Consideremos un set de enteros S={a1, …, an}, construimos un grafo G con 2n vértices, en donde cada elemento ai corresponde a dos vértices vi y ui. Para cada vi, se agrega una arista de vi hacia ui con peso ai, y se agrega una arista desde todos los uj hacia ui con peso 0. Para cada ui, se agregan aristas desde ui hacia todos los otros vj con peso 0.

Si encontramos un ciclo de peso 0, entonces todos los pesos desde vi a ui en el ciclo deben ser cero. Por otro lado, si tenemos un subconjunto C ⊆ S que suma 0, construimos un ciclo eligiendo todas las aristas (vi, ui) correspondientes a los elementos de C y se conectan dichas aristas mediante las que tengan peso 0 hasta obtener el ciclo.

Como este problema es por lo menos tan difícil como subset sum, el cual es NP-Completo, tenemos que el problema de encontrar los ciclos de peso cero también es NPC.

**Bellman-Ford**

Funcionamiento: Este algoritmo se basa en la propiedad que indica que, en un grafo no dirigido sin ciclos negativos, un camino mínimo entre dos nodos cualesquiera tiene a lo sumo |V| pasos. Luego, el algoritmo realiza |V| iteraciones, en las que recorre todas las aristas en cada una de ellas. En la iteración sobre la arista (u, v), chequea qué es mejor: “*llegar a v con el camino que ya encontré o pasando por u, sumando el peso de la arista”*. Con esto, se realizan |V|·|E| iteraciones, siendo esa la complejidad del algoritmo.

Este algoritmo corresponse a la programación dinámica, ya que *tiene memoria en cada iteración sobre el coste de camino en iteraciones anteriores*.

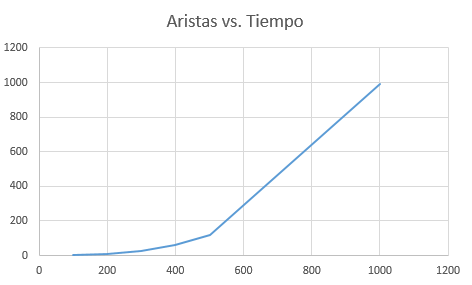
Ventajas: Resuelve el problema siempre que no haya ciclos negativos. Detecta los ciclos negativos.

Desventajas: La complejidad es O(|V|·|E|) para un cada vértice fuente. No encuentra el camino que no repita ningún vértice, en caso de que existan ciclos negativos.

Detección de ciclos negativos: Basándonos en la propiedad indicada al comienzo, una vez que el algoritmo termina las |V| iteraciones sobre cada arista, se deberían conocer todos los caminos mínimos. Por esto, se pueden recorrer todas las aristas una vez más. De esta forma, si se encuentra un mejor camino significa que se viola la propiedad, por lo que se puede concluir que en dicho grafo existe un ciclo negativo.

Complejidad: Como se vio anteriormente, la complejidad del algoritmo es O(|V|·|E|). Pero, en nuestro problema de grafos completos, |E| = |V|2-|V|, por lo que la complejidad queda O(|V|·(|V|2-|V|))=O(|V|3)

En el siguiente gráfico se puede ver la evolución de los tiempos para casos de 100, 200, 300, 400, 500 y 1000 aristas.



**Floyd Warshall**

Funcionamiento: Este algoritmo compara todos los posibles caminos de un nodo a otro. Para esto, utiliza tres iteraciones anidadas sobre los nodos: una para nodo de origen, otra para nodo de destino, y la última para nodos intermedios. Con esto, se define qué es mejor: si el camino directo entre origen destino, o el camino desde origen hacia destino pasando por el intermedio el al paso anterior a llegar al destino. Por esto, el algoritmo es O(|V|3).

Ventajas: La complejidad del algoritmo es independiente de la cantidad de aristas del grafo. Aún así, se prueban todas las combinaciones de aristas. Detecta ciclos negativos.

Desventajas: No encuentra el camino que no repita ningún vértice, en caso de que existan ciclos negativos.

Detección de ciclos negativos: al igual que con Bellman-Ford, la detección de ciclos negativos se realiza realizando una nueva corrida del algoritmo. De esta forma, si se disminuyen los caminos, se concluye que hay un ciclo negativo, especialmente los caminos que tienen como origen y destino el mismo nodo.

Complejidad: la complejidad de este algoritmo es O(|V|3). Para nuestro caso de grafos completos, la complejidad no varía ya que es independiente de la cantidad de aristas.

En el siguiente gráfico se puede ver la evolución de los tiempos para casos de 100, 200, 300, 400, 500 y 1000 aristas.

