Heurística 3:

1. Calculamos todas las aristas (líneas de cada una de las ciudades con todas las otras) y las metemos en una estructura que nos permita fácilmente obtener siempre la más chica. Función: crearAristas()
2. Ahora nuestro objetivo es ir recorriendo cada una de las ciudades restantes pero ponemos 2 restricciones para que no crezca innecesariamente el tamaño del recorrido.

2.a – esFactible() – Sólo vale que una ciudad esté conectada por máximo dos aristas. Por eso esta función cuenta el número de aristas que se están utilizando.

Count\_if cuenta en el rango dado si la functor que se pasa como tercer argumento es true

2.b-hayCiclos() – Recorre desde un extremo de la arista hasta lo más lejos que puede llegar para comprobar que no se haya creado ningún ciclo innecesario.

3- Ejecución de la heurística:

Mientras quede alguna arista sin probar, comprobamos si debemos meterla (es Factible y ¡hayCiclos).

Una vez esto ha sido ejecutado con cerrar\_ciclo, cerramos la última arista que se podría haber quedado suelta en algunos casos, para ello recorremos las aristas y si alguna tiene menos de dos enlaces la cerramos con la otra que se habría quedado sin dos enlaces.

1. Por último pasamos las aristas a trayectoria/ camino (ciudades en orden) con aristasATrayectoria() que va añadiendo primero la primera ciudad, después la segunda, y desde entonces para cada valor del array de aristas, busca el nuevo par, elimina el anterior y añade la ciudad segunda del par.