Informe Diego Aarón Moreno Galván

Tarea 3: Least common ancestor

Programación eficiente

**Descripción de la creación y lectura de árboles aleatorios.**

Para crear los árboles realicé una función en la cual se ponía un nodo raíz con un valor random, después se generaba un número aleatorio n. Si n era par, el nodo tendría un hijo izquierdo, si era impar no tendría y se marca con -1 la posición donde iría el hijo izquierdo. De la misma manera se hace para ver si va a tener o no un hijo derecho. Se realiza esto sucesivamente para cada nodo y en principio se podría generar cualquier árbol con este método.

El árbol lo guardo en un txt para leerlo y hacer pruebas sobre que nodos preguntar el LCA. Claramente, primero ejecutar la creación de árbol, verlo, cambiar los parámetros de los nodos deseados y ejecutar el programa para resolver el LCA.

**Descripción del algoritmo.**

Se programó un algoritmo O(N) sencillo, el cual solo calcula los niveles y después sube por el padre de cada nodo hasta que lleguen a ser el mismo ancestro.

Después, se programó el algoritmo de jump pointers, donde se modificaron los padres de cada nodo, es decir, si un nodo está a nivel h, su padre ahora será el ancestro que está a floor(log2(h)) ancestros de él. De esta manera se reduce el número de veces que subimos para ver el ancestro de los nodos.

Luego, se realizó el algoritmo con Ladder decomposition, el cual necesita primero hacer la descomposición long path, la cual se hizo como el código visto en clase. La descomposición Ladder se realizó actualizando el padre de cada nodo. Si un nodo esta en un long path de k, entonces el padre de dicho nodo será el que esté a 2k ancestros de él. El algoritmo de Ladder consiste en subir toda <<la escalera>> si al subirla no estará en la misma <<escalera>> que el otro nodo. En cuanto al subir un nodo la <<escalera>> llegue a la escalera del otro nodo, se para y se realiza el algoritmo normal O(N), pero ya acortamos la distancia así que se hará en O(logN).

Finalmente, se realizó el algoritmo híbrido O(1), el cual consiste en subir una vez de la manera jump pointers la cual es subir de un paso de floor(logh) y después subir de la manera Ladder explicada anteriormente.

Salida.

Se imprimen al final los resultados del normal, el jump pointers, el de Ladder y el híbrido entre los dos últimos para comparar los resultados. Quise comparar los tiempos, pero salían casi igual porque no pude construir un ejemplo lo suficientemente grande para que existiera diferencia.