TPO - Diseño y Análisis de Algoritmos III

∫ Alumnos: Martina Virgilli – Martin Ferreira
∫ Profesor: Guillermo Rodríguez
∫ Noviembre 2023

Etapas del Trabajo Práctico de Algoritmos de Grafos

Para este trabajo elegimos el lenguaje Python, ya que estábamos más familiarizados con este.

Etapa 1 – Investigación:

Antes de empezar, realizamos las investigaciones correspondientes para asegurarnos de hacer correctamente todas las instancias del trabajo. Investigamos tanto desde las diapositivas de clase como en distintos foros de internet y video tutoriales de Youtube para guiarnos.

Etapa 2 – Clase Grafo:

Ya que no disponíamos de una clase Grafo ya hecha y no encontramos ninguna por internet que nos convenciera, decidimos crearla nosotros mismos y así además practicar incluso como crear el grafo principal.

Etapa 3 – Primer algoritmo implementado:

Una vez que teníamos la clase Grafo y los apuntes necesarios comenzamos a crear la clase que contiene el primer algoritmo que elegimos, Prim. Elegimos este algoritmo ya que era uno sobre los que más información teníamos y más fácil se nos haca para arrancar. Tuvimos varios intentos fallidos, pero al final conseguimos que funcionara correctamente incluso mostrando en consola como resultaba el Árbol de expansión mínima al final.

Etapa 4 – Segundo algoritmo implementado:

Ya mas familiarizados con todo decidimos implementar el algoritmo de Floyd que, a pesar de no ser el más sencillo, aunque corto, pudimos hacer que funcionara al final indicando en consola como quedarían formados los pares de todos los nodos del

Grafo. Este nos costó un poco más, pero con ayuda de un poco mas de investigación y tutoriales pudimos resolverlo.

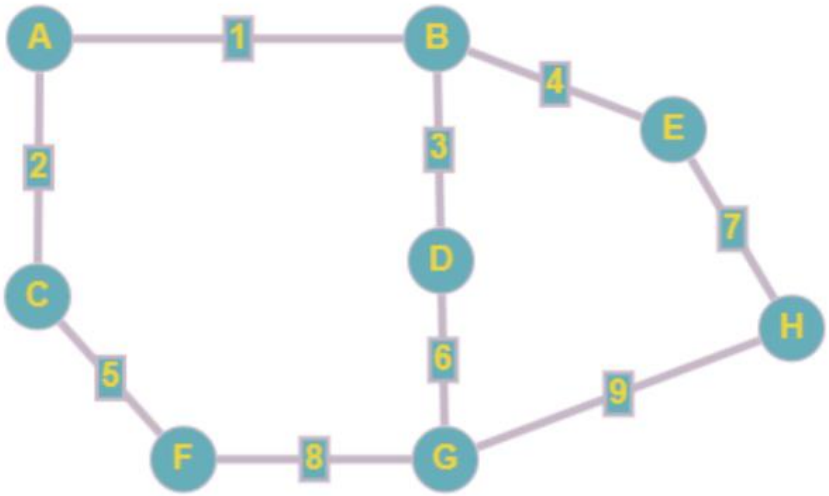
Hicimos comentarios en el paso a paso de cada función que nos ayudaron a nosotros a siempre saber exactamente lo que hacíamos y además ayudaran a que otros lo entiendan al leer el código.

También implementamos una cosa nueva que se verá en el código, que personalmente nunca lo habíamos usado y por eso lo aclaramos.

Biblioteca **heapq**: Esta biblioteca proporciona funciones para usar listas como colas de prioridad, en nuestro código la utilizamos para manejar la cola prioridad del algoritmo de Prim, ya que requiere agregar y eliminar vértices de la cola prioridad de forma más eficiente.

Aprendimos este elemento durante la investigación y decidimos implementarlo para hacer el algoritmo más eficiente.

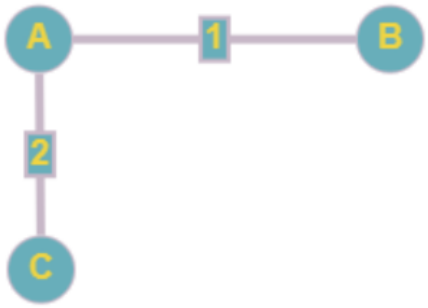
Representación del Grafo:



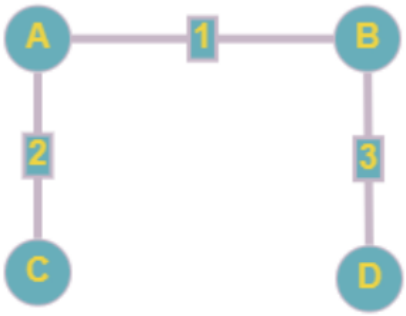
Algoritmo de Prim:

1

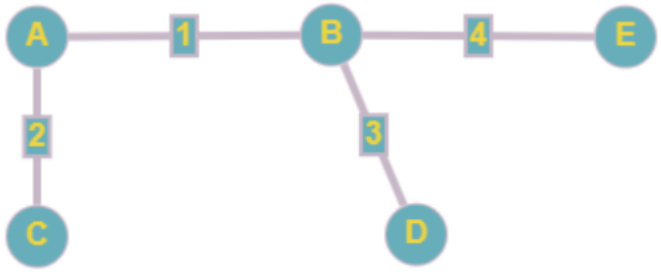
2



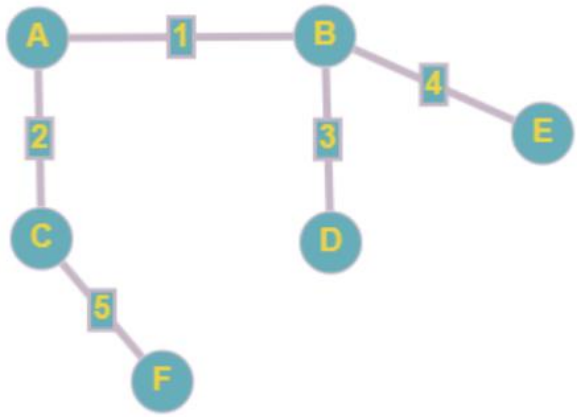
3



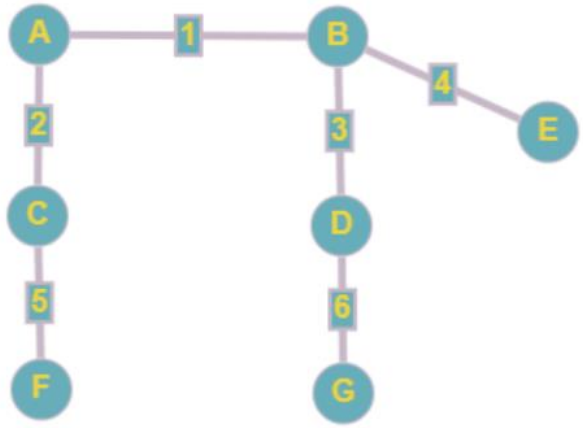
4



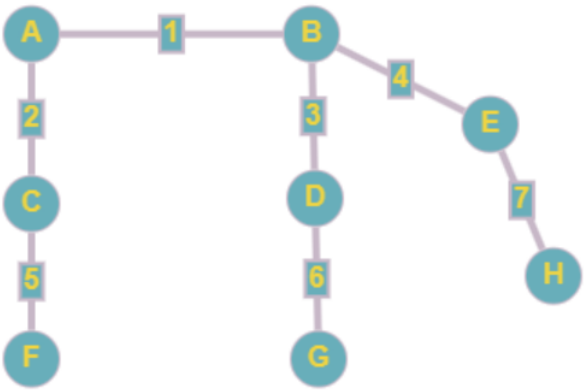
5



6



7



Algoritmo de Floyd: