Estructuras de Datos y Algoritmos - IIC2133
Control 5

1) Se tiene un *stream* s = d1 d2 d3 … de largo indefinido donde cada dato d = (u,v) representa una arista no direccional que se agrega a un grafo G inicialmente vacío. La idea es que G permanezca acíclico, por lo que si la arista a agregar forma un ciclo se detiene la lectura del stream y se retorna G. **Explica en detalle** cómo llevar a cabo este proceso, determinando de manera eficiente en cada paso si la arista a agregar forma un ciclo en G. **Cuidado**: no conoces todos los vértices por adelantado.

Propuesta de solución:

Mencionar que se puede resolver mediante una implementación de un algoritmo similar a Kruskal, mediante conjuntos disjuntos. En este caso, no es necesario ordenar las aristas (dado que llegan a través del stream).

Para cada elemento del stream d = (u,v), se revisa si ya existen. Debo revisar en una tabla de hash/arreglo dinámico de manera que la complejidad de determinar si los vértices fueron o no explorados anteriormente sea O(1) **[1 punto]**.

Si alguno no existe, se realiza makeset(u) y/o makeset(v) y se indica que estos son su propios representantes. Se agregan a la tabla de hash/arreglo dinámico **[1 punto]**.

Se hace find(u) y find(v), identificando a sus representantes **[1 punto]**.

Si tienen el mismo representante, significa que pertenecen al mismo conjunto y no los uno
Justificar su correctitud. Si dos vértices tienen el mismo representantes, implica necesariamente que pertenecen al mismo subconjunto (fueron unidos en algún momento por una arista del stream). De esta manera, el algoritmo debe finalizar su ejecución sin incorporar la arista que genera el ciclo. En caso de que no tengan el mismo representante, implica que no pertenecen al mismo subconjunto del grafo G, lo que implica que unirlos no generaría un ciclo **[2 puntos]**.

Si tienen diferente representante, los uno con Union(u,v) **[1 punto]**.

2)

a) El primer for toma un tiempo O(V). La línea 6 consiste en inicializar el min heap binario con todos los vértices, esto tiene complejidad de O(V) igualmente. **[0.5 puntos]**

En el loop while se recorren todos los vértices y en cada uno de ellos hay que extraer el menor, complejidad O(1), y reordenar el heap, complejidad O(logV), es decir, tiene complejidad O(VlogV). **[1.5 puntos]**

Para el caso del for dentro del while se recorrerán en total todas las aristas, y se puede tener que ordenar el heap para cada caso, por lo que tiene complejidad O(ElogV). **[1.5 puntos]**

La complejidad total sería O((E + V)logV). **[0.5 puntos]**

b) Cualquier ejemplo que demuestre lo dicho en el enunciado y que sea correcto **[2 puntos]**