Práctica 5 – Reordenación del Genoma

Definición del problema y objetivo

El objetivo de esta práctica es estudiar e implementar dos algoritmos de ordenación de permutaciones. El primero es un algoritmo de fuerza bruta denominado ***prefix sort***. El segundo es un algoritmo voraz denominado ***break point reversal***. En el campo de la bioinformática, estos algoritmos se pueden utilizar para estudiar la distancia evolutiva entre especies. En el documento rearrangement.pdf (disponible en el campus virtual) puedes encontrar la definición del problema y los algoritmos.

Realización de la práctica

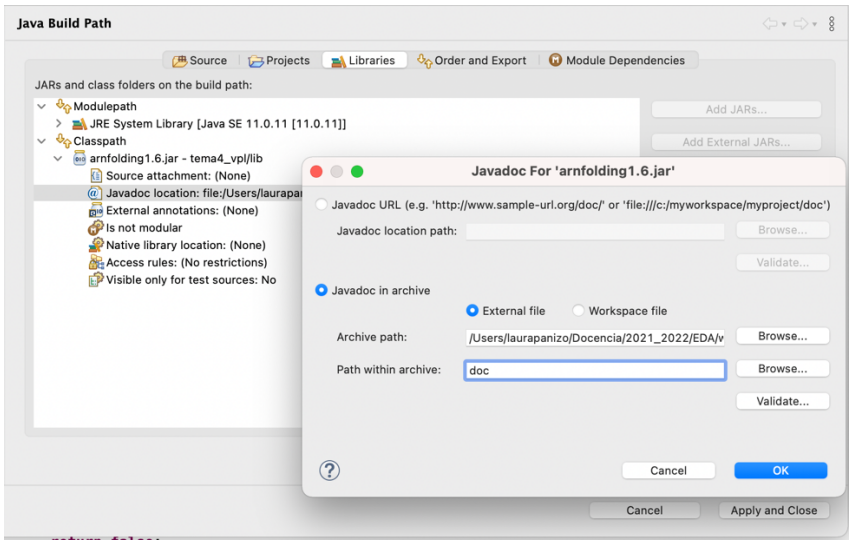
El estudiante tiene que implementar:

- El algoritmo ***prefix sort*** que se describe en las transparencias en el método _run de la clase PrefixSort
- Los métodos getCandidates y getOperationQuality del algoritmo ***break point reversal***. Estos métodos hay que implementarlos en la clase BreakPointReversalAlgorithm. El método run de dicha clase ya tiene implementado el esqueleto del algoritmo y no hay que modificarlo.

Además de las clases PrefixSort y BreakPointReversalAlgorithm, se proporcionan las siguientes clases y librerías:

- rearrangement1.6.jar y rearragement1.6-doc.jar: esta librería contiene algunas clases necesarias para la implementación de los métodos.
    - Permutation: Esta clase representa una permutación. Ofrece métodos para obtener el tamaño de la permutación (size()), el elemento que ocupa una posición concreta (get(int posicion)), o la posición que ocupa un elemento (indexOf(int elemento)).
    - Reversal: Esta clase nos permite realizar la operación de inversión de un fragmento de la permutación. El constructor Reversal(int start, int end) recibe como parámetro los índices de inicio y fin del fragmento que hay que invertir. Para realizar la inversión sobre una permutación concreta se utiliza el método apply(Permutation p) que invertirá el orden de los elementos de p entre las posiciones start y end.

  Para importar la librería en el proyecto eclipse:
    - Copiar el fichero jar en el proyecto. Por ejemplo, puedes crear una carpeta lib dentro del proyecto para mantener los ficheros organizados.
    - Sobre el nombre del proyecto, pulsa el botón secundario y selecciona Configure Build Path > Pestaña Libraries
    - Si en el área central te aparecen Module Path y Class Path selecciona Class Path
    - Pulsa el botón Add JARs y selecciona el fichero rearrangement1.6.jar
    - Para añadir la documentación asociada despliega la librería asociada al jar, y haz doble click sobre Javadoc location. Selecciona Javadoc in archive > External file. En el path añade el fichero rearragement1.6-doc.jar, y en Path within archive selecciona doc (como aparece en la imagen).



- TestRearrangementAlgorithm: en esta clase se encuentra en método main (no hay que modificarlo). El método main recibe en los argumentos de entrada el modo de ejecución, que puede ser uno de los siguientes:
    - –g <algoritmo> <tamaño>: genera aleatoriamente una permutación con el tamaño especificado y utiliza el algoritmo que se indica para realizar la ordenación. Los posibles valores del argumento <algoritmo> son {prefixsort, breakpointreversal}. Además, se puede añadir un cuarto argumento opcional <seed> que se utiliza como semilla para la generación aleatoria de la permutación.
    - –f <algoritmo> <input-file>: utiliza el <algoritmo> para ordenar todas las permutaciones que se encuetran en el fichero <input_file>. Cada línea del fichero contiene la longitud de la permutación y la permutación, excepto la última línea que solo contiene -1.
    - – r <algorithm> <intial_length> <doubling> <num_test>: en este modo se generan permutaciones aleatorias con diferentes longitudes. Se generarán num_test permutaciones de cada longitud. La longitud inicial de las permutaciones viene dada por init_length, y se irá aumentando la longitud tantas veces como indique el parámetro doubling. Todas las permutaciones se ordenarán con el algoritmo que se pasa como primer parámetro. Opcionalmente se puede añadir como último parámetro una semilla para la generación aleatoria de las permutaciones. El programa genera un fichero con los tiempos de ejecución que se puede utilizar para realizar un estudio experimental de la complejidad del algoritmo.
- Fichero de prueba (permutations.txt) para usar con el modo -f y la correspondiente salida (permutations.out).
- analyze.R: este fichero contiene un script que procesa los ficheros que se generan con el modo –r. Hay una guía en el campus que explica como instalar el entorno R y ejecutar el script.

Entrega y evaluación

La evaluación de la práctica consiste en la entrega y evaluación automática del código del alumno y una memoria (de no más de 3 páginas) en la que el alumno presenta los objetivos, el entorno de ejecución donde se ha realizado las pruebas, los resultados teóricos esperados y los resultados que ha obtenido en el estudio experimental de la complejidad del algoritmo. A continuación, se detalla la nota que se puede obtener:

Nota	Descripción
0	No pasa el test en laboratorio ni entrega memoria o el sistema detecta plagio en el código o la memoria
[1,4]	No pasa el test en laboratorio. Entrega la memoria y pasa los test fuera de plazo
[4,6]	Pasa parcialmente los test de laboratorio. Entrega memoria y pasa test fuera de plazo
[6,10]	Pasa los test en laboratorio y entrega memoria

Para la evaluación automática del código el estudiante debe subir los ficheros PrefixSort.java y BreakPointReversalAlgorithm.java a una actividad del campus que realizará una serie de pruebas sobre la implementación. El código hay que subirlo siempre al final de la sesión de laboratorio correspondiente, y si no se pasan todas las pruebas, el alumno tendrá una segunda oportunidad fuera de plazo. La fecha se comunicará al final de la sesión de laboratorio.