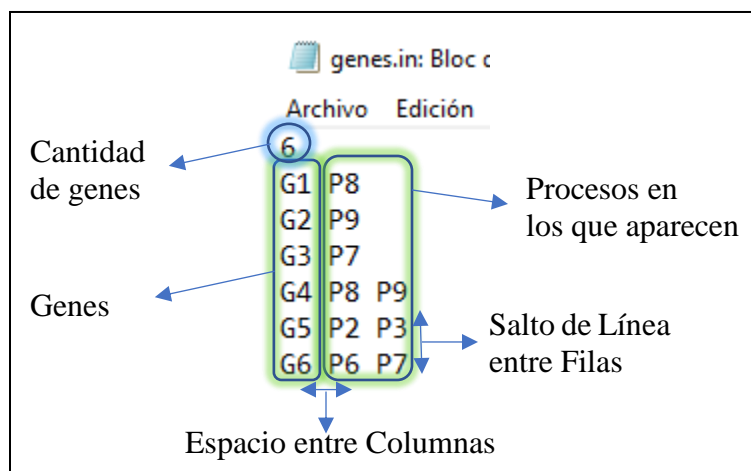


Figura 6.6: Estructura del archivo de entrada (genes.in)

El archivo “genes.in” contiene la siguiente estructura para evitar errores de lectura:

- La cantidad de genes es el primer elemento del archivo.
- Los datos dentro del archivo el primer elemento de cada fila debe ser “Gn” donde n corresponde a números del 1 al 9, los siguientes elementos de cada fila corresponden a los procesos donde se encuentran cada gen que es representado por “Pn” donde n corresponde a números del 1 al 9
- El largo de la columna debe ser igual al largo introducido al principio del archivo.
- Cada columna debe ir separado por un espacio.
- Cada fila debe ir separada por un salto de línea [tecla Enter].
- La última fila no lleva salto de línea (algún espacio o salto de línea de más, puede provocar error de lectura en el archivo).

Con solo tener la estructura mencionada anterior mente, es suficiente para conseguir que el programa funcione perfectamente

6.4 POSIBLES ERRORES

El programa no es lo bastante robusto, lo cual provocaría errores de ejecución en caso de ingresar alguna matriz con caracteres no permitidos.

a. Caracteres no Permitidos

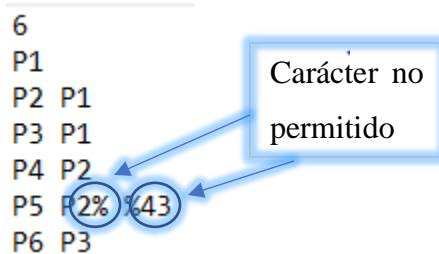


Figura 6.7: Ejemplo de archivo erróneo 1

Si los archivos tienen datos no permitidos como letras o cualquier tipo de carácter, a la consola arrojaría el siguiente mensaje de error “*los Datos del fichero están corruptos*”.

b. Archivo no Encontrado

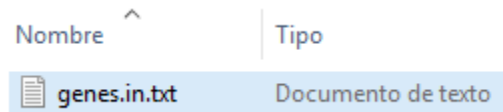


Figura 6.8: Ejemplo de nombre erróneo

Para que el programa pueda leer el archivo, debe tener como nombre “*genes*” o “*procesos*” y extensión “.in”, por lo general siempre se comete el error de dejar la extensión “.txt” (“*genes.in.txt*”). La consola arrojaría el siguiente mensaje de error “*Error en la apertura del archivo*”.

c. Sin Memoria

Si el programa tiene problemas al asignar memoria a las matrices de los procesos o genes, la consola arrojaría el siguiente mensaje de error “*Error al asignar Memoria*”.

Ante cualquier problemas o consulta puede llamar a nuestra mesa de ayuda al fono 555-55555555 o un Mail a [sinsoporte@determinante.cl](mailto:sinsoporte@ determinante.cl)