Exercici 2 (2 punts). Tenim un graf no dirigit i connex $G = (V, E)$ i una coloració de les arestes amb dos colors, roig i blau ($c : E \to \{R, B\}$). Doneu un algorisme per a obtenir un arbre d'expansió amb el mínim nombre d'arestes blaves.

Solución A:

Sea $n = |V|$, para que el problema tenga sentido $n > 1$, si no no hay aristas. Si $n = 2$, solo puedo tener una arista que determina un único árbol de expansión, que por supuesto tiene número mínimo de aristas azules.

Si $n > 2$, Asigno peso a las aristas de E, $w(e) = 1$ si e es azul y $w(e) = 0$ si e es roja.

Sea T un MST de G con esta asignación de pesos, $w(T)$ es exactamente el número de aristas azules en T. Por lo tanto, un MST T es un árbol con el número mínimo de aristas azules posibles.

Para calcular el MST utilizo el algoritmo de Prim. Podemos adaptar su implementación para hacerla más eficiente.

Podemos prescindir de la cola de prioridad que utiliza la implementación, manteniendo una lista de aristas rojas y otra de azules. Mientras tengamos aristas rojas utilizaremos estas y cuando esa lista quede vacía pasaremos a tratar las azules. Una vez incorporado un nuevo vértice al MST, insertaremos las aristas del corte en la lista correspondiente a su color.

Al no tener que gastar $O(\lg n)$ en la inserción, el coste total del algoritmo adaptado es $O(n + m)$.

Solución B:

Sea $n = |V|$, para que el problema tenga sentido $n > 1$, si no no hay aristas. Si $n = 2$, solo puedo tener una arista que determina un único árbol de expansión, que por supuesto tiene número mínimo de aristas azules.

Si $n > 2$, Asigno peso a las aristas de E, $w(e) = 1$ si e es azul y $w(e) = 0$ si e es roja.

Sea T un MST de G con esta asignación de pesos, $w(T)$ es exactamente el número de aristas azules en T. Por lo tanto, un MST T es un árbol con el número mínimo de aristas azules posibles.

Para calcular el MST utilizo el algoritmo de Kruskal. Podemos adaptar su implementación para hacerla más eficiente.

Podemos mejorar el coste de la ordenación inicial, ya que solo tenemos dos valores y nos basta con insertar las aristas rojas al inicio de una lista y las azules al final. Con esto, en tiempo $O(n)$ tenemos ordenadas las aristas.

El resto del algoritmo tiene el coste amortizado de n operaciones MakeSet y $O(m)$ operaciones de tipo Union o Find.

El coste total del algoritmo es $O(\alpha(n)m$ donde α es la inversa de la funció de Ackermann.

Solución C:

Para construir el árbol pedido sigo el siguiente procedimiento:

- Sea G_r el grafo formado solo por las aristas rojas. Calculo las componentes conexas de G_r y un bosque de expansión B, formado por un árbol de expansión en cada componente conexa. Esto se puede obtener haciendo un BFS. Si solo hay una componente conexa, tenemos un árbol de expansión con 0 aristas rojas, y hemos acabado.
- Si no, considero el grafo G' en el que cada componente conexa de G' se colapsa a un vértice y donde, dos vértices están conectados, si hay un arista azul entre algún par de vértices en las respectivas componentes conexas.
- Construyo un árbol de expansión T para G', utilizando un BFS.
- El resultado final está formado por las aristas de R y las de T.

El algoritmo produce un árbol ya que el grafo G es conexo. Además si tuviesemos un árbol de expansión con más aristas rojas implicaria necesariamente que el calculo de las componentes conexas fue incorrecto ya que el número total de vertices que se pueden conectar con aristas rojas sería mayor que el total en sus componentes conexas.

El coste total del algoritmo es $O(n + m)$.