Pregunta 1.

a ¿Cuál es el orden de complejidad del algoritmo de Prim en el peor caso por n y m fijado?
Solución:
Para un árbol de n nodos y m arcos tal que cada par de nodos esté conectado (grafo completo) se tienen entonces que $m = n \cdot (n-1)$, por lo tanto como el algoritmo de Prim debe visitar todos los arcos e implementándolo con una tabla desordenada con búsqueda lineal su complejidad sería $O(n \cdot (n-1)) = O(n^2)$.

□

b. ¿Se puede mejorar el algoritmo de Prim con una lista ordenada de los arcos?
Solución:
No, si se tuviera una lista ordenada desde un comienzo se perdería la eficiencia al tener que inspeccionar si los nodos correspondientes a cada arco sean alcanzable en el spanning tree generado.

□

c. Se puede mejorar el algoritmo de Prim con una estructura de tipo Union Find? Hint: Las estructuras de datos de tipo Union Find, descritas en `https://en.wikipedia.org/wiki/Disjoint-set_data_structure`, soportan las operaciones de creación y de unión de conjuntos en tiempo a dentro de $log^*(O(n))$, y las consultas para saber si dos elementos están en el mismo conjunto o no en tiempo amortizado a dentro de $O(\alpha(n))$, donde $\alpha(n)$ es la función invertida de Ackerman, cual crece muy muy lentamente.

Solución:
Esta implementación utilizando estructuras unión find es utilizada por el algoritmo de Kruskal y no tendría mucho uso en el algoritmo de Prim el cual se enfoca principalmente en generar nodos conexos por arcos de menor peso entre los nodos ya agregados al árbol, utilizando estructuras union find se perdería eficiencia al comparar los nodos de árboles disjuntos que no estén unidos por el arco con menor peso entre ellos y que a la vez sean alcanzables.

□