**Algoritmo de Dijkstra.**

El algoritmo de Dijktra asignará unos valores de distancia inicial y asignará paso por paso las distancia de acuerdo a como visite todos los nodos del grafo, haciendo uso de los pasos siguientes.

1.- Asignar a cada nodo un valor de distancia. Poner en cero para el nodo inicial y infinito para los otros nodos.

2.- Marcar todos los nodos como visitados

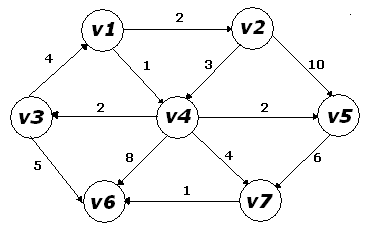
3.- Para el nodo corriente v, considere todos los vecinos no visitados [u1, u2, u3, …, un] y calcule la distancia desde el nodo inicial. Por ejemplo si el nodo corriente v tiene distancia de 6 y una arista que lo conecta con un nodo u es 2 , la distancia de u a v será 6+2 = 8. Si la distancia es menor que la distancia almacenada previamente (infinito al comienzo y cero para el nodo inicial), cambiar la distancia almacenada.

4.- Cuando todos los vecinos del nodo corriente son visitados, marcarlo como visitado.

5.- Poner el nodo no visitado con la distancia mas pequeña como en nodo corriente y repita a partir de lo paso 3.

**Ejemplo 1**

Para el siguiente grafo calcular la distancia minima del nodo v1 a todos los demás. ( ver ejemplo03 en Grafo.java)



Inicialmente la tabla de distancias queda:

|  |  |  |  |
| --- | --- | --- | --- |
| **Nodo** | **Visitado** | **Distancia** | **anterior** |
| v1 | F | 0 | - |
| v2 | F | Inf | - |
| v3 | F | Inf | - |
| v4 | F | Inf | - |
| v5 | F | Inf | - |
| v6 | F | Inf | - |
| v7 | F | Inf | - |

cola = [v1,0.0][v2, oo ][v3, oo ][v4, oo ][v5, oo ][v6, oo ][v7, oo ]

Visitamos el nodo v = v1 sucesores [v2 2.0] -> [v4 1.0]

|  |  |  |  |
| --- | --- | --- | --- |
| **Nodo** | **Visitado** | **Distancia** | **anterior** |
| v1 | V | 0 | v1 |
| v2 | V | 2 | v1 |
| v3 | F | Inf | - |
| v4 | V | 1 | v1 |
| v5 | F | Inf | - |
| v6 | F | Inf | - |
| v7 | F | Inf | - |

cola = [v4,1.0][v2,2.0][v3, oo ][v7, oo ][v5, oo ][v6, oo ]

Visitamos el nodo v = v4 sucesores [ v3 2.0]-> [v5 2.0] -> [v6 8.0] ->[v7 4.0 ]

|  |  |  |  |
| --- | --- | --- | --- |
| **Nodo** | **Visitado** | **Distancia** | **anterior** |
| v1 | V | 0 | v1 |
| v2 | F | 2 | v1 |
| v3 | F | 3 | v4 |
| v4 | V | 1 | v1 |
| v5 | F | 3 | v4 |
| v6 | F | 9 | v4 |
| v7 | F | 5 | v4 |

cola = [v2,2.0][v5,3.0][v3,3.0][v7,5.0][v6,9.0]

Visitamos el nodo v = v2 sucesores [v4 3.0]-> [v5 10.0]->

|  |  |  |  |
| --- | --- | --- | --- |
| **Nodo** | **Visitado** | **Distancia** | **anterior** |
| v1 | V | 0 | v1 |
| v2 | V | 2 | v1 |
| v3 | F | 3 | v4 |
| v4 | V | 1 | v1 |
| v5 | F | 3 | v4 |
| v6 | F | 9 | v4 |
| v7 | F | 5 | v4 |

cola = [v5,3.0][v7,5.0][v3,3.0][v6,9.0]

Visitamos el nodo v = v5 sucesores [v7 6.0]

|  |  |  |  |
| --- | --- | --- | --- |
| **Nodo** | **Visitado** | **Distancia** | **anterior** |
| v1 | V | 0 | v1 |
| v2 | V | 2 | v1 |
| v3 | F | 3 | v4 |
| v4 | V | 1 | v1 |
| v5 | V | 3 | v4 |
| v6 | F | 9 | v4 |
| v7 | F | 5 | v4 |

cola = [v3,3.0][v7,5.0][v6,9.0]

Visitamos el nodo v = v3 sucesores [v1 4.0]-> [v6 5.0]

|  |  |  |  |
| --- | --- | --- | --- |
| **Nodo** | **Visitado** | **Distancia** | **anterior** |
| v1 | V | 0 | v1 |
| v2 | V | 2 | v1 |
| v3 | V | 3 | v4 |
| v4 | V | 1 | v1 |
| v5 | V | 3 | v4 |
| v6 | F | 8 | v3 |
| v7 | F | 5 | v4 |

cola = [v7,5.0][v6,8.0]

Visitamos el nodo v = v7 sucesores [ v6 1.0 ]

|  |  |  |  |
| --- | --- | --- | --- |
| **Nodo** | **Visitado** | **Distancia** | **anterior** |
| v1 | V | 0 | v1 |
| v2 | V | 2 | v1 |
| v3 | V | 3 | v4 |
| v4 | V | 1 | v1 |
| v5 | V | 3 | v4 |
| v6 | F | 6 | v7 |
| v7 | V | 5 | v4 |

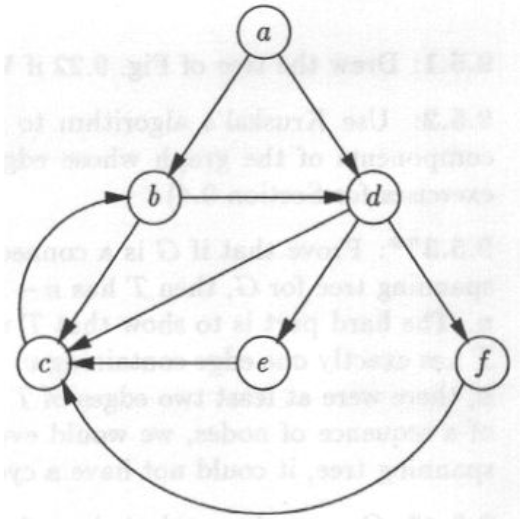
cola = [v6,6.0]

Visitamos el nodo v = v6 el cual no tiene sucesores

|  |  |  |  |
| --- | --- | --- | --- |
| **Nodo** | **Visitado** | **Distancia** | **anterior** |
| v1 | V | 0 | v1 |
| v2 | V | 2 | v1 |
| v3 | V | 3 | v4 |
| v4 | V | 1 | v1 |
| v5 | V | 3 | v4 |
| v6 | V | 6 | v7 |
| v7 | V | 5 | v4 |

**Ejemplo 2**

Para el grafo mostrado en el figura calcular la distancia minima a todos los nodos suponiendo como inicial el nodo a. (ver ejemplo01 en Grafo.java)



|  |  |  |  |
| --- | --- | --- | --- |
| **Nodo** | **Visitado** | **Distancia** | **anterior** |
| a | F | 0 | a |
| b | F | Inf | - |
| c | F | Inf | - |
| d | F | Inf | - |
| e | F | Inf | - |
| f | F | Inf | - |

cola = [a,0.0][b, oo ][c, oo ][d, oo ][e, oo ][f, oo ]

Visitamos el nodo v = a sucesores [b |1.0 ] -> [d | 1.0]

|  |  |  |  |
| --- | --- | --- | --- |
| **Nodo** | **Visitado** | **Distancia** | **anterior** |
| a | V | 0 | a |
| b | F | 1 | a |
| c | F | Inf | - |
| d | F | 1 | a |
| e | F | Inf | - |
| f | F | Inf | - |

cola = [b,1.0][d,1.0][c, oo ][f, oo ][e, oo ]

Visitamos el nodo v = b sucesores [c | 1.0] -> [d | 1.0]

|  |  |  |  |
| --- | --- | --- | --- |
| **Nodo** | **Visitado** | **Distancia** | **anterior** |
| a | V | 0 | a |
| b | V | 1 | a |
| c | F | 2 | b |
| d | F | 1 | a |
| e | F | Inf | - |
| f | F | Inf | - |

cola = [d,1.0][e, oo ][c,2.0][f, oo ]

Visitamos el nodo v = d sucesores [c | 1.0] -> [e | 1.0] -> [f | 1.0]

|  |  |  |  |
| --- | --- | --- | --- |
| **Nodo** | **Visitado** | **Distancia** | **anterior** |
| A | V | 0 | a |
| B | V | 1 | a |
| C | F | 2 | b |
| D | V | 1 | a |
| E | F | 2 | d |
| F | F | 2 | d |

cola = [c,2.0][e,2.0][f,2.0]

Visitamos el nodo v = c sucesores [b |1.0 ]

|  |  |  |  |
| --- | --- | --- | --- |
| **Nodo** | **Visitado** | **Distancia** | **anterior** |
| A | V | 0 | a |
| B | V | 1 | a |
| C | V | 2 | b |
| D | V | 1 | a |
| E | F | 2 | d |
| F | F | 2 | d |

cola = [f,2.0][e,2.0]

Visitamos el nodo v = f sucesores [c | 1.0]

|  |  |  |  |
| --- | --- | --- | --- |
| **Nodo** | **Visitado** | **Distancia** | **anterior** |
| A | V | 0 | a |
| B | V | 1 | a |
| C | V | 2 | b |
| D | V | 1 | a |
| E | F | 2 | d |
| F | V | 2 | d |

cola = [e,2.0]

Visitamos el nodo v = e sucesores [c 1]

|  |  |  |  |
| --- | --- | --- | --- |
| **Nodo** | **Visitado** | **Distancia** | **anterior** |
| A | V | 0 | a |
| B | V | 1 | a |
| C | V | 2 | b |
| D | V | 1 | a |
| E | F | 2 | d |
| F | V | 2 | d |

**Algoritmo de Prim**

El algoritmo de Prim se utiliza para calcular el árbol de expansión de un grafo. El algoritmo es

1. Dada un grafo conexo pesado con vértices V y aristas E

2. Inicialice: Vnew = {x}, donde x es un nodo arbitrario (punto inicial) de V, Enew = {}

3. Repetir mientras en la cola existan elementos Vnew = V:

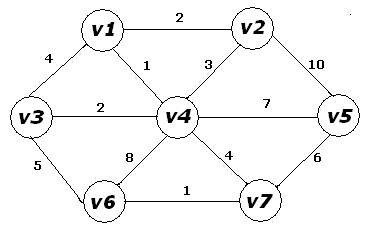
o Seleccionar una arista (u,v) con pesos inicial tal que u esta en V y tenga el menor costo y v no esta en Vnew.

o Adicionar v a Vnew y (u, v) a Enew

4. salida: Vnew y Enew que describe e árbol de expansión minima.

**Ejemplo 3**

Para el grafo mostrado en la siguiente figura calcular el árbol de expansión mínima. (ver ejemplo 4 en Grafo.java)



Árbol de expansión Mínima

Inicialmente

|  |  |  |  |
| --- | --- | --- | --- |
| **Nodo** | **Visitado** | **Distancia** | **anterior** |
| v1 | F | 0 | v1 |
| v2 | F | Inf | - |
| v3 | F | Inf | - |
| v4 | F | Inf | - |
| v5 | F | Inf | - |
| v6 | F | Inf | - |
| v7 | F | Inf | - |

cola = [v1,0.0][v2, oo ][v3, oo ][v4, oo ][v5, oo ][v6, oo ][v7, oo ]

Visitando nodo v1

|  |  |  |  |
| --- | --- | --- | --- |
| **Nodo** | **Visitado** | **Distancia** | **anterior** |
| v1 | V | 0 | v1 |
| v2 | F | 2 | v1 |
| v3 | F | 4 | v1 |
| v4 | F | 1 | v1 |
| v5 | F | Inf | - |
| v6 | F | Inf | - |
| v7 | F | Inf | - |

cola = [v4,1.0][v2,2.0][v3,4.0][v7, oo ][v5, oo ][v6, oo ]

Visitando nodo v4

|  |  |  |  |
| --- | --- | --- | --- |
| **Nodo** | **Visitado** | **Distancia** | **anterior** |
| v1 | V | 0 | v1 |
| v2 | F | 2 | v1 |
| v3 | F | 2 | v4 |
| v4 | V | 1 | v4 |
| v5 | F | 7 | v4 |
| v6 | F | 8 | v4 |
| v7 | F | 4 | v4 |

cola = [v2,2.0][v7,4.0][v3,2.0][v5,7.0][v6,8.0]

Visitando Nodo v2

|  |  |  |  |
| --- | --- | --- | --- |
| **Nodo** | **Visitado** | **Distancia** | **anterior** |
| v1 | V | 0 | v1 |
| v2 | V | 2 | v1 |
| v3 | F | 2 | v4 |
| v4 | V | 1 | v1 |
| v5 | F | 7 | v4 |
| v6 | F | 8 | v4 |
| v7 | F | 4 | v4 |

cola = [v3,2.0][v7,4.0][v6,8.0][v5,7.0]

Visitando Nodo v3

|  |  |  |  |
| --- | --- | --- | --- |
| **Nodo** | **Visitado** | **Distancia** | **anterior** |
| v1 | V | 0 | v1 |
| v2 | V | 2 | v1 |
| v3 | V | 2 | v4 |
| v4 | V | 1 | v1 |
| v5 | F | 7 | v4 |
| v6 | F | 5 | v3 |
| v7 | F | 4 | v4 |

cola = [v7,4.0][v5,7.0][v6,5.0]

Visitando el Nodo v7

|  |  |  |  |
| --- | --- | --- | --- |
| **Nodo** | **Visitado** | **Distancia** | **anterior** |
| v1 | V | 0 | v1 |
| v2 | V | 2 | v1 |
| v3 | V | 2 | v4 |
| v4 | V | 1 | v1 |
| v5 | F | 6 | v7 |
| v6 | F | 1 | v7 |
| v7 | V | 4 | v4 |

cola = [v6,1.0][v5,6.0]

Visitando el nodo v6

|  |  |  |  |
| --- | --- | --- | --- |
| **Nodo** | **Visitado** | **Distancia** | **anterior** |
| V1 | V | 0 | v1 |
| V2 | V | 2 | v1 |
| V3 | V | 2 | v4 |
| V4 | V | 1 | v1 |
| V5 | F | 6 | v7 |
| V6 | V | 1 | v7 |
| V7 | V | 4 | v4 |

cola = [v5,6.0]

Finalmente visitando el nodo v5

|  |  |  |  |
| --- | --- | --- | --- |
| **Nodo** | **Visitado** | **Distancia** | **anterior** |
| v1 | V | 0 | v1 |
| v2 | V | 2 | v1 |
| v3 | V | 2 | v4 |
| v4 | V | 1 | v1 |
| v5 | V | 6 | v7 |
| v6 | V | 1 | v7 |
| v7 | V | 4 | v4 |