Actividad 1: Selección de algoritmo según el contexto

Tarea

Generar 5 escenarios del mundo real donde se requiera construir un MST y comparar el uso de los algoritmos de Prim y Kruskal.

Tabla Comparativa de Escenarios

Escenario	Algoritmo Recomendado	Justificación	Estructuras de Datos	Escalabilidad (10x nodos)
1. Red de Fibra Óptica Urbana Conectar todos los edificios de un centro denso donde casi cualquier par de edificios tiene una línea de vista o conducto posible (Grafo Denso).	**Prim**	En grafos densos donde el número de aristas E se acerca a V^2, Prim con matriz de adyacencia o Heap es más eficiente ($O(E + V\log V)$ o $O(V^2)$) que ordenar todas las aristas como requiere Kruskal ($O(E\log E)$).	**Matriz de Adyacencia** + **Array de distancias mínimas** (o Priority Queue para grafos muy grandes).	Al crecer 10x en nodos, las aristas crecerían ~100x. Prim escala mejor con la densidad cuadrática que Kruskal, cuyo paso de ordenamiento se volvería el cuello de botella.

2. Red de Distribución Eléctrica Rural Conectar comunidades lejanas donde las conexiones posibles son limitadas y específicas (solo caminos existentes). Grafo Disperso.	**Kruskal**	En grafos dispersos ($E \approx V$), el paso de ordenar las aristas es rápido. Kruskal es conceptualmente simple y eficiente para grafos con pocas aristas.	**Lista de Aristas** + **Union-Find (DSU)** para gestión de conjuntos disjuntos.	Si los nodos crecen 10x y el grafo se mantiene disperso (grado promedio constante), E también crece ~10x. Kruskal se mantiene muy eficiente ($O(E\log\ E)$).
3. Diseño de Circuitos VLSI Conectar pines en un chip con la mínima longitud de alambre. Millones de nodos, muy pocas conexiones permitidas por restricciones físicas.	**Kruskal**	Ideal para grafos extremadamente dispersos. Además, Kruskal puede trabajar bien si las aristas ya vienen pre-ordenadas o si se generan dinámicamente.	**Union-Find** con compresión de caminos y unión por rango.	Escala lineal-logarítmica. La estructura Union-Find es casi lineal ($O(E\alpha(V))$), lo que es crucial para millones de nodos.
4. Red de Sensores Inalámbricos (Ad-hoc) Sensores que se despliegan aleatoriamente y necesitan formar una red conectada para transmitir datos al nodo base.	**Prim**	Prim puede crecer el árbol desde el nodo base (raíz) hacia afuera. Esto es útil para asegurar que la red se construye conectada al sumidero de datos desde el principio.	**Lista de Adyacencia** + **Binary Heap (Min-Heap)**.	Al aumentar nodos, la gestión del Heap es eficiente ($O(\log\ V)$ por operación). Si la densidad aumenta, Prim sigue siendo robusto.

5. Segmentación de Imágenes (Clustering) Agrupar píxeles similares. Se construye un MST y se cortan las aristas más costosas (diferencia de color) para separar regiones.	**Kruskal**	Kruskal construye el árbol uniendo componentes (bosque). Para clustering, podemos detener el algoritmo antes de que sea un único árbol (cuando queden k componentes), obteniendo los clústeres directamente.	**Lista de Aristas** + **Union-Find**.	Si la imagen es 10x más grande, Kruskal permite detenerse temprano o manejar la estructura de bosque naturalmente, lo cual es una ventaja funcional sobre Prim.