Universidad de los Andes
Facultad de Ingeniería

Facultad de Ciencias

Maestría en Biología Computacional
BCOM4006 – Algoritmos en Biología Computacional
Semestre: 202020

Tarea 2 – Ensamblaje de genomas

Objetivos

1. Realizar una implementación sencilla del algoritmo OLC para ensamblaje de genomas
2. Diferenciar conteos de lecturas de conteos de k-mers
3. Practicar una implementación de grafos dirigidos

Instrucciones

Realizar los pasos de desarrollo de software y análisis de datos descritos a continuación. Redactar un informe que muestre los resultados de los diferentes pasos. Se adjunta un proyecto en java que incluye código para procesar archivos de lecturas en formato fastq y un posible diseño para el grafo de sobrelapes. La especificación del formato fastq se puede encontrar en el siguiente enlace:

https://en.wikipedia.org/wiki/FASTQ_format

El programa se puede ejecutar a partir de la clase “ReadsAnalyzerExample”. Los parámetros del programa son los siguientes:

args[0]: Comando a ejecutar. Puede ser “Overlap” o “Kmers”
args[1]: Archivo con las lecturas a analizar en formato fastq (o fastq.gz)
args[2]: Parámetro opcional. Para el comando “Overlap” permite cambiar el sobrelape mínimo. Para el comando “Kmers”, permite cambiar el tamaño del k-mer

La carpeta data contiene 4 archivos de lecturas de ejemplo en formato fastq que provienene de un fragmento de texto en lenguaje natural. Al final el programa debe poder ensamblar de la mejor forma posible el texto original partir de los diferentes archivos.

1. [25%] Implementar todos los métodos de la clase “KmersTable”. Probar el programa implementado en la clase “ReadsAnalyzerExample” utilizando el comando “Kmers” y las lecturas dadas. Probar con cada fastq de ejemplo tamaños de k-mer 5, 10, 20, 50 y 75 y registrar la cantidad de k-mers, la abundancia media y el tiempo de ejecución.

2. [25%] De la clase “OverlapGraph” implementar los primeros métodos hasta el método “calculateOverlapDistribution”. Probar el programa implementado en la clase “ReadsAnalyzerExample” utilizando el comando “Overlap” y las lecturas dadas.

Reportar para cada fastq de ejemplo disponible en la carpeta “data” la distribución de abundancias de secuencias, la distribución de sucesores por secuencia y el tiempo de ejecución.

3. [20%] Implementar los métodos faltantes de la clase “OverlapGraph”. Correr el programa para verificar si el programa trata de ensamblar una secuencia. Reportar la secuencia que se genera y el tiempo de ejecución para cada fastq de ejemplo. Reportar para cada fastq entre qué tamaños de sobrelape se puede ensamblar la secuencia.

4. [20%] Completar el script “SimpleReadsSimulator” para generar lecturas aleatorias de una secuencia de por lo menos 10Kbp en formato fasta. Probar los comandos “Overlap” y “Kmers” de “ReadsAnalyzerExample” con tamaños de secuencia 50, 100, 200, 500 y con profundidad promedio de 5X, 10X, 20X, 50X y 100X. Reportar tiempos de ejecución de los algoritmos en cada caso. Determinar la cantidad de lecturas máxima que puede procesar cada comando con un máximo de memoria de 4Gb.

5. [10%] Mejorar el script reads simulator incluyendo un parámetro nuevo que permita simular una tasa de error aleatoria. Simular datos con un tamaño de lectura de 50bp y profundidad promedio de 20X. Visualizar y comparar la distribución de abundancia de k-mers en cada caso

Bono [10%]: Buscar y descargar de la base de datos Sequence Read Archive (SRA) lecturas reales de secuenciación de genoma completo con Illumina de alguna bacteria y buscar también en NCBI un genoma ensamblado de la misma bacteria. Ejecutar el comando “Kmers” con la mayor cantidad de lecturas que sea posible y reportar la distribución de abundancias de k-mers. Comparar esta distribución con la distribución que se obtiene con lecturas simuladas sin errores con la misma profundidad promedio. Reportar si se ven diferencias importantes entre las distribuciones y las posibles causas de estas diferencias.

En caso de trabajar en otro lenguaje de programación, se debe desarrollar un programa que cumpla todos los requerimientos del programa descrito en los puntos anteriores, es decir, que incluya las siguientes funcionalidades:

1. Calcular la distribución de conteos de k-mers a partir de un archivo de lecturas en formato fastq
2. Realizar un ensamblaje de-novo utilizando el algoritmo de Overlap-Layout-Consensus a partir de un archivo de lecturas en formato fastq.
3. Simular lecturas a partir de una cadena en formato fasta.

Se debe incluir en la entrega el código fuente del programa y un archivo README.txt que incluya instrucciones claras sobre cómo ejecutar el programa para cumplir los diferentes requerimientos.

Entrega

Se debe entregar un archivo zip con el programa desarrollado incluyendo el código realizado para cada uno de los puntos de la tarea. Dentro de la carpeta del proyecto se debe entregar un archivo pdf que tenga un reporte de análisis de los resultados de los experimentos que incluya las gráficas de las distribuciones solicitadas junto con una discusión de los resultados.

Ejecución de programas en java que usan librerías por linea de comandos

La opción “-cp” del comando java recibe en general rutas a todos los archivos .class que necesita el programa para trabajar. Por ejemplo en este caso se utilizan algunas clases del jar “NGSEPcore_3.2.0.jar” disponible en la carpeta lib del proyecto. Estando en la linea de comandos en la carpeta “ReadsAnalyzer”, el comando para ejecutar el programa “ReadsAnalyzerExample” con un máximo de 4Gb de memoria disponible en linux o MAC sería así:

```
java -Xmx4g -cp lib/NGSEPcore_3.2.0.jar:bin uniandes.algorithms.readsanalyzer.ReadsAnalyzerExample
```

y en windows sería así:

```
java -Xmx4g -cp lib\NGSEPcore_3.2.0.jar;bin uniandes.algorithms.readsanalyzer.ReadsAnalyzerExample
```

una vez lo ejecuten de esta forma, el programa debe escribir:

```
Command and input file are mandatory parameters
```

Indicando que se deben agregar los parámetros obligatorios. En general, se pueden poner rutas (relativas o absolutas) hacia todas las carpetas que tengan un esquema de paquetes finalizado por archivos .class o hacia archivos .jar. Para el caso de los archivos .jar, se debe incluir la ruta completa al archivo, no solo a la carpeta en la que se encuentra. Cada ruta se separa por “:” en linux o MAC y por “;” en Windows.