Universidad de los Andes
Ingeniería de Sistemas y Computación

Proyecto 2 Diseño y Análisis de Algoritmos

Camilo Andres Daza Ramirez - 201416461

Mateo Calderón Rincón - 202213589

Algoritmo de Solución

1. Validación de si es posible generar el enlace

El código implementado contiene diferentes partes y consideraciones que se van a explicar en este documento. La primera parte del código incluye una verificación que se hace para que desde el inicio, se descarten aquellas entradas que no pueden generar el compuesto (como pasa en el segundo caso que nos dan), para ello se hace una validación de todos los átomos ingresados por parámetro, en donde lo que se quiere ver es que todos los átomos a excepción de dos como máximo aparezcan un número par de veces en la entrada, esto quiere decir que esos átomos tienen una pareja para conectarlos mediante enlaces Toll y Boltz, y que los dos átomos que no aparecen un número par de veces van a ir al inicio o al final de la cadena, ya que no se podrían encontrar con otro átomo que fuera igual en carga y magnitud para conectarse, solo estarían conectados al otro átomo que se le da en la entrada. Por eso los átomos que aparecen un número impar de veces no pueden ser más de dos, ya que no podrían hacer parte de las conexiones del medio, lo que haría que la cadena estuviera desconectada. Si encontramos más de dos átomos de un elemento apareciendo de manera impar, descartamos el caso porque no es posible hacer la conexión.

Por lo tanto, en este caso después de realizar diferentes verificaciones veríamos que no es posible. Para ello implementamos una solución más rápida y confiable al proceso, armando un diccionario con cada nodo como llave y como valor el número de vértices iguales (carga y magnitud) para luego sacar el módulo 2, de las veces que ese vértice esté presente, para saber si esta un número par o impar de veces, si más de dos vértices están con un módulo 2 igual a 1, sabemos inmediatamente que para esa entrada no es posible continuar, lo que se ve reflejado en procesar_caso_prueba().

2. Creación de nodos

Después de que la entrada ha sido verificada adecuadamente, comenzamos con las bases para el grafo, para ello la función parte2() toma cada uno de los átomos que se tienen en la entrada y se les asigna un valor de id y de esCola, en donde id es un identificador único (no se deja el valor del átomo original ya que van a existir átomos con el número) y un valor booleano de esCola, que representa si ese átomo debe estar en el inicio o al final de la cadena, ya que esta solo una vez y queremos evitar que suceda lo mencionado anteriormente. Adicionalmente, se añaden nodos que representan a los elementos en la naturaleza, estos serán iguales a los presentes en los elementos, 2 por cada número, el positivo y el negativo, por ejemplo si se tiene (1,3) como elemento, se agregan -1,1,3,-3 como nodos y se asigna el valor de esCola como False.

3. Creación de matriz de adyacencias

Teniendo esa información se pasa con la construcción de la matriz de adyacencia en la función parte3(), la cual tendrá como índices a todos los elementos que se detectaron en el paso anterior (átomos iniciales de los elementos, y átomos de la naturaleza), inicialmente en cada casilla de la matriz se tendrán valores de infinito, referenciando que no hay una conexión directa entre esos vertices. Luego pone en cero toda la diagonal y también las conexiones de aquellos átomos que están conectados desde la entrada, para forzar que dichas conexiones se mantengan en el grafo final, luego empieza a llenar la matriz de adyacencia para todos aquellos valores que pueden estar conectados, para ello se usa la función auxiliar calcular_peso_minimo() que hace los

cálculos correspondientes para los valores de LTP, actualizando la matriz en caso de que el valor descubierto sea menor al que se encuentra en la matriz.

Después de tener la matriz de adyacencia, se consideraron varias opciones, ya que ahora no solo se debía pasar por el camino más corto, sino también solo podíamos pasar como máximo una vez por todos los nodos. Inicialmente, se consideró la alternativa de usar el algoritmo de Floyd-Warshall, que tomaba la matriz de adyacencia construida anteriormente y devolvía la matriz de distancias más cortas, esto con el fin de asegurar la premisa del enunciado. Después con esa matriz, se usaba un algoritmo de Hamilton adaptado para que usase esa matriz, asegurándose que solo se pasase una vez por cada nodo. Pero esa implementación tenía algunos problemas, ya que pasaba obligatoriamente por cada nodo, es decir, incluyendo todos los elementos de la naturaleza en la solución.

Luego de esto, se intentó establecer condiciones al momento de crear la matriz de adyacencias para que las colas no tuvieran conexiones además del nodo de su elemento y por medio de un dijkstra semi-modificado que se generara el menor camino que fuera desde la cola1 hasta la cola2. No obstante, el algoritmo aun ignora los elementos del intermedio y busca los caminos óptimos con los elementos de la naturaleza, dejando por fuera elementos que necesariamente deben estar en el camino de solución. En este momento la solución no genera los caminos correctos para casos de n elementos, donde n>2.

Análisis de complejidades espacial y temporal

Complejidad temporal: Los algoritmos de la parte 1 y 2 corren en función de n, el número de masas distintas recibidas, ahora la parte 3 corre en función de (3n)^2 al momento de crear la matriz, y de O(3n+3n^2)log(3n) el algoritmo Dijkstra utilizado. Por lo anterior la complejidad del algoritmo es de O(3n+3n^2)log(3n) donde n corresponde a las distintas masas recibidas en los elementos de la entrada.

Complejidad espacial: el algoritmo guarda 2 arreglos que corresponden a los números de los elementos (el cual como máximo sería de tamaño n si todos los números son diferentes, y el otro corresponde al arreglo de nodos, el cual es igual en el peor caso a 3n, en donde se tendría los n nodos de los elementos y 2n adicionales por cada número como nodo en la naturaleza. Adicionalmente, el algoritmo usa una matriz de adyacencia de máximo 3nx3n, en donde se ubicarían todos los nodos del ejercicio y el peso de ir de uno al otro. Por último, la complejidad espacial del algoritmo Dijkstra utilizado fue de O(3n+3n^2)

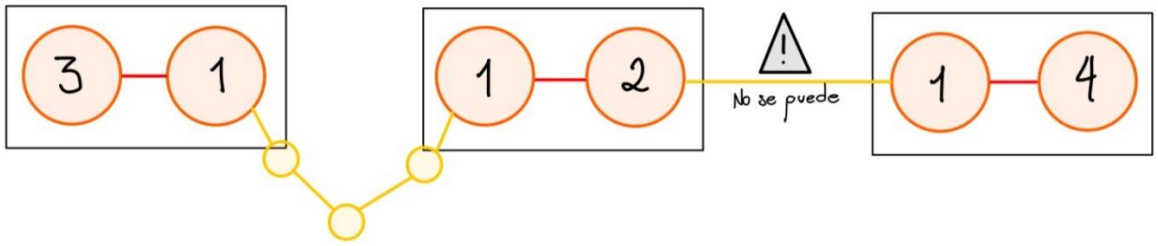
Respuestas a los escenarios de comprensión de problemas algorítmicos

ESCENARIO 1: Se admiten como respuesta compuestos en los que un elemento aparezca más de una vez.

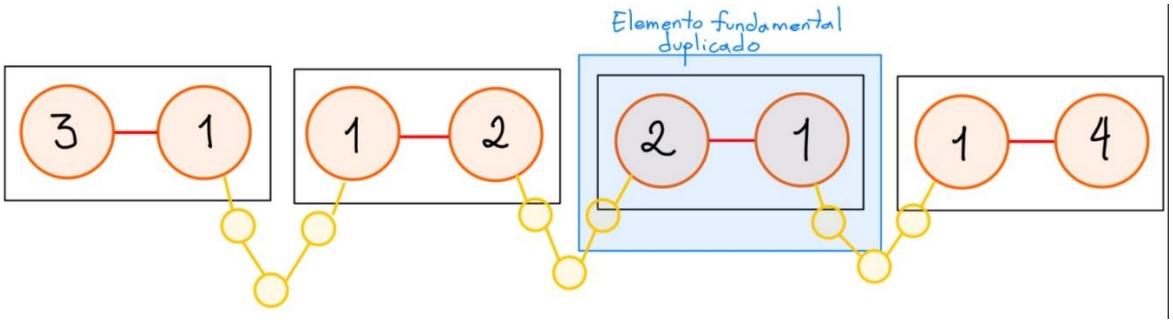
Este escenario requeriría ajustar nuestra solución al momento de leer la entrada no sería necesario validar que no esté repetido un mismo elemento, aun se mantendría la condición de no poder formar enlaces starks con elementos de la naturaleza así que esto se mantendría igual. Por otro lado, esto ayudaría en algunos casos para conectar aquellos elementos fundamentales sin conexión posible, ya que si pudiéramos tenerlos más de una vez aseguraríamos que podríamos conectarlos entre ellos, en el segundo caso del enunciado podríamos conectar (2,1) con su copia invertida, es decir, (1,2), lo cual nos ayudaría a que aquellos elementos fundamentales que no habíamos podido conectar con ningún otro, ahora si tengan una conexión. Pero eso no es suficiente, ya que a la hora de intentar conectarlos con todo el compuesto formado seguiríamos sin poder hacerlo en el caso del ejemplo, lo que nos resultaría en exactamente el mismo resultado. Aunque es necesario mencionar que, si podía funcionar para algunos casos, y este ejemplo lo comprueba.

Elementos Fundamentales: (3,1), (1,2), y (1,4).

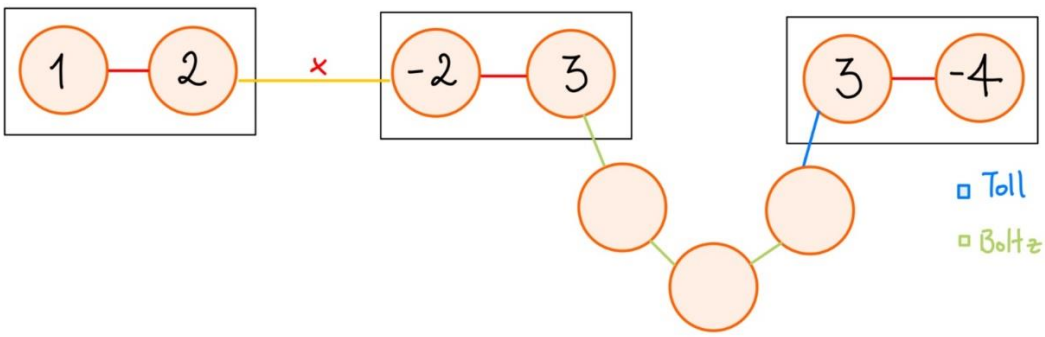
En el caso general obtendríamos este resultado:



Con el ajuste propuesto en este escenario:



ESCENARIO 2: Se admiten como respuesta compuestos que incluyan directamente enlaces toll para enlazar elementos fundamentales. Este escenario haría que para unir dos compuestos ya no se necesitaran enlaces boltz, solo toll, lo que facilitaría mucho más el problema. En primer lugar, ya no se debería verificar que cada átomo (mirando su carga y magnitud) apareciera 2n veces (pares veces), sino que ahora se podría verificar el valor absoluto de esos átomos, lo que le quitaría una restricción ya que el segundo caso del enunciado se podría ejecutar, uniendo el 2 con –2 solamente usando el enlace toll. Adicionalmente, para el resto de los casos sabríamos que tenemos una buena conexión solamente usando el enlace toll, y ahora el uso de enlaces boltz solo quedaría para verificar que el valor de la conexión sea lo más bajo posible, pero ya no sería una restricción. Y de esta manera, sabríamos que como resultado tendríamos n-1 enlaces para unir elementos fundamentales, y que a partir de eso podríamos incluir los enlaces boltz para que el grafo sea más optimo. Como resumen del escenario propuesto obtendríamos el siguiente resultado para el caso del enunciado.



Con la modificación del escenario ya sería posible enlazar todos los elementos fundamentales: