Universidad Nacional de San Martín

Licenciatura en Ciencia de Datos

Algoritmos II

Introducción Bellman Ford

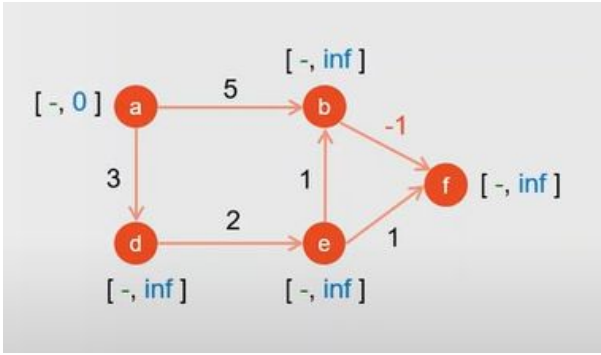
-También devuelve el **camino de menor costo** desde un **nodo origen** a cualquiera que **tenga conexión.**

-SOPORTA ARISTAS CON PESOS NEGATIVOS!!

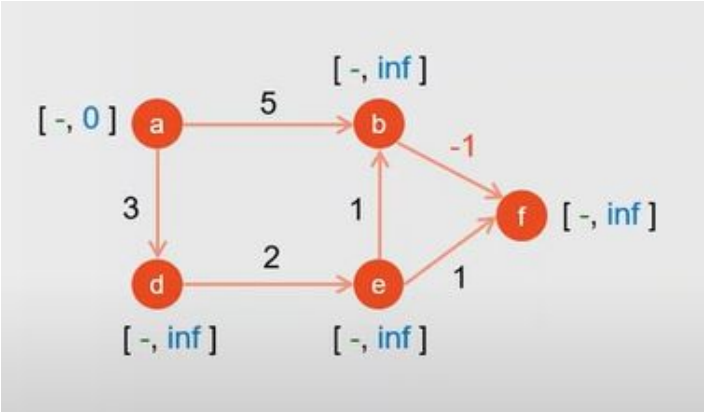
-DETECTA CICLOS NEGATIVOS!

-Se incorporan a las etiquetas [nodo predecesor, costo mínimo desde nodo origen]

-Diferencia Dijkstra: no tenemos info de qué nodos visitamos porque **<u>no vamos a estar marcando nodos</u>** de visita.



Algoritmo



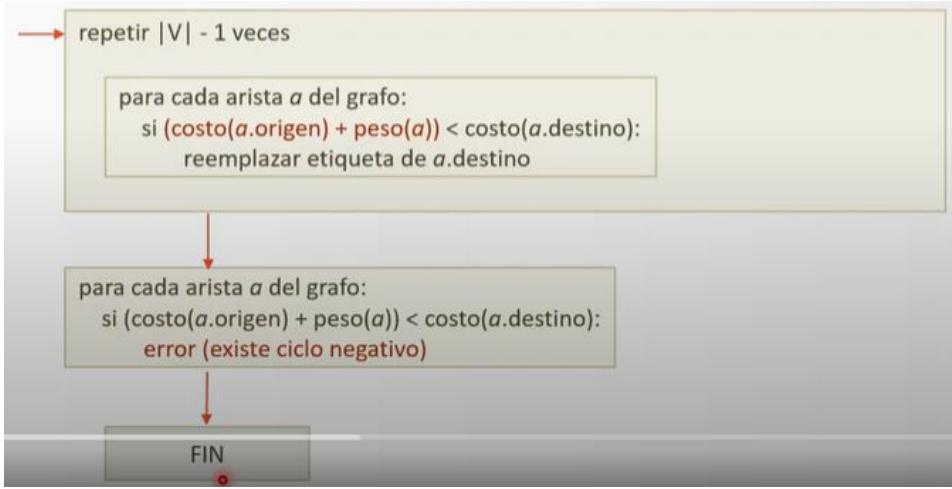
Situación inicial:

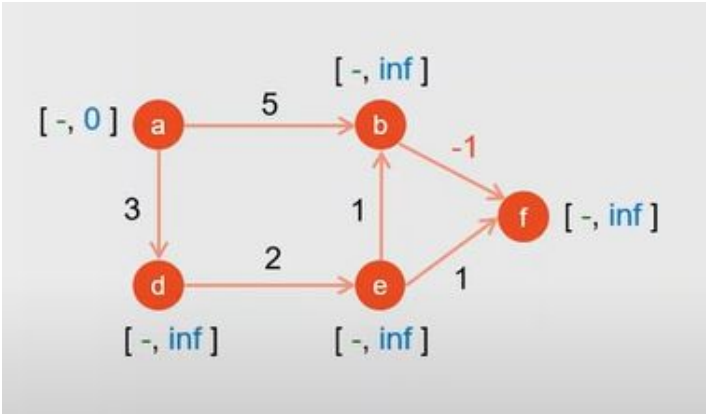
-**Costo mínimo** = inf
-**Predecesores** = vacío
-Costo **nodo inicial** =0

Algoritmo

-Va a hacer una **repetición de V-1** veces un recorrido del cálculo y el análisis de cada **arista**.

-Se recorre **cada ARISTA** del grafo, **V-1 veces**, y en c/u se calcula la condición: si el **costo del origen de la arista + el peso de la arista** es **menor** al costo del nodo destino (predecesor, origen de arista), se **reemplaza la etiqueta**
-Luego, el caso es el mismo que el del **"repetir"**, sólo que **se hace una vez más la validación de cambio de etiqueta**, para ver si **existe un CICLO NEGATIVO**

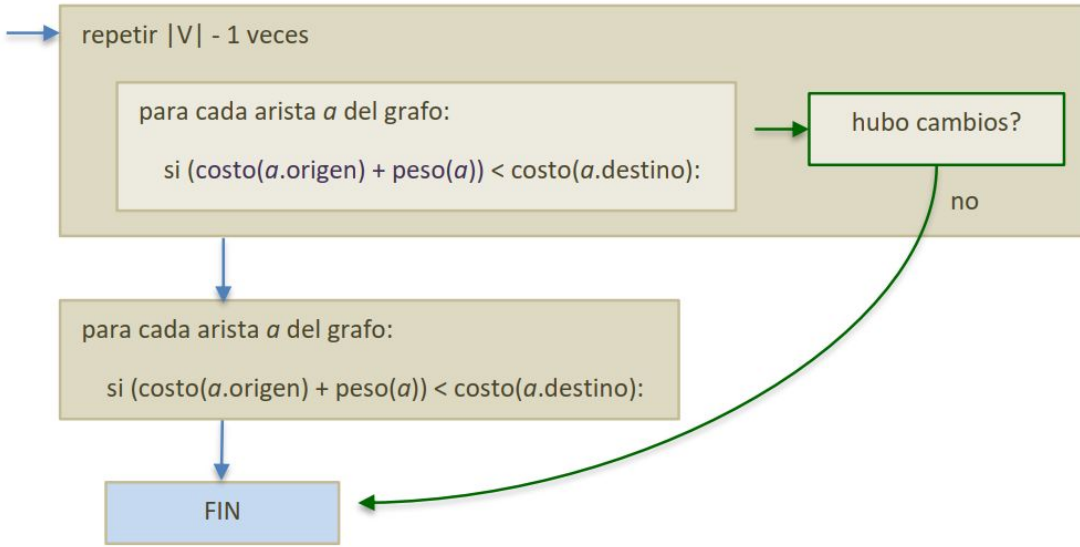




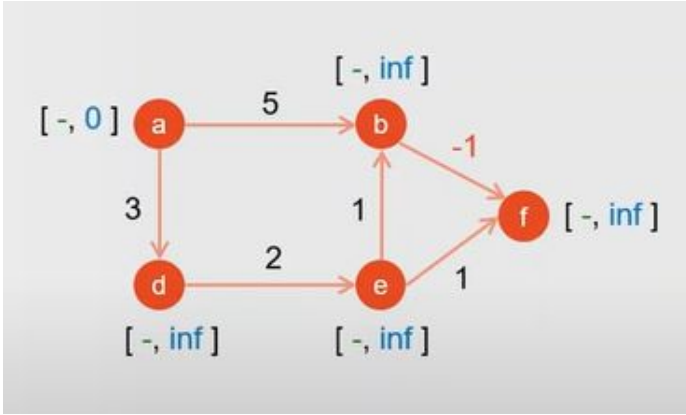
Algoritmo: modificación

-Optimización: si no hubo cambios en el **tenemos info de qué nodos visitamos**, **terminar.**

-No va al segundo cambio porque **se terminó antes del V-1 veces**. **Esto garantiza que no hubo CICLO NEGATIVO.**



Ejemplo



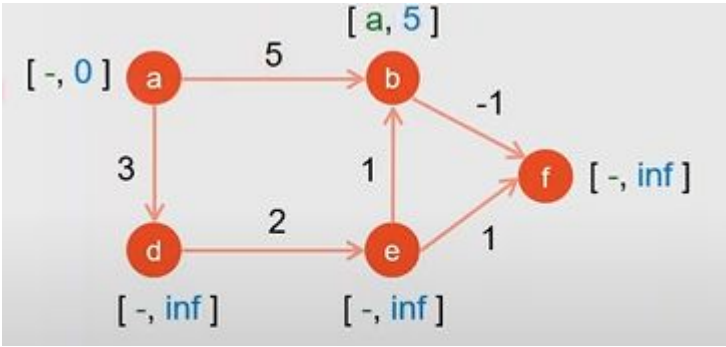
Comenzamos con el repetir en la primera iteración:

Iteración: 1

Vamos a recorrer el conj de aristas: **(a,b), (a,d), (d,e), (e,b) , (e,f), (b,f)** para **actualizar etiquetas**.

Nos posicionamos en la primera arista : **(a,b)**.

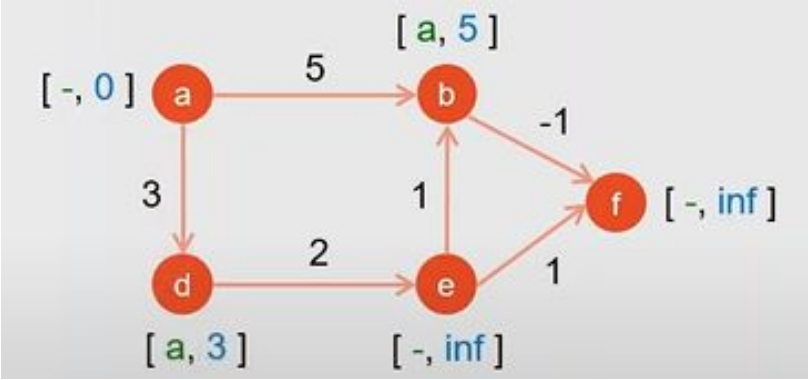
-Costo a (0)+ peso etiqueta = 5, es menor que costo de nodo destino => **reemplazo**



Ejemplo

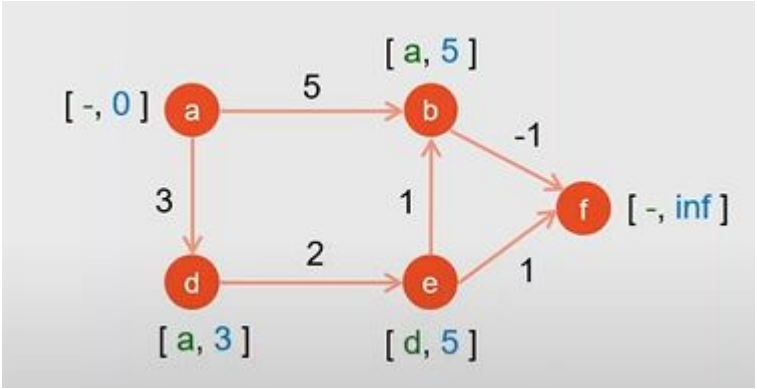
-Seguimos dentro del **para** y nos movemos a arista **(a,d)**

-Pasa lo mismo que en el caso anterior => **reemplazo**.



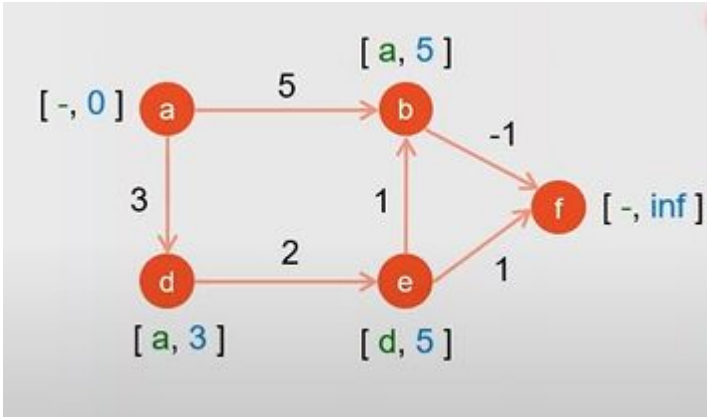
Ejemplo

-ídem siguiente arista **(d,e)**



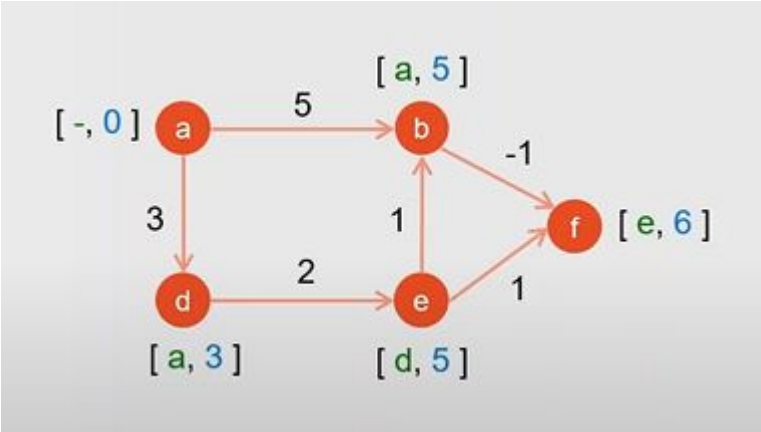
Ejemplo

-Se pasa a **(e,b)**: 5+1 = 6 < 5 => no modificamos.



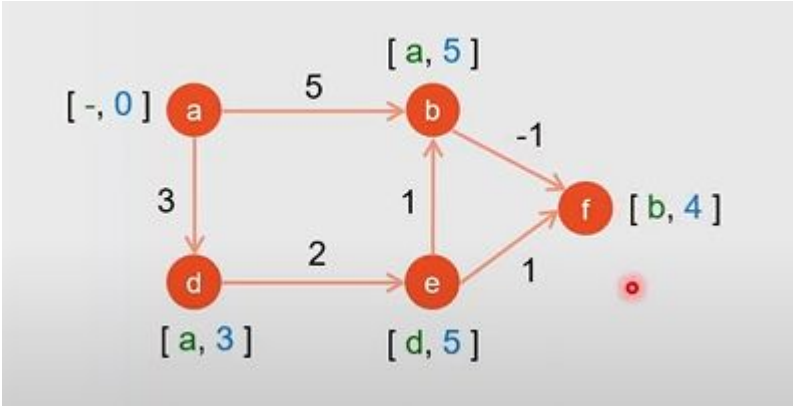
Ejemplo

-Pasamos a **(e,f)**. Reemplazo



Ejemplo

Última arista: **(b,f):** 5-1=4 => reemplazo.

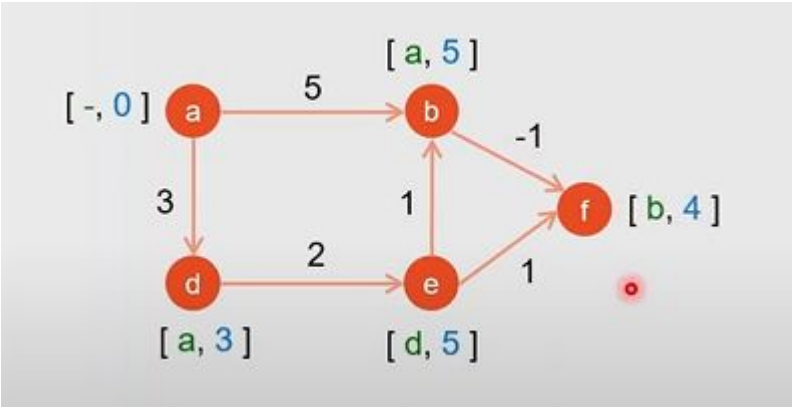


Ejemplo

Como terminó en el **“para” completo, volvemos al “repetir” V-1 veces**. Tenemos 5 nodos.

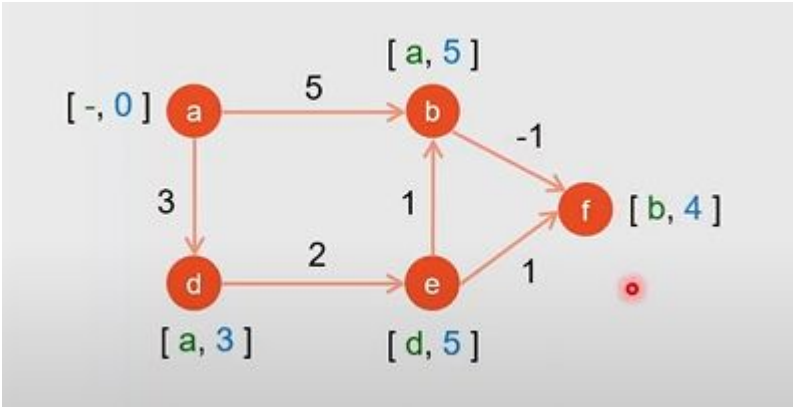
-Iteración 2:

-Verificamos **(a,b)**, no hay cambios. No modificamos etiqueta de **b**



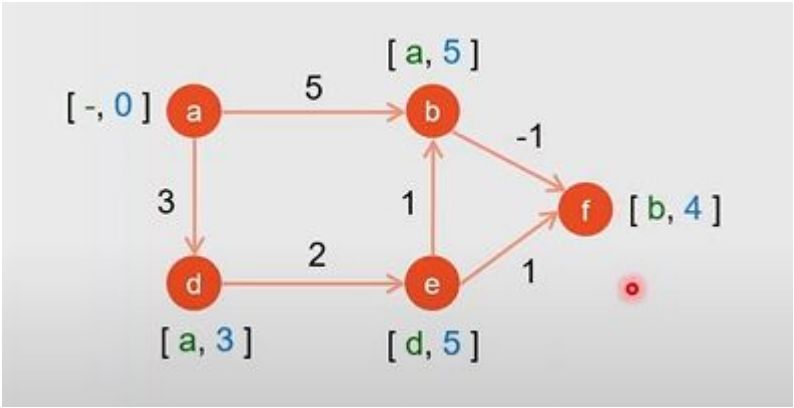
Ejemplo

-**(a,d):** ídem, no hay cambios.



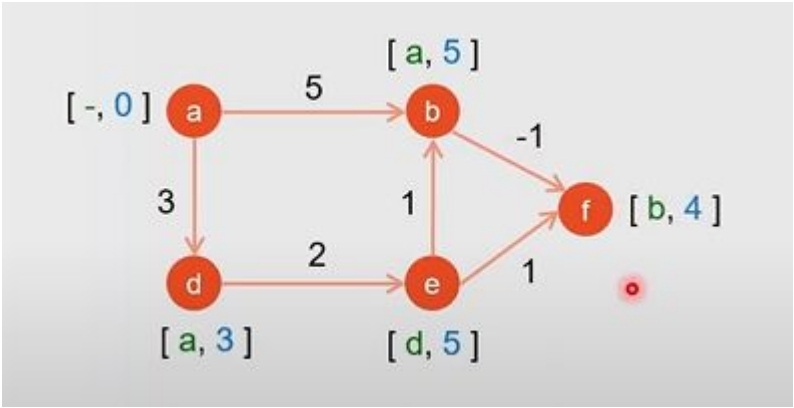
Ejemplo

(d,e): tampoco se modifica



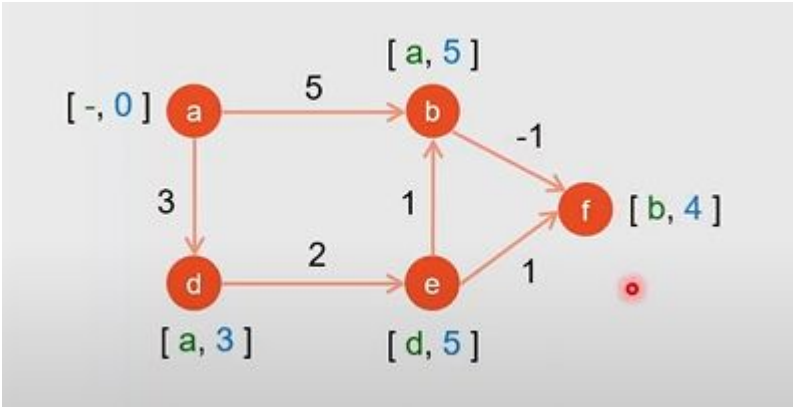
Ejemplo

(e,b):5+1=6>5. No se modifica



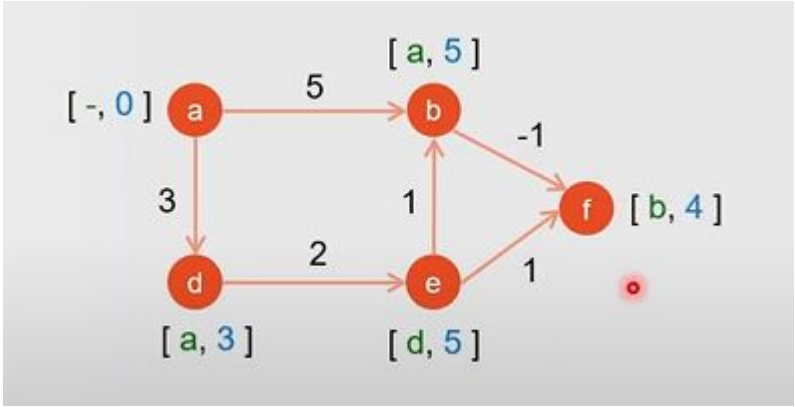
Ejemplo

(e,f): 5+1=6 > 4 no se modifica



Ejemplo

(b,f): 5-1 =4 = 4 => tampoco se modifica. => **LLEGAMOS A LA CONVERGENCIA (no hubo cambios) => FIN.**



Camino mínimo (a,f) = [a,b,f]

Costo mínimo = 4 (el costo de **etiqueta del nodo destino**).

También se construye posicionándose en el **nodo destino y siguiendo los predecesores**.

Ejercicio

Implementar el **algoritmo de Bellman Ford recursivo adaptando** la representación de grafos como conjunto de grafos y aristas (desarrollar sólo las funciones que sean necesarias únicamente). Debe recibir **dos nodos** y devolver una **lista con el camino mínimo.**

```
def __init__(self):
        self.nodos: set[T] = set()
        self.aristas: dict[tuple[T, T], int] = {}
```

```
 def bellman_ford(self, inicio: T, destino: T) -> list[T]:
```

Nota: modificar el método **agregar_arista** para que soporte la asignación de un peso.