Universidad Nacional de San Martín

Licenciatura en Ciencia de Datos

Algoritmos II

Concepto de camino mínimo

-Buscar el **camino mínimo de menor costo** partiendo desde un nodo **origen** a uno **destino (grafo ponderado).**

-Es asignado a las **aristas.**

-El **costo asignado** representa una abstracción de un **problema a resolver.** Ej: económico, tiempo, navegación (cuánto tarda una persona en llegar de una ciudad a otra), combustible consumido, etc.

-Puede ser **negativo**, por ejemplo, para representar

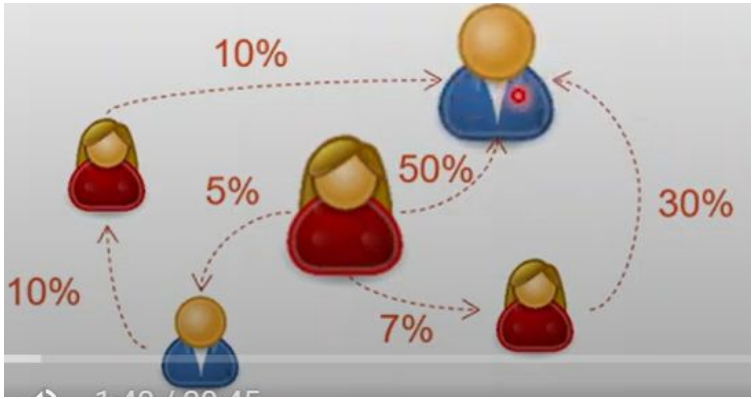
ganancias.

-El **ejemplo** del diagrama representa **personas** que

quieren **llevar un paquete a otra**, y el **porcentaje**

representa un **costo económico** (intereses), desde la persona **“**

origen” hasta otra persona **“destino”** (roja a azul)



EL CAMINO MÍNIMO NO SIEMPRE ES EL DE MENOR

ESCALA O INTERMEDIARIOS

Otro ejemplo del concepto con pesos negativos

-El grafo representa un **mapa turístico** donde los nodos son **ciudades y pueblos**, queremos llegar a otro de forma más eficiente desde el **punto de vista económico**. El costo de las aristas podría ser el **costo de un pasaje** de un medio de transporte, las **aristas naranjas** son más **dañinas** al medio ambiente y las **verdes** más **ecológicas**. Para fomentarlas, podemos asignarles **pesos negativos** (el gobierno podría estar **pagándole** a la persona para que lo use).

Continuación de pesos negativos

CUIDADO EN ESTOS CASOS :QUE NO SE DÉ UN CICLO NEGATIVO!!!! (BUCLE INFINITO) -> siempre se reduce el costo

Algoritmos: 1) Dijkstra

-Devuelve el camino de menor costo desde un **nodo origen** a cualquiera de los nodos con el que tiene **conexión.**

-NO SOPORTA PESOS NEGATIVOS!!!

-Vale para grafos **dirigidos y no dirigidos.**

-Se incorporan **etiquetas** a los nodos:

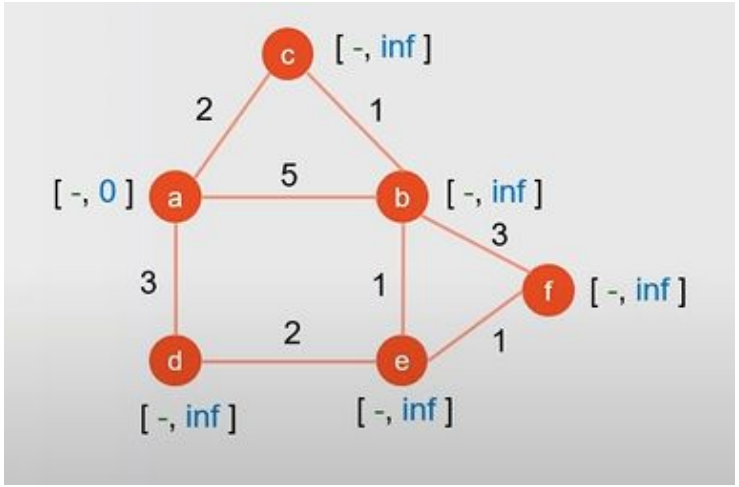
- **[nodo predecesor, costo mínimo desde nodo origen]**

Ej nodo **c**, **corchetes**:

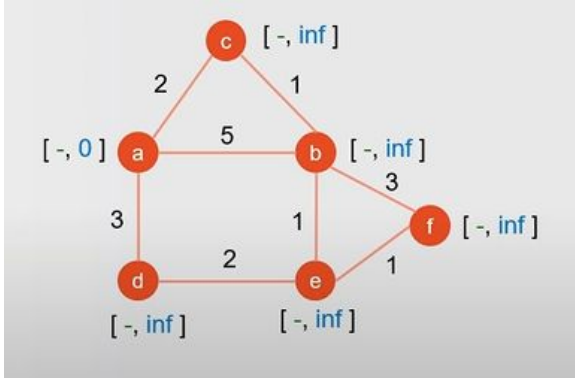
predecesor (nodo desde el cual llego),

costo ínimo asociado.

-Marca de **nodo visitado** (para no hacer **bucles**)



Situación inicial: trabajamos con todos los nodos



-No pasamos un nodo origen y destino sino **sólo nodo inicial.**

-El algoritmo **calcula el camino mínimo para todos los nodos restantes.**

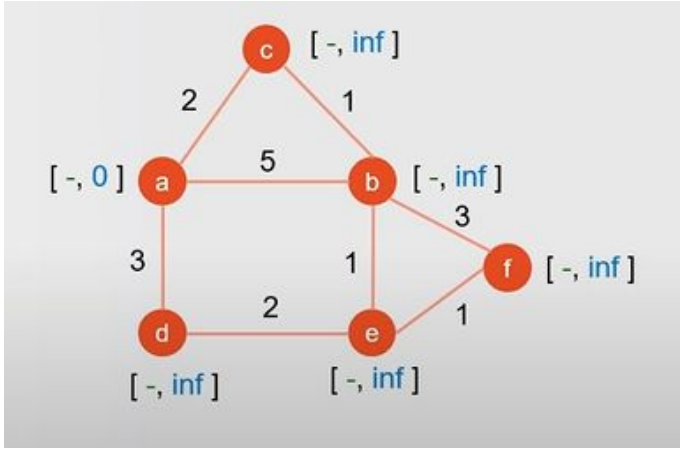
-Costo mínimo de nodos (2da parte de etiqueta) = **inf** (o máx TDD)

-Predecesores pueden estar apuntados a **nada o a sí mismos** (1era parte etiqueta). Inicialmente **No nos importa tener el valor.**

-El costo del **nodo inicial** es 0 (ver **nodo a**), porque si quiero recorrer de **a** a **a no tengo un costo.** Estamos posicionados ahí.

-Todo el resto de los nodos **están sin marcar.**

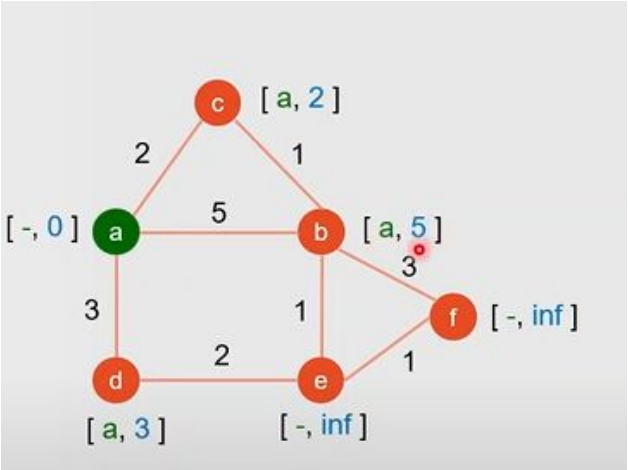
¿Qué vamos a hacer?



-Se le pasa un nodo **inicial** y un nodo **destino**.
-**Buscar mínimo: etiqueta con menor valor**. En situación **inicial** están **todos con inf** excepto **inicial** => se **empieza** por este.
-Los **predecesores** están **vacíos, no nos importa el valor**
-Lo **marcamos y le asignamos 0 a su costo.**
- Para **cada vecino del nodo inicial** comparamos si el **costo** de ese **nodo** (2do componente de etiqueta) **+ peso de arista** (ej: de **a** a **c** 0+2) es **menor a la etiqueta del destino**, y si es así **reemplazamos la etiqueta**.
-Se **repite para cada vecino** y **volvemos al primer paso**

Ejemplo

-Calculamos **pesos y predecesores** del nodo a, **elegimos el c por menor costo** y le asignamos al costo como **predecesor a** y el **costo 0 de a + 2**. También le asignamos el a como predecesor y costo a **sus vecinos**, **b y d**.

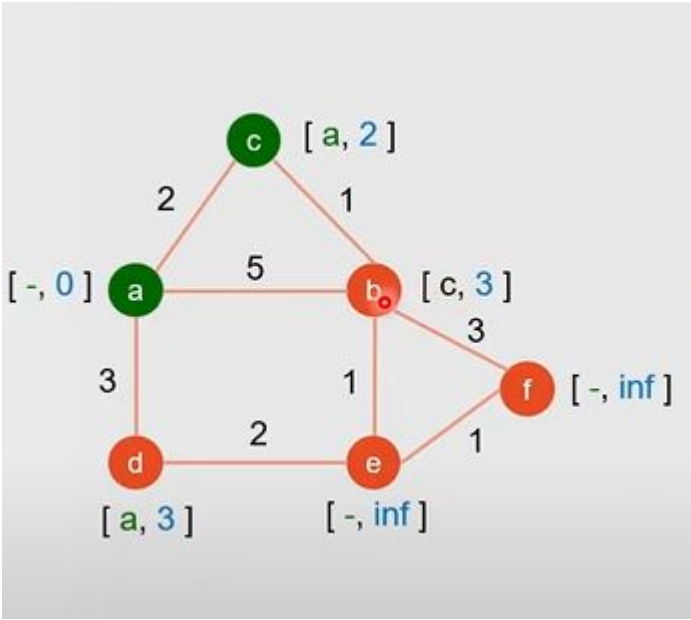


-Luego vemos que el **nodo a** ya está **marcado**,entonces no vamos a volver a visitarlo.

Ejemplo

Volvemos al **“sin marcar”** y buscamos el **próximo que tenga peso mínimo,** que es el **c,** y lo **marcamos.**

-Sus vecinos son **a y b**, pero **a ya está marcado no lo vamos a usar**



-Vamos al **b**. Calculamos el costo, que es 3 (2+1), y es menor a 5 => reemplazamos la etiqueta de **b.** Ahora el costo es **3** pero tengo que ir desde **c**

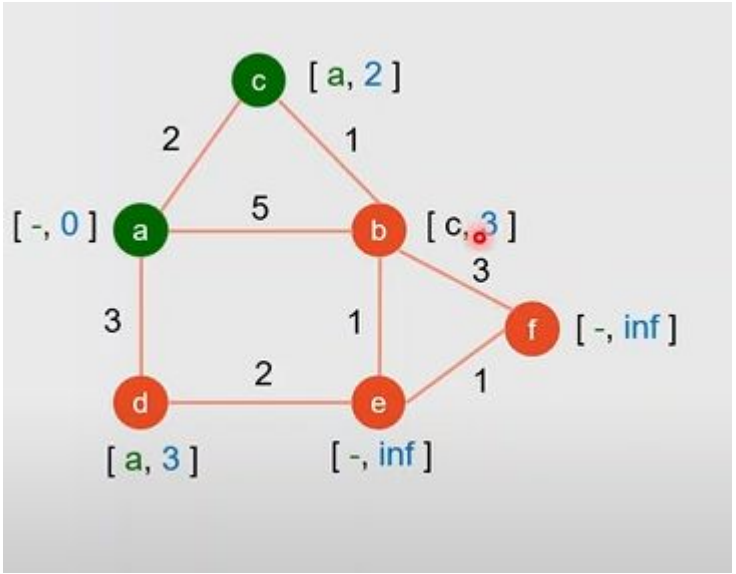
Ejemplo

-Volvemos al **sin marcar,** porque **c ya tiene todos sus vecinos visitados**.
Quedan cuatro sin marcar y hay que elegir el **mínimo.**

Podemos elegir **indistintamente**

entre el **b** y el **d** porque tienen

costo mínimo 3.

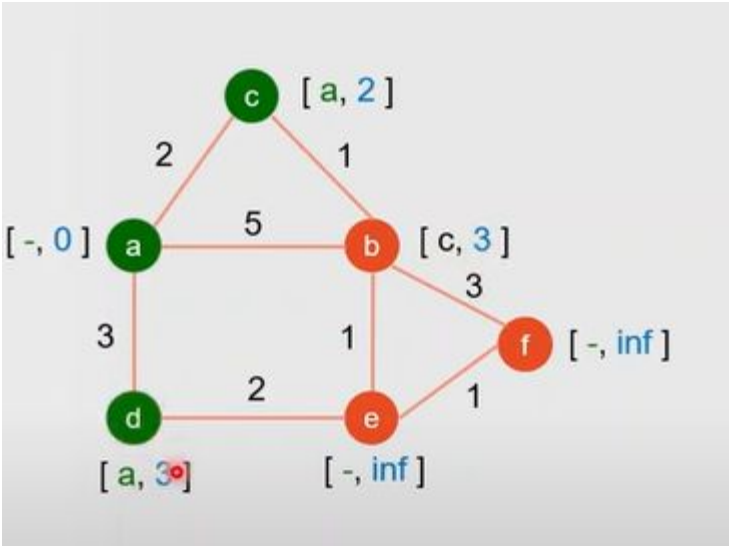


Ejemplo

-Tomamos el **d** de **manera trivial**

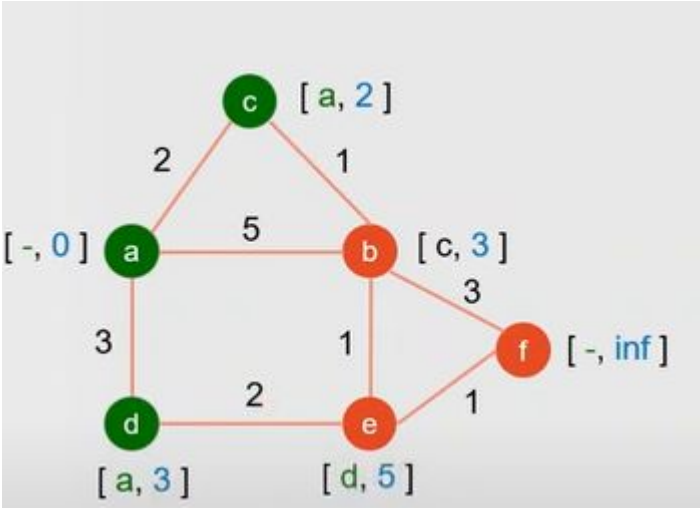
-Vemos cuáles son sus vecinos: **a y e**.

-El **a ya está marcado**, vamos al **e**



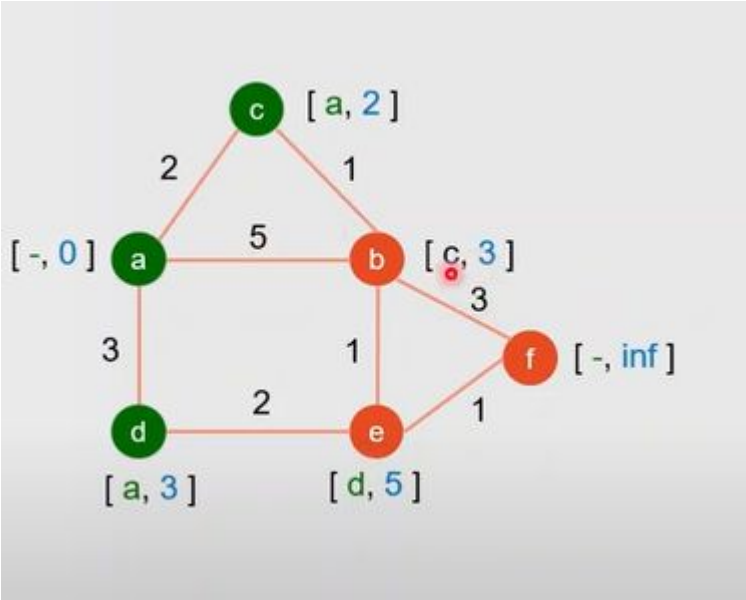
Ejemplo

-Se reemplaza la etiqueta del **e.**



Ejemplo

-Ya **no quedan vecinos** por recorrer de **d**, entonces nuevamente volvemos a ver el **mínimo** de los que hay **sin marcar**, que es el **b.**

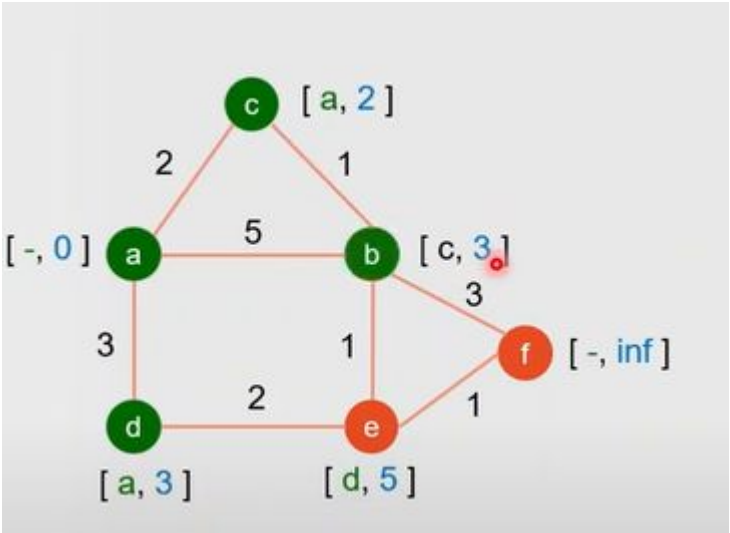


Ejemplo

-Desde el **b, que lo marcamos:**

-Tiene los vecinos **a** y **c** que no se van a modificar, **ya fueron recorridos** y tienen **etiquetas menores**.

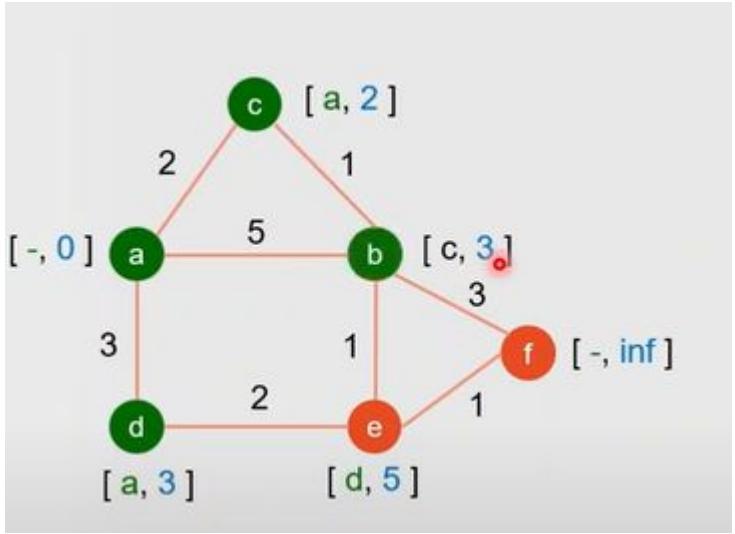
-Nos quedan los vecinos **f** y **e.**

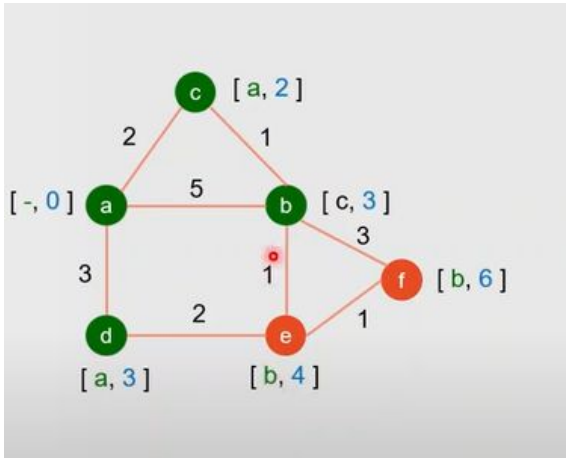


Ejemplo

-**e** tiene costo **5, PERO** desde **b** el costo de llegar es 3+1= **4**, **lo reemplazamos**

-Y el mismo cálculo para **f**



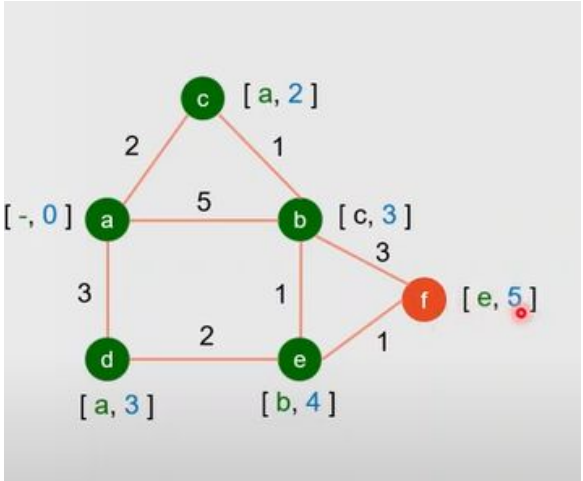


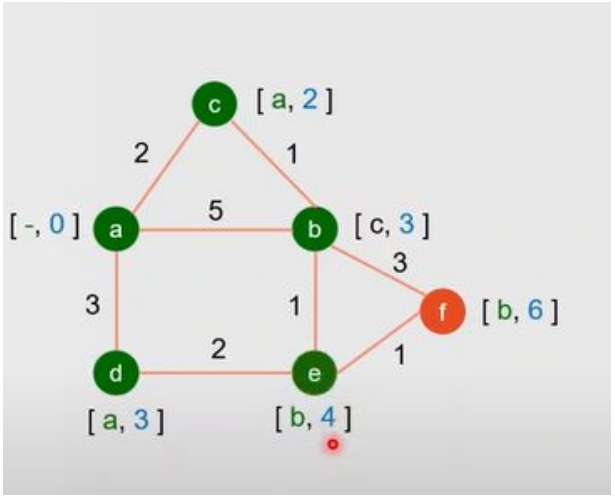
Ejemplo

-Ya **b** no tiene vecinos, y nos quedan **e** y **f** por ver. Empezamos por el **e** porque es el **mínimo.**

-Sus vecinos **b** y **d** los descartamos, así que

evaluamos respecto al **f: 4+1=5 < 6 => reemplazo.**



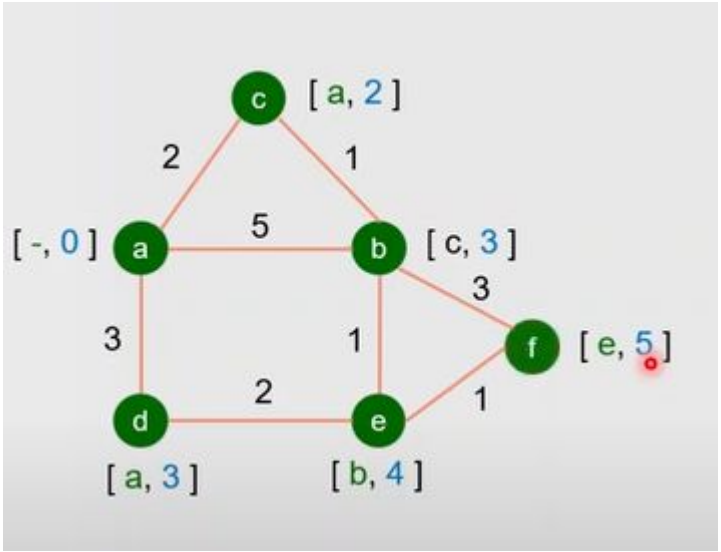


Ejemplo

-Nos queda revisar el **f:**

Tiene sus vecinos ya marcados y todos los nodos del grafo están marcados.

=> finaliza el algoritmo.



Ejemplo

-El **camino mínimo,** por ejemplo, desde **a** a **f:** se empieza **DESDE EL DESTINO.**

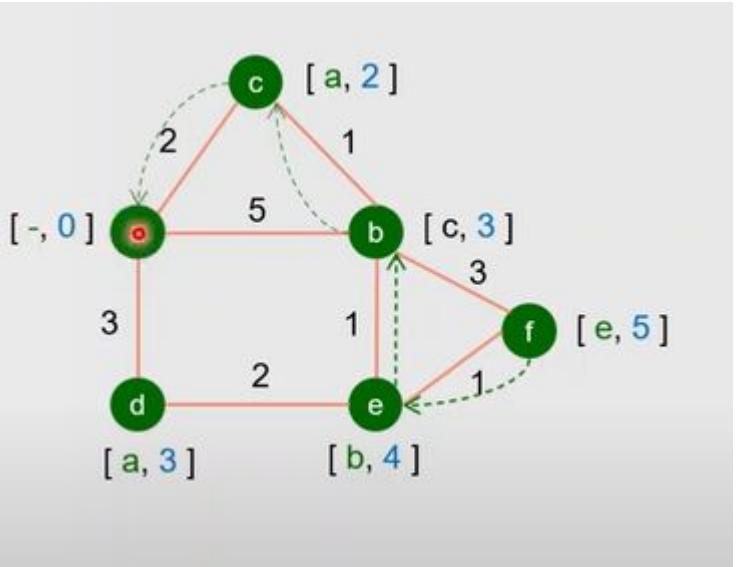
Predecesor: e (=> vamos al **e**).

-**e** tiene de predecesor a **b**, y a su vez al **c** desde el **b.**

-c tiene como predecesor óptimo el **a,** por ende tenemos el

camino mínimo: a-c-b-e-f (inverso).

costo mínimo = 5



Ejercicio

Implementar el **algoritmo de Dijkstra recursivo** adaptando la representación de grafos como **conjunto de grafos y aristas**. Debe **recibir dos nodos y devolver una lista con el camino mínimo.**

Firma:

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