Cuestiones

Cuestiones sobre circuitos y camino eulerianos

Añadir código Python y figuras cuando se juzgue necesario.

Cuestión 1:

Tras las funciones anteriores nos hemos dado cuenta de que las mismas no tratan sobre grafos NO dirigidos. Indicar brevemente que cambios harían falta, si alguno, para que nuestras funciones pudieran buscar caminos y circuitos eulerianos en tales grafos

Nuestras funciones están diseñadas para multígrafo dirigido, por ello llevan mas comprobaciones y pueden tener una cantidad mayor de líneas.

Las funciones realizadas si se podrían usar para grafos no dirigidos pero habría que modificar algunas comparaciones y asignaciones por ejemplo la función de adyacencia e incidencia no tendría sentido ya que podría salir y entrar del nodo por cualquier camino no utilizado, se usaría como una función para saber el numero de ramas que tiene cada nodo e ir restando ramas según se vayan visitando, esa sería otra modificación no se borraría solo la rama del nodo de donde se comienza sino que también del destino para no repetir.

En las funciones de comprobación si hay camino o no también cambiarían algunas comparaciones y otras incluso no harían falta porque poder recorrer un multigrafo dirigido es mucho más complejo que uno no dirigido.

En las funciones como unir los dos caminos no haría falta modificarla porque no tiene ninguna comparación ni asignación de una dirigido simplemente une dos cadenas en una.

Cuestión 2:

El criterio sobre existencia de caminos o circuitos eulerianos solo funciona cuando el grafo no contiene subgrafos disjuntos. ¿Como podríamos detectar dicha condición?

Las funciones de existencia de caminos o circuitos eulerianos que nosotros hemos creado creemos que poseen ese criterio, ya que comparamos en cada nodo las adyacencias e incidencias y que no se repita que dos nodos posean una adyacencia mas que las incidencias ya que serian dos posibles nodos para comenzar el camino o el circuito. Obviamente también comprobamos que la suma de las incidencias de todos los nodos sea igual a la suma de las adyacencias de todos los nodos, además de comprobar que el grafo es conexo.

La función de existencia de circuito llama a la función de existencia de camino y luego realiza la comprobación de que el nodo inicial sea el mismo que el final, por lo que la mayor comprobación está en la función de existencia de camino.

Cuestiones sobre reconstrucción de secuencias

Cuestión 1:

Si se tiene una secuencia P de longitud Lp y se usan lecturas (reads) de longitud Lr para reconstruirlas mediante un camino o circuito euleriano sobre un grafo G = (V, E), queremos estudiar cuantos nodos |V| y cuantas ramas |E| hay y cual sería el sparsity factor ρ.

Para ello vamos a hacer un pequeño estudio experimental de estas cantidades, fijando una longitud P suficientemente grande y estimando el número de vértices, ramas y el sparsity factor para diferentes valores de R. Escribir una función Python que estime dichos números de vértices, ramas y sparsity factor, describir sus resultados y comentarlos.

La función consiste en dejar fija la longitud de la secuencia y variar la longitud de lectura, desde un tamaño pequeño a uno grande, generamos secuencias y sus caminos correspondientes y los comprobamos.

Con esto, hemos visto que el tamaño de la secuencia no es significativo, mientras que lo que sí que lo es, son el número de vértices y de ramas, cuantas menos haya, mejor. Esto es porque si hay pocos nodos, significa que el tamaño de lectura es alto, y si además hay pocas ramas, hay pocas coincidencias entre los nodos, y con ello, hay menos caminos eulerianos, siendo más fácil encontrarlo.

Cuestión 2:

El coste de reconstruir una secuencia no solo depende de la búsqueda de un camino euleriano sino también del tiempo empleado en la construcción del grafo.

Discutir los costes asociados a las funciones spectrum(sequence, len_read), spectrum_2(spectr) y spectrum_2_undirected_graph(spectr) definidas más arriba de acuerdo a la implementación que se haya seguido y, en consecuencia, el coste total del algoritmo de reconstrucción de secuencias.

Según nuestra implementación, los costes son los siguientes:

Coste de spectrum:

El coste sería O(N – len_read) = O(N), dado que recorremos la secuencia, por completo pero dando “saltos”.

Coste de spectrum_2:

El coste es O(N), pues recorremos el l-espectro elemento por elemento para construir el (l-1)-espectro.

Coste de spectrum_2 _graph:

El coste es O(N^2), pues recorremos el espectro dos veces para generar el diccionario.

Coste total:

El coste total estaría formado por la suma de dichos costes, siendo el resultado 2*O(N) + O(N^2)