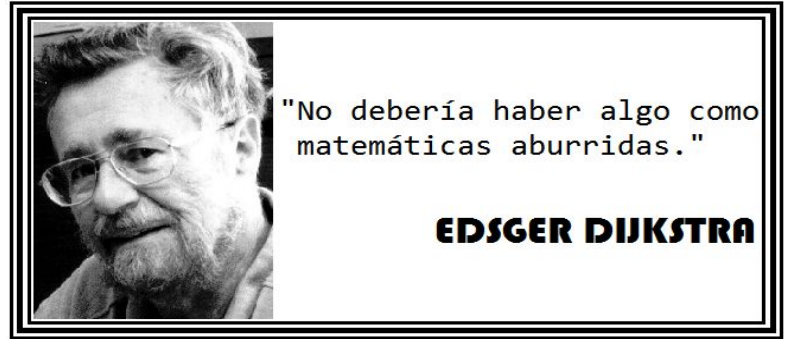
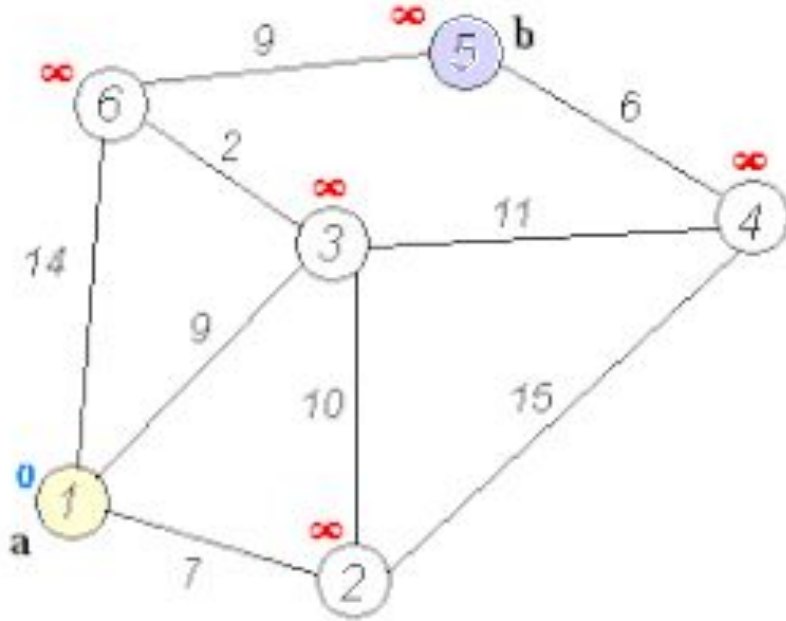


Algoritmo de Dijkstra

Análisis de Algoritmos 3CV18
Equipo Verde:

Reséndiz Díaz Marco Antonio
Santisbón Álvarez José Pablo
Soto Barragán Aldair David
Torres Limón Fernando
Valdovinos Castellanos Jesús Neftali
Valle Ortiz Edilberto Sergio
Vázquez Hernández Isaac

Algoritmo de Dijkstra



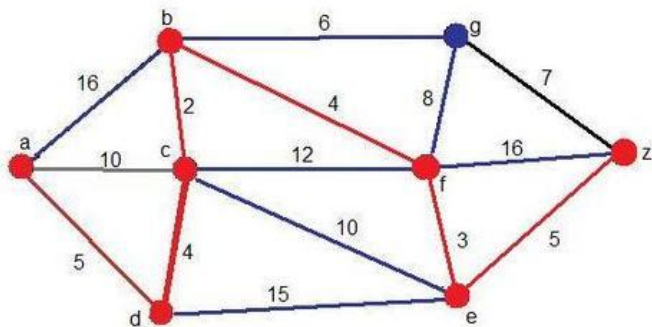
Antecedentes

Edsger Wybe Dijkstra nació en Rotterdam en 1930. Se especializó en informática generando aportes como lo son la notación polaca inversa (notación postfija), el algoritmo Shunting Yard, también fue una de los principales diseñadores del lenguaje de programación Algol.

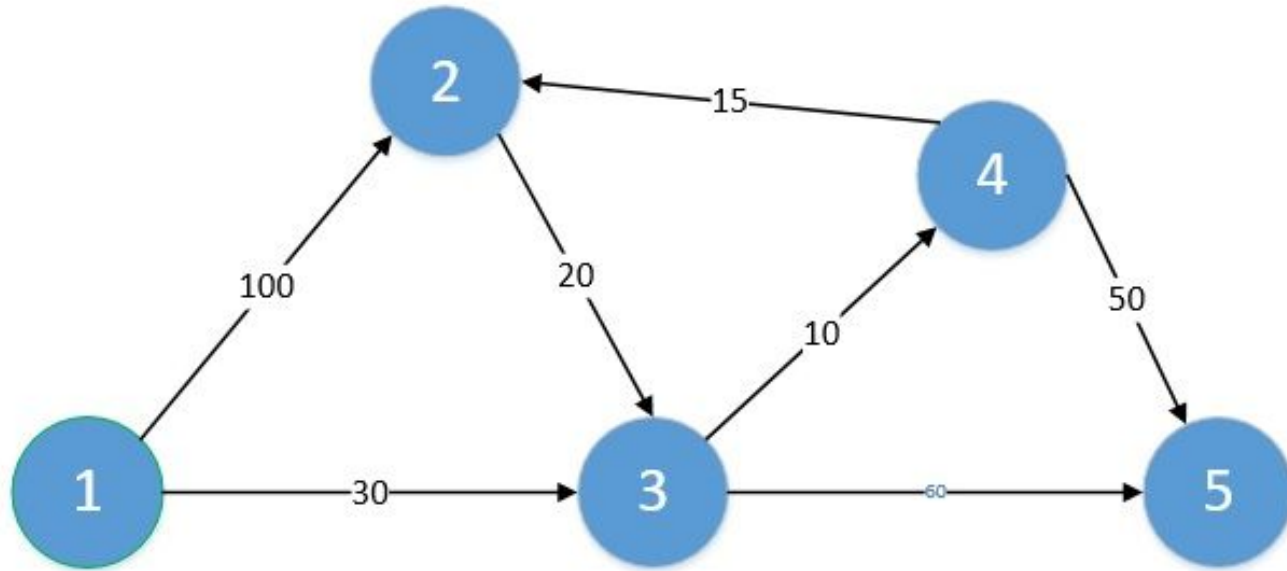


En 1956 anunció su algoritmo de caminos mínimos que desde entonces es conocido bajo su apellido, “El algoritmo de Dijkstra” surgiendo a partir del planteamiento de un problema propuesto que consistía en encontrar el camino más corto entre dos ciudades de los Países Bajos.

Consiste en ir explorando todos los caminos más cortos que parten del vértice origen y que llevan a todos los demás vértices, cuando se obtiene el camino más corto desde el origen al resto que componen al grafo, el algoritmo finaliza.



Problema de Ejemplo



Iteración 0

Etiqueta nodo 1 = [valor acumulado, procedencia] *iteración*

Etiqueta nodo 1 = [0, -] 0



Iteración 1

A partir del nodo 1 = [0, -] 0 observamos que los nodos que están relacionado con este son nodo 2 y el nodo 3,
ahora evaluamos:

Etiqueta nodo2 = [0+100,1](1)

Etiqueta nodo 3=[0+30,1](1)

observamos y nuestro nodo permanente será el nodo 3 y partiremos de aqui

para nuestro siguiente paso



Iteración 2

En este paso, evaluamos las posibles salidas desde el nodo 3, es decir los nodos 4 y 5. De manera que debemos asignar las etiquetas para cada nodo:

Etiqueta nodo 4 = [valor acumulado, procedencia] *iteración*

Etiqueta nodo 4 = [30 + 10, 3] 2

Etiqueta nodo 4 = [40, 3] 2



30 es el valor acumulado en la etiqueta del nodo de procedencia, en este caso el valor acumulado para el nodo 3. 10 es el valor del arco que une el nodo procedencia (3) y el nodo destino (4). 2 es el número de la iteración.

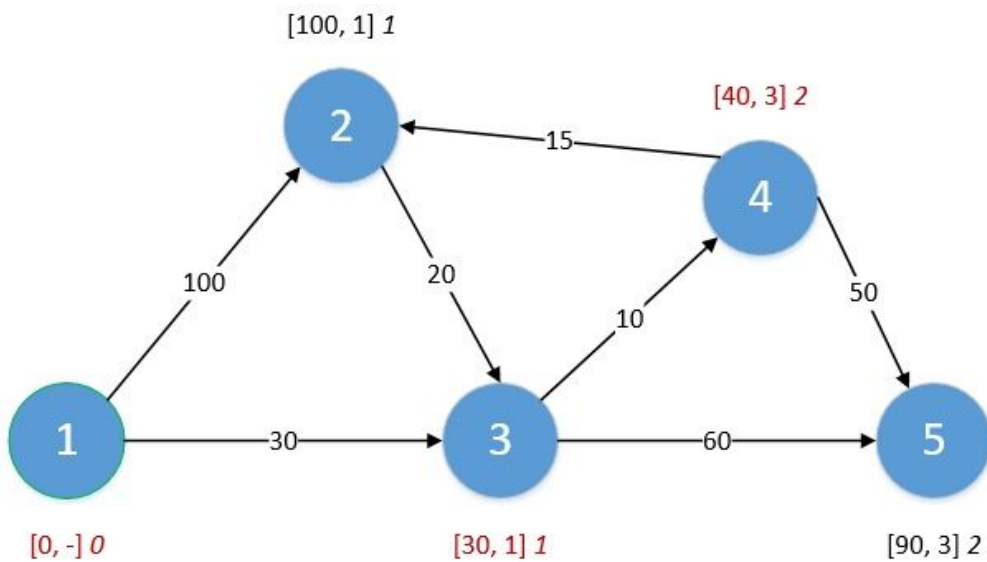
Etiqueta nodo 5 = [valor acumulado, procedencia] iteración

Etiqueta nodo 5 = [30 + 60, 3] 2

Etiqueta nodo 4 = [90, 3] 2

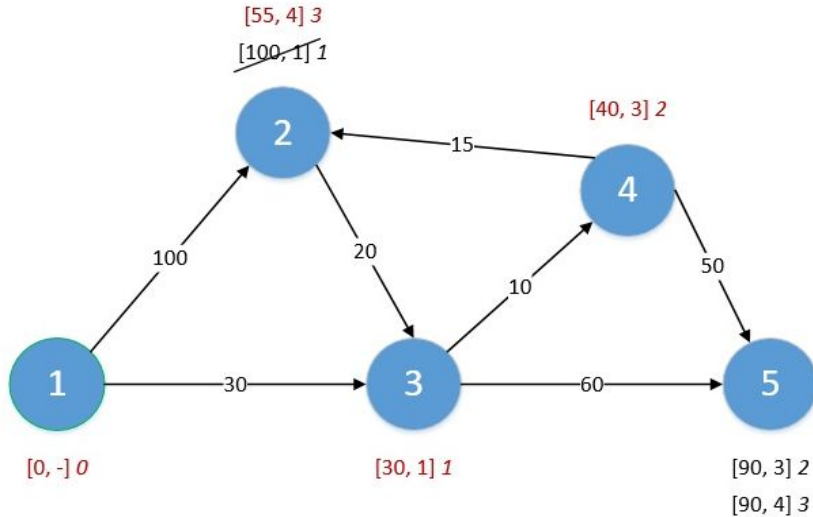
30 es el valor acumulado en la etiqueta del nodo de procedencia, en este caso el valor acumulado para el nodo 3. 60 es el valor del arco que une el nodo procedencia (3) y el nodo destino (5). 2 es el número de la iteración.

En este caso, podemos observar que la etiqueta del nodo 4, contiene el menor valor acumulado posible para llegar a este. Así entonces, la etiqueta del nodo 4 pasa a ser permanente.



Nodo	Etiqueta	Estado
1	$[0, -]$	Permanente
2	$[100, 1]$	Temporal
3	$[30, 1]$	Permanente
4	$[40, 3]$	Permanente
5	$[90, 3]$	Temporal

Iteración 3



En este paso, evaluamos las posibles salidas desde el nodo 4, es decir los nodos 2 y 5.

Etiqueta nodo 2= $[40+15,4]3$

Etiqueta nodo 5= $[40+50,4]3$

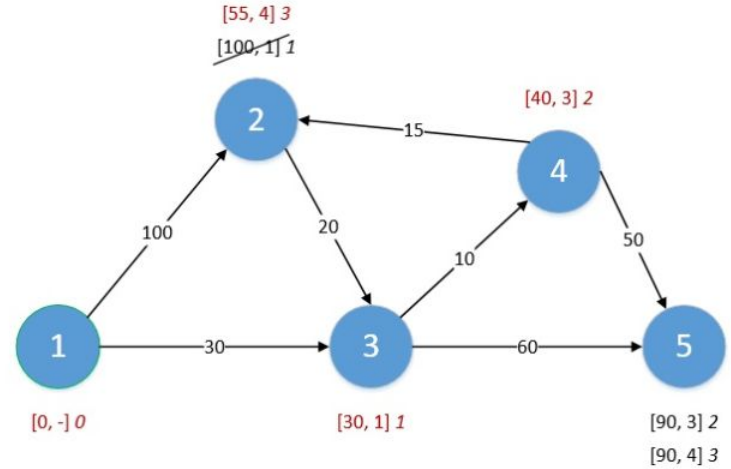
Nuestro nodo 2 pasará a ser permanente.

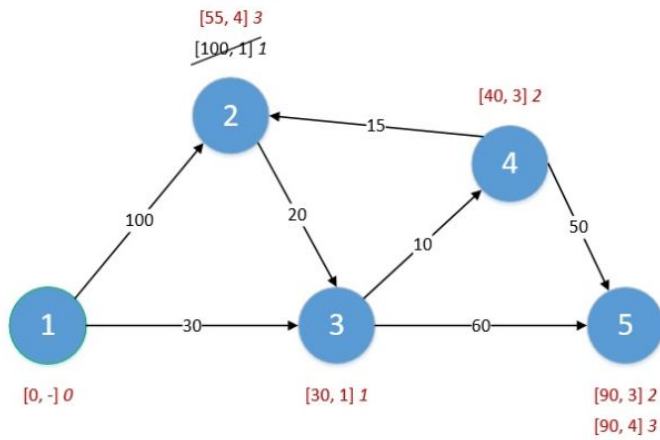
Y el nodo 5 tendrá otra etiqueta temporal.

Iteración 4

El siguiente paso es evaluar las salidas desde el nodo 2 y el nodo 5.

El nodo 3 siendo el único destino del nodo 2 no puede ser considerado ya que cuenta con etiqueta permanente y no puede ser reetiquetado.





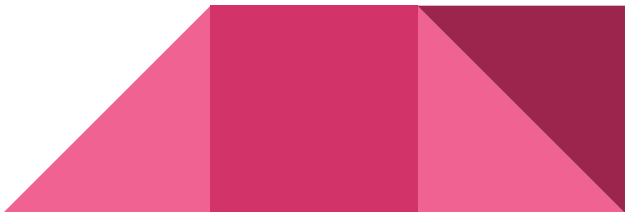
Se evalúa el nodo 5 y es un nodo que no cuenta con destinos.

La etiqueta del nodo 5 pasa de ser temporal a permanente.

Se puede observar que dicho nodo cuenta con 2 etiquetas que tienen el mismo valor, por lo tanto, tiene dos alternativas óptimas.

Implementación en Código (Algoritmo)

- El algoritmo de Dijkstra determina el camino más corto desde un vértice al resto del grafo .
- “En cada iteración se come el mejor pedazo”
- Dado un grafo dirigido $G=(V,A)$ valorado y con factores de peso no negativos, cada arco (V_i ,V_j) tiene asociado un coste C_{ij} . De tal forma que si V_0 es el vértice origen y $V_0, V_1,...,V_k$ es la secuencia de vértices que forman el camino de V_0 a V_k , entonces tiene la longitud del camino dada por:


$$\sum_{i=1}^{k-1} C_{i,i+1}$$


Algoritmo - C++

```
void Dijkstra (GrafoMatriz g, int origen){
    bool *F;
    F= new bool [n];
    //valores iniciales
    for(int i=0; i<n; i++){
        F[i] = false;
        D[i] = g.Ovalor_(s,i);
        ultimo[i] = s;
    }
    F[s] = true; D[s] = 0; //marca origen e inicializa distancia
    //Pasos para marcar los n-1 vértices. Algoritmo voraz
    for (int i=1; i<n; i++){
        int v=minimo(F);
        //selecciona vértice no marcado de menor distancia
        F[v] = true;
        //actualiza distancia de vértice de menor distancia
        for(int w=0; w<n;w++){
            if(!F[w]){
                if(D[v] + g.Ovalor_(v,w) < D[w]){
                    D[w] = D[v] + g.Ovalor_(v,w);
                    ultimo[w]=v;
                }
            }
        }
    }
}
```

Usos y Aplicaciones

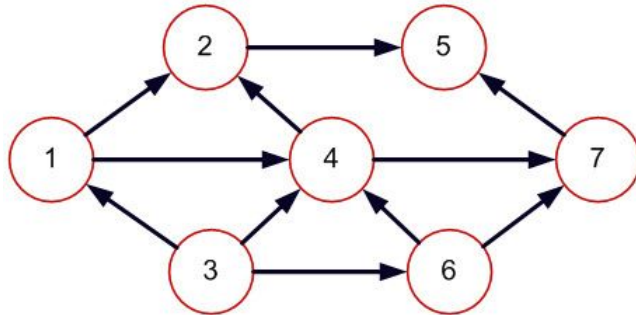
Las aplicaciones del algoritmo Dijkstra son muy diversas y de gran importancia en distintas áreas del conocimiento. Se presentan a continuación algunas de ellas:

- Encaminamiento de paquetes por los routers
 - Aplicaciones para Sistemas de información geográficos
 - enrutamiento de aviones y tráfico aéreo.
 - Tratamiento de imágenes,
 - Aplicaciones médicas.
 - Problemas de optimización de una función de coste para moverse entre diversas posiciones.
- 

Tipos de Redes que puede resolver el Algoritmo

Este algoritmo tiene la capacidad de resolver el problema de la ruta más corta, tanto para redes cíclicas, como para redes acíclicas. De manera tal que los bucles que presente una red, no restringen el uso del algoritmo.

Hace uso y define etiquetas a partir del nodo origen y para cada uno de los nodos subsiguientes. Estas contienen información relacionada con un *valor acumulado* del tamaño de los arcos y con la *procedencia más próxima de la ruta*.



Conclusiones

Como pudimos observar, el algoritmo de Dijkstra si bien, es un modelo que aborda el denominado problema de la ruta más corta; en la práctica, puede utilizarse para optimizar: distancia, costos, tiempo.



¿Por qué funciona este algoritmo?

La clave está en que:

- En cada paso, para todos los nodos u que ya fueron visitados, el algoritmo tiene calculada la mínima distancia del nodo inicial a u .

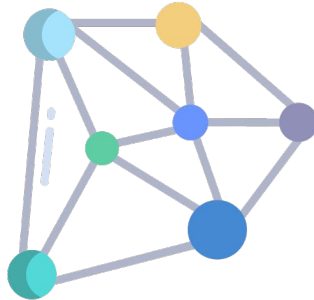
Para los nodos v que aún no fueron visitados, el algoritmo tiene calculada la distancia mínima si solo podemos utilizar nodos ya visitados como puntos intermedios del camino.



→ El próximo nodo a ser visitado es el más cercano al nodo inicial que aún no fue visitado. Entonces, este ya tiene calculada la distancia correcta.

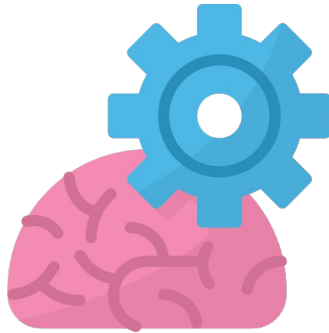
Si no fuese el caso, habría una mejor solución usando nodos no visitados como nodos intermedios, pero este nodo es el más cercano al origen entre todos los no visitados.

→ Al comienzo, las distancias están bien calculadas.



¡La complejidad computacional del algoritmo de Dijkstra no es alta!

Aunque el número de iteraciones necesarias es de $N-1$, siendo N el número de nodos total del grafo. Así, se puede decir que su complejidad es clase P (tiempo polinómico).



¡El algoritmo de Dijkstra es muy rápido siempre que el grafo tenga un tamaño reducido!



Referencias:

- ❖ Salazar, B., 2021. *Algoritmo de Dijkstra | Ingeniería Industrial Online*. [online] Ingeniería Industrial Online. Available at: <<https://www.ingenieriaindustrialonline.com/investigacion-de-operaciones/algoritmo-de-dijkstra/>> [Accessed 27 April 2021].
- ❖ Aplicaciones del Algoritmo Dijkstra. (2021). Retrieved 27 April 2021, from <https://www.ingenieradeideas.com/2014/04/las-aplicaciones-del-algoritmo-de.html#:~:text=Otras%20aplicaciones%3A%20enrutamiento%20de%20aviones,para%20moverse%20entre%20diversas%20posiciones.>
- ❖ Joyanes, L., & Zahonero, I. (2007). *Estructura de Datos en C++* (1.^a ed.). McGraw-Hill Education.
- ❖ Algoritmo de Dijkstra: Origen y Definición. (2021). Retrieved 27 April 2021, from <https://ingenieradeideas.com/2014/04/algoritmo-de-dijkstra-origen-y.html>
- ❖ Sclar, M. (2016). *Camino mínimo en grafos* [PDF]. Retrieved 27 April 2021, from <http://www.oia.unsam.edu.ar/wp-content/uploads/2017/11/dijkstra-prim.pdf>.
- ❖ Montes Romero, Á. (2017). *Planificación de caminos basada en modelo combinando algoritmos de búsqueda en grafo, derivados de RRT y RRT**. [PDF]. Universidad de Sevilla. Retrieved 27 April 2021, from <http://bibing.us.es/proyectos/abreproy/71064/fichero/1064-MONTES.pdf>.

