# Propuesta: Grafo de relaciones de alimentos

## Grupo 3

Raúl Mauricio Peña Losada, Anderson Vargas, Omar Camilo Sarmiento, Pablo Mancera Salgado, Alejandro Higuera Castro

## 1.Obtención de datos

Si bien los datos de la relación entre cada par de alimentos pueden ser complicados de obtener debido a que la mayoría se mantienen ocultos al público, se puede acceder a datos un paso por debajo, la composición aromática de los alimentos (Ingredientes), y calcularlo algorítmicamente para los nodos.

A continuación presentamos algunos datasets en formato CSV o JSON que contienen datos sobre los alimentos como ingredientes y que pueden llegar a ser útiles:

1. DataSet de Kaggle con 360 alimentos y sus ingredientes (JSON) https://www.kaggle.com/datasets/kaggle/recipe-ingredients-dataset
2. DataSet de Data.world con 10.000 alimentos y sus ingredientes (CSV) https://data.world/datafiniti/food-ingredient-lists

A partir de estos DataSets se puede determinar un coeficiente de similaridad entre los alimentos.

## 2. Problema

A la hora de clasificar el sabor de un alimento, el 80% de lo que consideramos sabor viene en realidad del olor. Por tanto, al analizar el “sabor” de un alimento, podemos utilizar el conocimiento acerca de los compuestos aromáticos presentes para realizar una predicción de que tan bien se van a relacionar dos alimentos, ya sea por alta similitud (vino tinto y queso) o por muy poca (sabores disonantes, p.e. la cocina típica en la India); esto podría llevar a recetas novedosas y pares poco esperados.

## 3. Algoritmo

Inicialmente, buscamos recorrer un grafo haciendo uso de la similitud entre alimentos como un peso y queremos minimizar esta similitud entre los alimentos para poder obtener una receta novedosa y poco esperada.

Debemos tener en cuenta ciertos aspectos técnicos de nuestro grafo antes de precisar el algoritmo o conjunto de problema a solucionar. El primer aspecto es que el grafo sobre el que trabajaremos es no dirigido, ya que la similitud de un alimento con otro es una relación simétrica, el alimento A es tan similar al Alimento B como lo es el alimento B al alimento A. El segundo aspecto es que consideraremos la similitud de un alimento con otro como un número positivo entre 0 y 1, así que estaremos trabajando con pesos siempre positivos en

nuestras aristas. El tercer aspecto, es que dado que todo alimento tendrá su correlación con cualquier otro del conjunto, estaremos hablando de un grafo completo, pues cada par de vértices del conjunto está conectado con una arista.

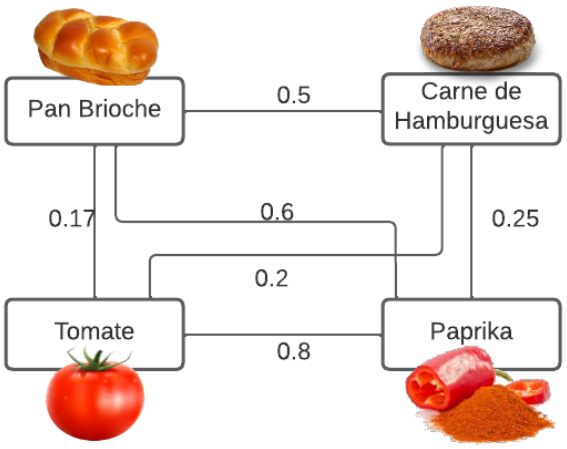


*Ilustración 1. Modelo de grafo para un conjunto de 4 alimentos*

De lo anterior, que nuestro problema se reduce a buscar el Minimum Spanning Tree (MST) / Árbol Recubridor Mínimo, es decir el camino o subconjunto de vértices que no tiene ciclos y tiene la menor suma posible de pesos.

Para encontrar el MST en nuestro caso usaremos algoritmos como el de Kruskal o el de Prims, a continuación detallaremos estos dos algoritmos.

## 3.1. Algoritmo de Kruskal

**Tipo:** Greedy Algorithm/Algoritmo Voraz (Elige el óptimo en cada paso en esperanza de encontrar una solución óptima)
**Descripción:** En cada iteración el Algoritmo de Kruskal buscará la arista con menor peso y la añadirá al MST siempre y cuando no se formen ciclos.
**Complejidad:** O(V+E) (O(n^2) para un grafo completo)
**Pasos:**

1. Ordenar los vértices de acuerdo con sus pesos.
2. Añadir las aristas al MST iniciando con la arista de menor peso hasta la arista de mayor peso.
3. Solo se añadirán artistas que no forman un ciclo, es decir aristas que solo conectan componentes no conectados. (Dado que se necesita saber la conectividad entre dos vértices será necesario usar DFS o Disjoint Sets, siendo Disjoints Sets la mejor solución).

## 3.2. Algoritmo de Prims

**Tipo:** Greedy Algorithm/Algoritmo Voraz (Elige el óptimo en cada paso en esperanza de encontrar una solución óptima)
**Descripción:** A diferencia del algoritmo de Kruskal, opta por añadir vértices para crecer el MST y se empezará a crecer el MST desde una posición determinada.

**Complejidad:** O((V+E)*logV) (O(n^2*log n) para un grafo completo)
**Pasos:**

1. Seleccionar un nodo arbitrario de inicio.
2. Mantener dos conjuntos disyuntos de vértices, uno con los vértices que están en el MST y que no están en el MST.
3. Seleccionar el vértices más económico que está conectado al MST y si no está en el MST se añade.
4. Revisar Ciclos manteniendo una cola para marcar nodos y que estos no se añadan.

## 3.3. Conclusiones

En una primera aproximación al problema se usarán ambos algoritmos para comparar su desempeño respecto al problema. Sin embargo el algoritmo de Kruskal dada su complejidad asintótica de O(n^2) parece ser más eficiente para abordar el problema frente al algoritmo de Prims cuya complejidad asintótica es O(n^2*log n).

# 4. Referencias


[1]A. Barnwal, "Kruskal's Minimum Spanning Tree Algorithm | Greedy Algo-2 - GeeksforGeeks", *GeeksforGeeks*, 2022. [Online]. Available: https://www.geeksforgeeks.org/kruskals-minimum-spanning-tree-algorithm-greedy-algo-2/. [Accessed: 19- May- 2022].

[2]O. Abdelaziz Abdelnabi, "Minimum Spanning Tree Tutorials & Notes | Algorithms | HackerEarth", *HackerEarth*. [Online]. Available: https://www.hackerearth.com/practice/algorithms/graphs/minimum-spanning-tree/tutorial/. [Accessed: 19- May- 2022].

[3]"MST Introduction | Minimum Spanning Tree Introduction - javatpoint", *JavaTPoint*. [Online]. Available: https://www.javatpoint.com/minimum-spanning-tree-introduction. [Accessed: 19- May- 2022].