# Proyecto Análisis de Algoritmos

Javier Esteban Sandoval Albarracín, Nombre del Tercer Autor, etc.
Ingeniería de Sistemas,
Pontificia Universidad Javeriana
Sandoval.javier@javeriana.edu.co, s.dominguez@javeriana.edu.co

***Resumen*- Este documento presenta el análisis primario respecto a los problemas presentados en el enunciado del proyecto semestral para la materia Análisis de Algoritmos.**

***Palabras Clave*- MST, Grafo, Prim, algoritmo, Kruskal, Dijkstra, Grafo, árbol, recurrencia, divide and conquer, arreglo, cola de prioridad.**

## I. Introducción

El análisis de algoritmos ha sido fundamental en el desarrollo de la computación moderna, desde el nacimiento de las matemáticas existen maneras de encontrar soluciones a problemas triviales y no tan triviales de la sociedad. Un algoritmo representa una serie de pasos con una entrada definida y una salida esperada, para esta definición nuestra solución en esta primera entrega representa una caja negra con la explicación detallada del problema y sus posibles soluciones.

## II. Glosario

*MST: Minimun Expanding Tree(Árbol de Expansión mínima)*

*LP: Linear Programing*

## III. MST

En esta sección se realizará la descripción detallada de los problemas planteados, de forma que el lector comprenda las instancias reales y las situaciones posibles que se pueden presentar.

El mínimo árbol de expansión se encuentra observando todos los subárboles dentro del grafo. De manera matemática se puede observar:

$$Dado\ un\ Grafo: G(v,e) y\ una\ arista\ e\ \rightarrow W: e \in R: \sum_{e \in T} w(e)$$

Cada arista de nuestro grafo es coloreada azul o roja, donde el número de aristas pertenecientes a cada color se rigen de la forma:

$$k\ aristas\ azules$$
$$n-k-1\ aristas\ rojas$$

**Entrada al algoritmo**: Para el algoritmo debe existir un archivo describiendo el grafo en texto plano con el número de aristas en la primera línea y luego una relación de tipo:

idNodo a…n(id nodos adyacentes) Azul-Rojo

**Salida:** Nodos Azules: k
Nodos Rojos: n – k – 1

### A. *Solución*

#### a. Algoritmo Voraz

Un algoritmo voraz es aquel que permite una substracción de las soluciones del árbol de manera tal que selecciona la solución que más se acomoda a un parámetro fijado. (Divide and Conquer)

7

Y la segunda parte del algoritmo es el Greedy Choice Property que toma opciones de solución local óptima de forma que estas me conduzcan a la solución global óptima.

Se sigue una Heurística aplicada a problemas de optimización, simples de forma que en la solución se me represente un árbol conectado aciclico.

#### *b. Algoritmo de Prim*

Debemos escoger un nodo S que será el punto de partida para el camino, existe paralelamente una cola de prioridad que almacena los nodos del grafo.

Pasos:

1. Invariante: $v.key = \min\{w(u,v) | u \in S\}$
2. Inicializar Todo{Cola, el grafo, invariante}
3. For $v \in V\{s\}\ v.key = \infty\ donde\ v\ representa\ los\ vértices.$
4. Hasta que la cola quede vacía:
   U: extrac_min(cola) se agrega u a subgrafo S
5. Sigue un pseudocódigo que se implementará en la segunda entrega

#### *c. Ejemplo Prim*

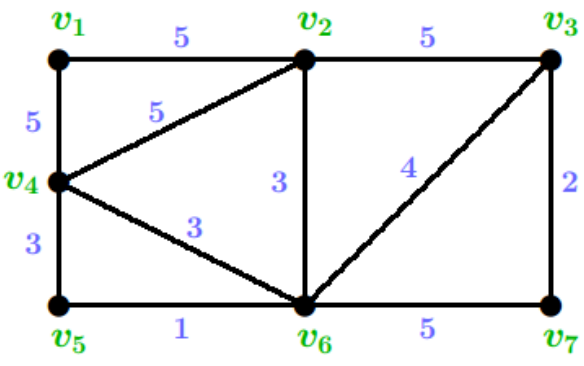
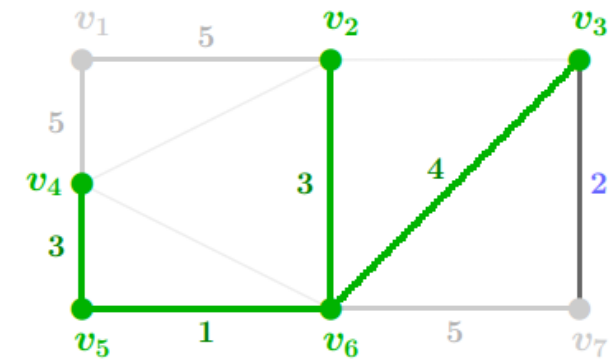



Figura 1. Grafo




$$W = \{ v_2, v_6, v_5, v_4, v_3$$

$$B = \{\{2,6\}\ \{5,6\}\ \{4,5\}\ \{3,6\}\ \ \ \ \ \ \}$$

Figura 2. Paso Intermedio Algoritmo Prim

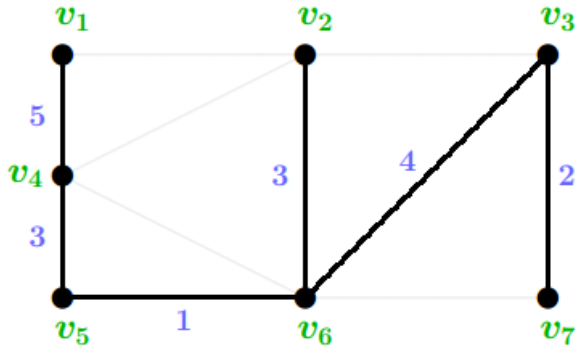



Figura 3. Solución final Algoritmo de Prim, donde Azules = 6, y Rojas = 5, donde el árbol son las aristas azules en este caso señaladas de Negro.

## IV. Distancia de Hamming

La distancia de Hamming en palabras sencillas es el número de elementos que varían de un arreglo a otro tal que los dos arreglos tengan un mismo tamaño n. Matemáticamente se define como una función *dist(u,v)* entre vectores U y V de n elementos tal que de los elementos $k_i$ para los dos vectores sean diferentes para i = j. Nuestro algoritmo debe hallar el MWC del grafo de haming tal que

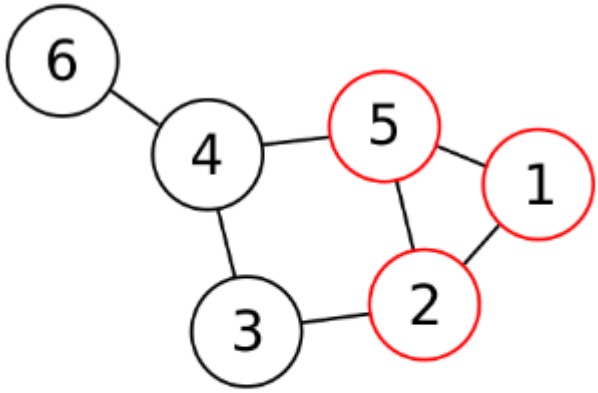
$$A(n,d) = \max |\{S \subset \{0,1\}\, n \mid \text{dist}(u,v) \geq d \text{ for all distinct } u,v \in S\}| \text{ y } A(n,d) = H(n,d)$$

donde n es el tamaño del grafo y d es la distancia de Hamming entre dos grafos. Se debe diseñar un algoritmo que me permita encontrar el máximo de arreglos que se pueden crear con dichas condiciones, es decir que difieran en esos *d* números. Las condiciones es que los vectores sean te tipo binario.

Entrada: Se generará un grafo de Hamming a partir de n = tamaño del vector, y m aristas del grafo.

Salida: número de subgrafos generados que sean completos dentro del grafo de Hamming.

A. Ejemplo:




The graph shown has one maximum clique, the triangle {1,2,5}, and four more maximal cliques, the pairs {2,3}, {3,4}, {4,5}, and {4,6}.

Figura 4. Problema de Cliques en Grafo[2]

Solución:

Salida: 1 Máximo clique que interconecta 3 nodos

Un Clique es el número máximo de componentes totalmente conectados dentro de un grafo, como se ilustra en la figura 4 el máximo número de componentes es 3 (sub-grafo completo).

Algoritmo Propuesto:

Nuestro algoritmo consta de listar todos los Cliques Máximos producidos por el grafo con una complejidad promedio de $2^n$ ya que siempre se recorren todos los vértices por lo tanto el número de vértices es igual al número de operaciones.

Algoritmo propuesto 2:

Para mejorar este algoritmo como segunda opción de solución podríamos implementar un algoritmo paralelo que recorra el grafo y guarde los que ya recorrió de manera que se evite recorrerlos varias veces. Dicha complejidad sería de la forma O(V log(v))

Algoritmo de Dijkstra: Resuelve el problema de hallar el camino de longitud mínima entre dos grafos, de aquí conocemos las complejidades aproximadas de nuestra propuesta.

## V. P LP

Para todas las tiendas que yo quiera crear, debo tener en cuenta las ubicaciones seleccionadas previamente, estas se correlacionan de manera tal que pueden construir una matriz y de dicha matriz podemos partir a un grafo. Para K tiendas cada tienda tiene una coordenada asignada (i,j) para una tienda Kn,

y unas coordenadas (x,y) para las ubicaciones seleccionadas. La distancia directa puede ser el peso de la arista que interconecte esas dos coordenadas.

A. Las aristas deben ser construidas a partir de la asignación del peso a una arista imaginaria entre los dos vértices, dichas aristas se crearán de manera implícita dada su relación por coordenadas y de esta manera crear un grafo a partir de la mattriz.
B. Ejemplo:

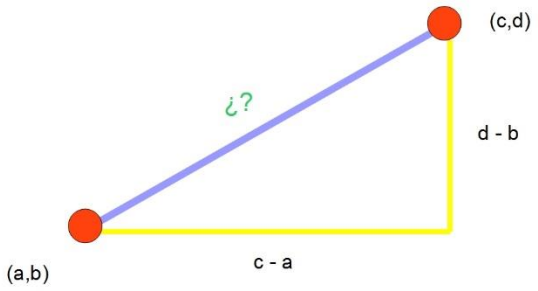



Figura 5. Distancia entre dos coordenadas.

La distancia dada para dos coordenadas se resuelve hallando la resta entre componentes de coordenadas

**Entrada:** Construcción del grafo a partir de las coordenadas dadas. (Matriz)

- La arista se crea implícitamente entre los dos puntos.
- La distancia se halla mediante la

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)}$$

**Salida:** La distancia d a la que deben recurrir los usuarios, dada cualquier ubicación me debe retornar el camino menor a una tienda (Algoritmo de Dijkstra) aplicado al grafo creado.

## VI. REFERENCIAS


[1] https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-046j-design-and-analysis-of-algorithms-spring-2015/recitation-videos/

[2] http://www.wikiwand.com/en/Clique_problem

[3]MIT: https://www.youtube.com/watch?v=tKwnms5iRBU