ISIS 1105 Diseño y Análisis de Algoritmos
Semestre 2024-20. Proyecto – PARTE 3
Entrega: Viernes, 6 de diciembre de 2024 6:00PM (por Bloque Neon)

0 OBJETIVOS

- Diseñar soluciones computacionales para problemas.
- Estimar costos de las soluciones planteadas.
- Implementar soluciones.

Se premiarán las mejores soluciones y se castigarán las peores, en cuanto a optimización y eficiencia en tiempo y espacio.

1 CONDICIONES GENERALES

El proyecto se divide en tres partes independientes entre sí. Este documento describe la PARTE III. Cada parte contiene un problema a resolver mediante soluciones implementadas en `Java` o `Python`.

Para cada problema se pide:
- Descripción de la solución.
- Análisis temporal y espacial.
- Una implementación en Java o Python

2 DESCRIPCIÓN DEL PROBLEMA

Subtipos en el cerebro de un Ultralisk

Teniendo en cuenta el éxito de los primeros experimentos, los terrans continuaron sus investigaciones y se dieron cuenta que las células calculadoras se pueden a su vez dividir en subtipos dependiendo de su capacidad para enviarse mensajes entre ellas. Todas las células que pertenecen a un mismo subtipo tienen la capacidad de enviarse mensajes entre ellas. Algunos pares de diferentes subtipos también pueden enviarse mensajes.

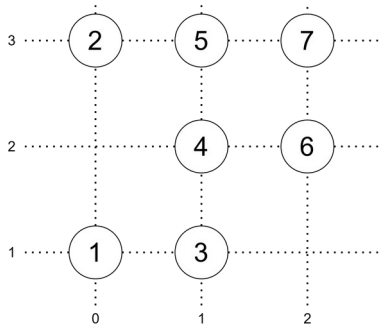
A continuación se recuerda el mecanismo de envío de mensajes entre células. Este mecanismo está mediado por la distancia entre las células y por los péptidos que presentan las células en su superficie. Gracias a tecnologías de mapeo espacial de células se puede contar con una distribución en un plano de dos dimensiones de las células que componen el cerebro. A partir de esta información se verificó que dos células pueden transmitirse mensajes directamente si están a menos de una cierta cantidad de unidades de distancia. Para dos células que pueden enviarse mensajes, la cantidad máxima de mensajes que se pueden enviar corresponde a la cantidad de peptidos compartidos que presentan las dos células en su superficie. De cada célula se sabe cuáles peptidos presenta y de cada peptido se conoce su secuencia característica de 5 aminoácidos.

Problema

Dada una distribución en dos dimensiones de las células cerebrales de un ultralisk, una distancia d y las secuencias de los péptidos que presenta cada célula, agrupar las células en la mínima cantidad de grupos posible, de modo que todos los pares de células dentro de cada grupo se puedan enviar mensajes entre ellas.

Ejemplo:

Para un conjunto de células distribuidas de acuerdo con la siguiente gráfica y los péptidos presentados por cada célula dados por la siguiente tabla:



Célula	Péptidos
1	AETQT, DFTYA, PHLYT
2	DSQTS, IYHLK, LHGPS, LTLLS
3	AETQT, DFTYA, HGCYS, LSVGG, SRFNH
4	DFTYA, HGCYS, IYHLK, SRFNH
5	DSQTS, IYHLK, LSVGG, LTLLS, TTVTG
6	AETQT, HGCYS, IYHLK, LSVGG, LTLLS
7	HGCYS, SRFNH, TTVTG

si d=1, Se necesitaría armar 4 grupos `{{1,3},{2,5},{4,6},{7}}`. Por otra parte, si d=2, se podrían armar 3 grupos: `{{1,3},{2,4,5},{6,7}}`.

3 ENTRADA Y SALIDA DE DATOS

En todas las soluciones que se presenten, la lectura de los datos de entrada se hace por la entrada estándar; así mismo, la escritura de los resultados se hace por la salida estándar.

Puede suponer que ninguna línea de entrada tiene espacios al principio o al final, y que los datos que se listan en cada línea están separados por exactamente un espacio.

A continuación, se establecen parámetros que definen su tamaño y formato de lectura de los datos, tanto de entrada como de salida.

Descripción de la entrada

La primera línea de entrada especifica el número de casos de prueba que contiene el archivo. El programa debe terminar su ejecución, una vez termine de resolver la cantidad de casos de prueba dados por este número.

Cada caso de prueba comienza con 2 números naturales que representan n y d, donde n representa la cantidad de células y d representala distancia máxima entre células para que se puedan enviar mensajes. Luego de esto hay n lineas. Cada linea inicia con 3 números naturales que representan el identificador de la célula y las coordenadas x, y de la célula en el plano. Estos números son seguidos por cadenas de 5 caracteres que representan los péptidos presentados por la célula. Recuerde que para esta entrega todas las células de la entrada son células calculadoras.

Restricciones para casos de prueba. Siendo m el total de aminoácidos: $1 \leq n \leq 10^5 \ \wedge \ 1 \leq m \leq 10^2$

Descripción de la salida

Para cada caso de prueba, la salida deben ser n lineas con 2 números enteros: el identificador de la célula y el número del grupo al que pertenece. Los grupos deben numerarse de 1 hasta k, siendo k el número de grupos.

Ejemplo de entrada / salida

Entrada	Salida
3 7 1 1 0 0 AETQT DFTYA PHLYT 2 0 2 DSQTS IYHLK LHGPS LTLLS 3 1 0 AETQT DFTYA HGCYS LSVGG SRFNH 4 1 1 DFTYA HGCYS IYHLK SRFNH 5 1 2 DSQTS IYHLK LSVGG LTLLS TTVTG 6 2 1 AETQT HGCYS IYHLK LSVGG LTLLS 7 2 2 HGCYS SRFNH TTVTG 7 2 1 0 0 AETQT DFTYA PHLYT 2 0 2 DSQTS IYHLK LHGPS LTLLS 3 1 0 AETQT DFTYA HGCYS LSVGG SRFNH 4 1 1 DFTYA HGCYS IYHLK SRFNH 5 1 2 DSQTS IYHLK LSVGG LTLLS TTVTG 6 2 1 AETQT HGCYS IYHLK LSVGG LTLLS 7 2 2 HGCYS SRFNH TTVTG 4 1 1 0 0 AETQT DFTYA 2 0 1 AETQT HGCYS 3 1 0 DFTYA IYHLK 4 1 1 HGCYS IYHLK	1 1 2 2 3 1 4 3 5 2 6 3 7 4 1 1 2 2 3 1 4 2 5 2 6 3 7 3 1 1 2 1 3 2 4 2

Nota: Se van a diseñar casos de prueba para valores de n y m mucho más grandes y dentro de los valores establecidos en el enunciado. Los casos mostrados en este documento son demostrativos de la estructura de entrada/salida esperada.

4 COMPRENSIÓN DE PROBLEMAS ALGORITMICOS

A continuación, se presentan un conjunto de escenarios hipotéticos que cambian el problema original. Para cada escenario debe contestar[1]: (i) que nuevos retos presupone este nuevo escenario -si aplica-?, y (ii) que cambios -si aplica- le tendría que realizar a su solución para que se adapte a este nuevo escenario?

ESCENARIO 1: Para cada grupo se permite que máximo un par de células dentro del grupo no se puedan mandar mensajes entre sí.

ESCENARIO 2: Se quiere agrupar las células en la mínima cantidad de grupos posible, pero con una restricción adicional en la que cada grupo es de máximo un tamaño T, donde T es un parámetro de entrada.

Nota: Los escenarios son independientes entre sí.

5 ENTREGABLES

El proyecto puede desarrollarse por grupos de hasta dos estudiantes de la misma sección. La entrega se hace por bloque neon (una sola entrega por grupo de trabajo).

[1] NO tiene que implementar la solución a estos escenarios, el propósito es meramente analítico.

El grupo debe entregar, por bloque neon, un archivo de nombre `proyectoDalgoP3.zip`. Este archivo es una carpeta de nombre `proyectoDalgoP3`, comprimida en formato `.zip`, dentro de la cual hay archivos fuente de soluciones propuestas y archivos que documentan cada una de las soluciones.

5.1 Archivos fuente de soluciones propuestas

Todos los programas implementados en *Java* o en Python

Para el problema:

- Entregar un archivo de código fuente en *Java* (`.java`) o python (`.py`) con su código fuente de la solución que se presenta.
- Incluir como encabezado de cada archivo fuente un comentario que identifique el (los) autor(es) de la solución.
- Denominar `ProblemaP3.java` o `ProblemaP3.py` el archivo de la solución que se presente.

Nótese que, si bien puede utilizarse un *IDE* como *Eclipse* o Spyder durante el desarrollo del proyecto, la entrega requiere incluir solo un archivo por cada solución. El archivo debe poderse compilar y ejecutar independientemente (sin depender de ninguna estructura de directorios, librerías no estándar, etc.).

5.2 Archivos que documentan la solución propuesta

La solución al problema debe acompañarse de un archivo de máximo 3 páginas que la documente, con extensión `.pdf`. El nombre del archivo debe ser el mismo del código correspondiente (`ProblemaP3.pdf`).

Un archivo de documentación debe contener los siguientes elementos:

0 Identificación
Nombre de autor(es)

1 Algoritmo de solución
Explicación del algoritmo elegido. Si hubo alternativas de implantación diferentes, explicar por qué se escogió la que se implementó. Generar al menos una gráfica que apoye la explicación del algoritmo implementado. No se debe copiar y pegar código fuente como parte de la explicación del algoritmo.

2 Análisis de complejidades espacial y temporal
Cálculo de complejidades y explicación de estas.

3 Respuestas a los escenarios de comprensión de problemas algorítmicos.
Respuesta a las preguntas establecidas en cada escenario. NO tiene que implementar la solución a estos escenarios, el propósito es meramente analítico.

La nota del informe corresponde a un 50% de la nota total de la entrega del proyecto, solamente si se entrega el código fuente de la solución implementada. En caso de no entregar el código fuente, no se hará evaluación del informe y la nota del proyecto será cero.

Además de la pertinencia del texto como explicación de la solución implementada, se evaluará la calidad en la redacción del texto y en el diseño de las gráficas. Se evaluará también que la explicación del algoritmo integre los conceptos relacionados con las técnicas de diseño de algoritmos cubiertas en el curso.

Téngase en cuenta que los análisis de 2 tienen sentido en la medida que la explicación de 1 sea clara y correcta. No se está exigiendo formalismo a ultranza, pero sí que, como aplicación de lo estudiado en el curso, se pueda describir un algoritmo de manera correcta y comprensible.

5.3 Consideraciones sobre la entrega

Lea bien las recomendaciones de entrada/salida. Su proyecto será evaluado, en gran parte, por una máquina: si en la ejecución del programa el formato de salida no coincide perfectamente con lo requerido, la calificación de la ejecución será cero, sin importar la cantidad de código que haya desarrollado.

La forma en que se presenta la documentación debe respetarse, aunque su evaluación no sea tan automática como la del software. Hay que nombrar los archivos como se espera que se nombren, comprimirlos como se pide que se compriman, etc. Cualquier desviación en cuanto a lo que se pide produce deméritos en la calificación final.

El software se evalúa desde la linea de comandos de Linux. No hay problema en desarrollar en cualquier otro sistema (Mac, Unix, ...) siempre que se produzca código estándar en java o python. Cada problema se debe resolver en un (1) archivo.