

# Kruskal — Árbol de Expansión Mínima (MST)

Manual técnico y guía de uso en GraphStar

## 1. Descripción General del Algoritmo

Kruskal es un algoritmo codicioso (greedy) que encuentra un Árbol de Expansión Mínima en grafos no dirigidos con pesos en aristas. Su principio es simple: ordena todas las aristas por peso en orden ascendente y las añade una por una al árbol si no crean un ciclo.

### Características Principales

- Grafo soportado:** No dirigido, conexo (si es desconexo, GraphStar lo detecta y muestra error).
- Complejidad temporal:**  $O(E \log E)$  donde  $E$  es el número de aristas (dominado por la ordenación).
- Salida:**  $V-1$  aristas (donde  $V$  es el número de nodos) que forman el MST.
- Costo total:** Suma de pesos de las aristas seleccionadas.

**Nota importante:** Kruskal solo funciona con grafos *no dirigidos*. Si intentas aplicarlo en un grafo dirigido, GraphStar mostrará una advertencia.

## 2. Implementación en GraphStar

### Estructura de Archivos

Archivo	Descripción
<code>src/algorithms/KruskalAlgorithm.js</code>	Módulo puro que implementa el algoritmo Kruskal con DSU.
<code>src/components/KruskalGraphStar.jsx</code>	Componente React que proporciona la interfaz de usuario.
<code>src/App.js</code>	Enrutador que maneja la vista 'kruskal'.

### Algoritmo de Union-Find (DSU)

El corazón de Kruskal es la estructura de datos `Disjoint Set Union (DSU)` o `Union-Find`, que permite detectar ciclos eficientemente.

```

class DSU {
  constructor(size) {
    this.parent = Array.from({ length: size }, (_, i) => i);
    this.rank = Array(size).fill(0);
  }

  find(x) {
    if (this.parent[x] !== x)
      this.parent[x] = this.find(this.parent[x]); // Compresión de rutas
    return this.parent[x];
  }

  union(x, y) {
    const rx = this.find(x);
    const ry = this.find(y);
    if (rx === ry) return false; // Ya están conectados (ciclo)

    // Unión por rango
    if (this.rank[rx] < this.rank[ry]) this.parent[rx] = ry;
    else if (this.rank[rx] > this.rank[ry]) this.parent[ry] = rx;
    else {
      this.parent[ry] = rx;
      this.rank[rx]++;
    }
    return true; // Aceptar arista
  }
}

```

**Compresión de rutas:** Al hacer find(x), comprimimos la ruta hacia el raíz para futuras búsquedas más rápidas.

**Unión por rango:** Al hacer union, conectamos el árbol más pequeño al más grande para mantener árboles balanceados.

## Determinismo con Tie-breaker

Para reproducibilidad, cuando dos aristas tienen el *mismo peso*, Kruskal las ordena por `originalIndex` (el orden en que fueron creadas en el grafo).

```





prepared.sort((a, b) => {
  if (a.weight !== b.weight) return a.weight - b.weight;
  return a.originalIndex - b.originalIndex; // Tie-breaker determinista
});

```

**Implicación:** Si ejecutas el mismo grafo varias veces, obtendrás *siempre* el mismo MST (mismas aristas), garantizando reproducibilidad.

## 3. Interfaz de Usuario en GraphStar

### Pasos para usar Kruskal

- 1. **Crear nodos:** Presiona  Crear Nodo para añadir vértices.
- 2. **Crear aristas:** Presiona  Agregar Arista y conecta dos nodos. Un modal te pedirá el peso.
- 3. **Desactivar "Grafo Dirigido":** Asegúrate de que el checkbox esté desmarcado.
- 4. **Aplicar Kruskal:** Presiona  Aplicar Kruskal.
- 5. **Ver resultados:** El panel lateral muestra:
  -  MST calculado - Costo total: X.XX
  - Tabla de iteraciones con paso, arista, peso y acción (Agregada / Rechazada).

## Resaltado Visual

Las aristas del MST se resaltan con un color **magenta brillante (#ff2dff)** para que sean claramente visibles:

- **Grosor:** 6px (más grueso que las aristas normales).
- **Efecto glow:** Sombra luminosa alrededor del trazo.
- **Animación dash:** Línea punteada que se mueve constantemente.
- **Orden de renderizado:** Las aristas resaltadas se dibujan *por encima* de las demás.

Si una arista magenta es difícil de ver, intenta "Limpiar Resultado" y vuelve a aplicar Kruskal. La lista de IDs de aristas resaltadas aparece también en la consola del navegador (Abrir DevTools: F12).

## 4. Tabla de Iteraciones Explicada

Columna	Significado
Paso	Número secuencial de inspección (1, 2, 3, ..., E).
Arista	Identificador de la arista: "nodoOrigen → nodoDestino".
Peso	Peso (costo) de la arista.
Acción	No forma ciclo (añadida al MST) o Forma un ciclo (rechazada).

**Criterio de parada:** El algoritmo se detiene cuando ha seleccionado exactamente V-1 aristas (donde V es el número de nodos). Para un grafo de 9 nodos, espera 8 aristas en el MST.

## 5. Casos de Error

- **"El grafo no es conexo"**: Significa que hay nodos aislados o grupos desconectados. Kruskal no puede crear un MST que cubra todos los nodos.
- **"Algoritmo de Kruskal requiere un grafo NO dirigido"**: Has marcado "Grafo Dirigido". Desactívalo.
- **"Se necesitan al menos 2 nodos"**: Añade más nodos.
- **"Se necesitan aristas para calcular un MST"**: Conecta los nodos con aristas ponderadas.

## 6. Ejemplo Práctico

**Grafo de ejemplo:** 9 nodos (A–I) conectados con 12 aristas ponderadas.

**Resultado esperado:** MST con 8 aristas y costo total 37.00.

**Las 8 aristas seleccionadas:** dependerán del orden original si hay empates de peso (por eso el tie-breaker es importante).

Para verificar que tu resultado es correcto, cuenta las aristas resaltadas en la tabla:

- Debe haber exactamente 8 aristas con acción "No forma ciclo".
- El costo total debe ser la suma de esos 8 pesos.

### GraphStar — Manual de Kruskal

Última actualización: 10 de noviembre de 2025

Para más información, consulta la documentación en línea o abre la Guía Interactiva desde la interfaz.