Video 43 (1)

Conceptos Principales

1. a
2. b
3. c

Notas

Resumen

Un **árbol de expansión mínima (MST)** es básicamente agarrar un grafo no dirigido y con pesos, y quedarte solo con las aristas necesarias para que **todos los vértices sigan conectados**, pero **con el costo total más bajo posible**. Quitás todo lo que sea "extra".

Esto se usa muchísimo en cosas como redes de comunicación, cableado, logística, conexiones entre ciudades, etc., porque literalmente te da la forma más barata de conectar todo.

En este video se trabaja con el algoritmo de **Prim**, que es uno de los clásicos para construir un MST.

¿Qué problema resuelve Prim?

Imaginá que tenés un montón de puntos conectados entre sí con distintos costos (caminos, cables, distancias). Vos querés conectar **todos** esos puntos pero gastando **lo menos posible**.

Prim sirve justo para eso: conectar todo sin hacer ciclos y usando las aristas más baratas posibles.

Cómo funciona Prim (explicado simple)

Prim empieza **desde cualquier vértice** —da igual cuál— y desde ahí va construyendo el MST paso a paso. La mecánica es siempre esta:

1. **Elegís un vértice inicial.**
2. Ponés en una tabla todas las aristas que salen de ese vértice.

   Cada entrada indica: *de dónde viene*, *a dónde va* y *cuánto cuesta*.
3. **Escogés la arista más barata** de la tabla que conecte con un vértice nuevo (uno que aún no esté en el MST).

   Si la arista conecta dos vértices que ya están dentro, la saltás, porque eso haría un ciclo.
4. Metés ese vértice al conjunto de "visitados" o "ya en el MST".

5. Agregás a la tabla **todas las aristas que salen del vértice recién agregado**, siempre que lleven a vértices no visitados.
6. Repetís hasta que **todos los vértices estén dentro**.

El MST final tendrá **n-1 aristas**, si el grafo tiene **n vértices**.

¿Qué estructuras usa?

Prim necesita:

1. **Una tabla/lista** donde ir guardando las aristas candidatas (las que salen desde el conjunto actual).

   Puede ser una lista normal, una matriz, o en implementaciones modernas hasta un heap (pero aquí no se usa heap).

2. **Un conjunto de visitados** para saber qué vértices ya están dentro del MST y evitar ciclos.
3. **Una estructura de resultados**, que básicamente son las aristas que se van eligiendo.

El video lo hace usando una tabla manual donde va anotando cada arista y su peso.

Idea clave para examen

Prim siempre funciona así:

Conjunto de vértices ya conectados + elegir la arista más barata que agregue un vértice nuevo.

Si te equivocás en algo, suele ser en:

- olvidar marcar un vértice como visitado,
- meter una arista que hace ciclo,
- o no actualizar bien la tabla cuando agregás adyacentes nuevos.

Ejemplo explicado sencillo (sin repetir todo el video)

Supongamos que empezás en A.

A conecta con D (4), K (2) y M (2).

El valor más bajo es 2, entonces te llevás A–K o A–M (cualquiera si empatan).

Luego, desde K agregás sus adyacentes… y así vas avanzando.

La lógica nunca cambia: siempre el **menor peso disponible** que conecte algo nuevo.

Por qué esto importa en programación en C

Cuando implementás Prim en C:

- Tenés que manejar bien las **matrices de adyacencia** o listas de aristas.

- Tenés que manejar **tablas auxiliares** (estructuras de datos).
- Tenés que controlar ciclos y visitados.
- Te entrena para trabajar con **grafos**, arreglos, ciclos anidados y decisiones de mínimo.

Prim es un algoritmo muy común en tareas y exámenes de estructuras de datos, sobre todo en temas de grafos.