1. ⋆Sea T un árbol generador de un grafo (conexo) G con raíz r, y sean V y W los vértices que están a distancia par e impar de r, respectivamente.

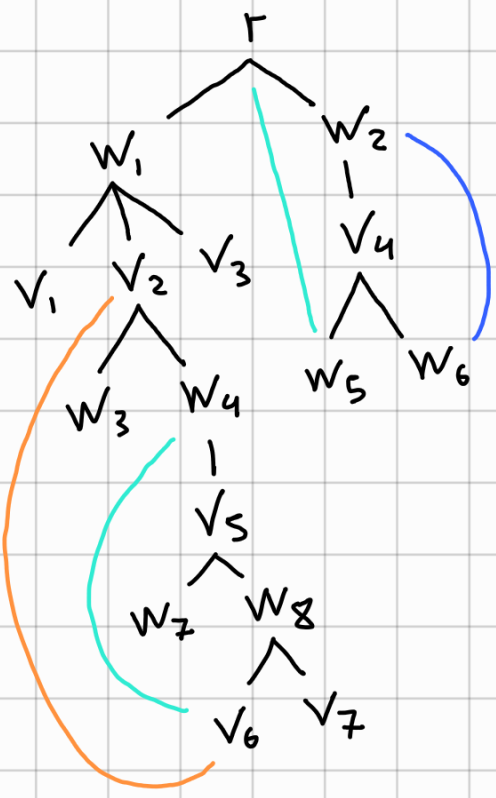
   a) Observar que si existe una arista $vw \in E(G) \setminus E(T)$ tal que $v, w \in V$ o $v, w \in W$, entonces el único ciclo de $T \cup \{vw\}$ tiene longitud impar.

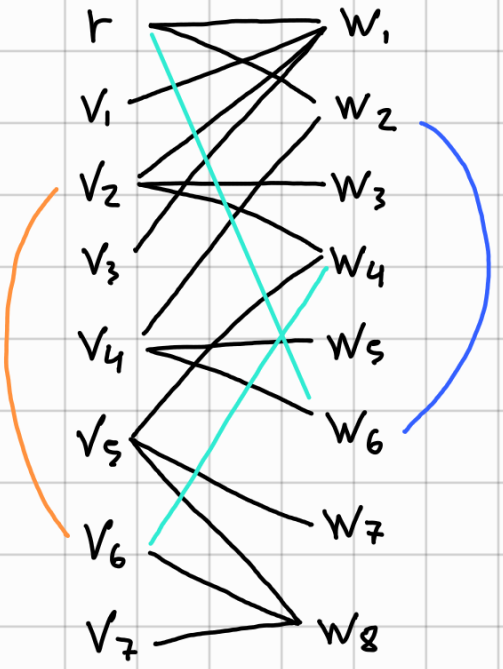
   b) Observar también que si toda arista de $E(G) \setminus E(T)$ une un vértice de V con otro de W, entonces (V, W) es una bipartición de G y, por lo tanto, G es bipartito.

   c) A partir de las observaciones anteriores, diseñar un algoritmo lineal para determinar si un grafo conexo G es bipartito. En caso afirmativo, el algoritmo debe retornar una bipartición de G. En caso negativo, el algoritmo debe retornar un ciclo impar de G. **Explicitar cómo es la implementación del algoritmo**; no es necesario incluir el código.

   d) Generalizar el algoritmo del inciso anterior a grafos no necesariamente conexos observando que un grafo G es bipartito si y solo si sus componentes conexas son bipartitas.

$v, w \in V$ $\quad v, w \in W$ $\quad e = (v, w)$ $\quad \forall e \in G_E \setminus T_E$





1) Inicializamos los conjuntos V y W para guardar la bipartición
2) Hacemos DFS sobre G trackeando 2 cosas para cada vértice:
   - El nivel del árbol en el que estamos (o la distancia con la raíz): v.nivel
   - Quién es el padre: v.padre
3) Al vértice r que tomamos como raíz: r.nivel ← 0
4) Para cada vecino u de cada vértice v que visitamos:
   - Si encontramos un backedge (v,u):
     - Si v.nivel y u.nivel ambos son par o impar, entonces la arista (v,u) conecta 2 vértices dentro de la misma partición y por lo tanto G no es bipartito. Para encontrar el ciclo C_{vu}, $|C_{vu}| = 2k+1$, recorremos el árbol desde v hasta u a través de los ancestros de v (utilizando la información de padres). Notar que u es un ancestro de v, y como tienen ambos la misma paridad están a un distancia 2k dentro del árbol. Luego la arista (v,u) cierra el ciclo entre v y u el cual resulta con una longitud impar 2k+1.
     - Si v.nivel y u.nivel tienen distinta paridad, encontramos un backedge "bueno" pues conecta 2 vértices que están en distintas particiones. Podemos ignorarlo pues no rompe la bipartición de G que estamos generando.
   - Caso contrario, hay que colocar u en el árbol y visitar sus vecinos.
     - u.nivel ← v.nivel + 1
     - u.padre ← v
     - Si u.nivel es par: $V \leftarrow V \cup \{u\}$ si no $W \leftarrow W \cup \{u\}$

5) Si terminamos el DFS, entonces G es bipartito y tenemos guardada la bipartición en V y W.

Para generalizar el algoritmo a grafos no conexos simplemente corremos el procedimiento para cada componente conexa reutilizando los conjuntos V y W.