

- 
1. Explique cual es la complejidad temporal de los algoritmos vistos en clase: **BFS, Dijkstra(Priority Queue), Bellman-Ford, Kruskal y Floyd-Warshall**.
  2. Explique por que el algoritmo de Dijkstra no funciona en un grafo que tiene arcos negativos, de ser necesario puede usar un ejemplo para ilustrar su explicación.
  3. Un grafo bipartito, es un grafo tal que el conjunto de nodos se puede expresar como dos conjuntos disjuntos, de manera que no hay dos nodos del mismo conjunto que sean adyacentes. Diseñe un algoritmo que reciba un grafo, en su implementación indique si su grafo esta implementado como lista o matriz de adyacencia, y retorne *True* si el grafo es bipartito y *False* en otro caso. La complejidad no debe superar  $\mathcal{O}(|V||E|)$ , donde  $V$  es el conjunto de nodos y  $E$  el conjunto de arcos del grafo.
  4. Un grafo ponderado desconectado no tiene árboles de expansión. Sin embargo, es posible encontrar un bosque de expansión de peso mínimo en un grafo de este tipo. Explique cómo modificar tanto el algoritmo de Kruskal como el algoritmo de Prim para hacer esto.
  5. Explique porque el flujo máximo se alcanza cuando la red residual no contiene caminos de aumento.
  6. Describa un algoritmo eficiente que encuentre un nuevo flujo máximo si la capacidad de una arista concreta aumenta en una unidad. Describa un algoritmo eficiente que encuentre un nuevo flujo máximo si la capacidad de una arista concreta disminuye en una unidad.
  7. Ejercicios prácticos propuestos:
    - (a) [1334. Find the City With the Smallest Number of Neighbors at a Threshold Distance](#)
    - (b) [743. Network Delay Time](#)
    - (c) [UVA11396 - Claw Decomposition](#)
    - (d) [UVA11367 - Full Tank?](#)
    - (e) [UVA11747 - Heavy Cycle Edges](#)

- (f) [UVA1317 - Concert Hall Scheduling](#)
- (g) [UVA10806 - Dijkstra, Dijkstra](#). Se puede resolver como un problema de flujo?