Arbol generador minimo (AGM).

Concepto:
. Es un AG de G con minima longitud.
. Sea T un AG de G y $c: E(G) \rightarrow R$ su funcion de costo. Entonces,
T es un AGM de (G,c) si y solo si todo camino de T es minimax de (G,c).

Prim

Algoritmo greedy.
Util para grafos densos.
Dado un G conexo, Prim determina un AGM de G.

- **Invariante:** En la k-esima iteracion tenemos un subgrafo de un AGM conexo con k aristas.
- **Complejidad:**
  - Estandar: O(n^2)
  - Heap binario: O((m+n)log(n))
  - Heap fibonacci: O(m+n*log(n))

Kruskal

Algoritmo greedy.
Util para grafos ralos.
Dado un G conexo, Kruskal determina un AGM de G.

- **Invariante:** En la k-esima iteracion obtenemos un subgafo de un AGM con k aristas sin ciclos.
- **Complejidad:**
  - Implementacion trivial: O(mn)
  - Union & Find: O(m*log(n))
  - Union & Find (version 2): O(m*log(n) + m*alpha(n))

Complejidades: $O(min(N^2, M * log(N)))$

Camino Minimo (uno a todos).

Concepto:
Propiedad de estructura suboptima de un camino minimo.

Dijkstra

Dado un grafo G orientado con pesos no negativos en las aristas, Dijkstra determina el camino minimo entre v y el resto de los nodos de G.

- **Invariante**: Al finalizar la k-esima iteracion tenemos el camino minimo entre v y los nodos de Sk.
- **Complejidad**:
    - Trivial: O(n^2)
    - Heap binario: O((m+n)log(n))
    - Heap fibonacci: O(m+nlog(n))

  O(min(n^2, m log n))

Bellman-Ford

. Usa programacion dinamica.

. Dado un G orientado sin aristas de longitud negativa alcamzables desde v, Ford determina un CM entre v y cada nodo alcanzable desde v.

. Si existe un ciclo de longitud negativa alcanzable desde v entonces hayun cambio de π en toda iteracion de la ejecucion de Ford.

. **Prop**: G tiene un ciclo de long negativa $\Leftrightarrow$ G' subgrafo solo con aristas (pred(u),u) tiene un ciclo con al menos 2 aristas.

- **Invariante**: Después de k iteraciones, dist(v) es la longitud del camino más corto desde s hasta v que usa como máximo k aristas.
- **Complejidad**:
    - Estandar: O(mn)

Camino Minimo (todos a todos, algoritmos matriciales).

A partir de G=(V,X) obtenemos una matriz de distancias D y una matriz de sucesores S.

D[I][J] = Distancia del camino minimo entre i y j.
S[I][J] = Es el primer sucesor desde i en el camino de i a j.

Floyd-Warshall

$$d_{ij}^{\ k} = min(d_{ij}^{\ k-1}, d_{ik}^{\ k-1} + d_{jk}^{\ k-1})$$

- **Invariante:** Al terminar la k-esima iteracion del algoritmo de Floyd, Dij es la longitud de los caminos minimos desde Vi a Vj solo pasando por vertices intermedios de {V1,...,Vk}.
- **Complejidad:**
  - Temporal: O(n^3)
  - Espacial: O(n^2)

Dantzig:

El algoritmo de Dantzig determina los CM entre todos los pares de nofos de un grafo orientado sin circuitos.

- **Invariante:** En la k-esima iteracion, Dantzig genera una matriz de k*k de caminos minimos en el subgrafo inducido por los vertices {v1,...,vk}.
- **Complejidad:** O(n^3)

Dijkstra en todos los nodos (Johnson)

- Complejidad: O(n*m*log (n)). ??
- Para un grafo ralo

Flujo Maximo.

Red: Digrafo con vertices especiales s y t.

Flujo factible: Es una funcion $f: X \rightarrow R^{+}$que cumple:

- [1] Restriccion de capacidades: $0 \leq f(e) \leq c(e)$
- [2] Ley de conservacion del flujo: $\sum_{e \in in(v)} f(e) = \sum_{e \in out(v)} f(e)$

El **valor del flujo** v(f)=F es

$$F = \sum_{e \in in(t)} f(e) - \sum_{e \in out(t)} f(e) = \sum_{e \in out(s)} f(e) - \sum_{e \in in(s)} f(e)$$

Corte: Un corte en una red es un subconjunto $S \subset V - \{t\}$ tal que $s \in S$.

Capacidad de un corte:

$$c(S) := \sum_{e \in ST} c(e), ST = \{(x \rightarrow y) \in E(N), x \in S, y \in T\}$$

Para cualquier corte vale que $F <= c(S)$.

Corte minimo:
Determinar un corte S con capacidad minima.
Si S es un corte de capacidad minima entonces v(f)=F=C(s).

Obtener el corte minimo a partir de el flujo maximo:
Recorrer BFS/DFS desde la red residual tras la ultima iteracion del algoritmo elegido. Todos los nodos alcanzables desde s son parte del corte minimo.

Red residual: Se define R=(V, X_R) a partir de una red N, una funcion de flujo valida f y una funcion de capacidades c.

- $(v \rightarrow w) \in X_R \quad si\, f((v \rightarrow w)) < c((v \rightarrow w))$
- $(w \rightarrow v) \in X_R \quad si\, f((v \rightarrow w)) > 0$

Camino de aumento:
Es un camino orientado de s a t en R.
Se pueden obtener en O(m).

El **algoritmo de camino de aumento** determina un camino de aumento en R=(N,F) si existe, en caso contrario devuelve un corte S de N.

Teo:
Sea N=(V,X) y f un flujo factible.
F flujo factibilidad de valor maximo ⇔ No existe camino de aumento en R(N,F).

Metodo Ford & Fulkerson:

Obtiene el flujo maximo con complejidad O(nmU), siendo U = max{c((i->j))}

No determina como hallar los caminos de aumento.

Compejidad: O(nmU), siendo U = max{c((i->j))}.

La razón de esta complejidad es que en cada iteración, se puede aumentar el flujo en al menos una unidad, y en el peor de los casos, se requieren U iteraciones por cada arista.

Teo: Si las capacidades son entears entonces el problema tiene un flujo maximo entero (idea demo: es la capacidad de un corte minimo).

Algoritmo de Edmunds & Karp:

El algoritmo Edmonds-Karp es una implementación específica del algoritmo Ford-Fulkerson que utiliza una búsqueda en amplitud (BFS) para encontrar los caminos aumentantes más cortos en términos del número de aristas.

Complejidad: O(nm^2)

Orden topológico:

Se puede obtener en O(N+M) mediante el algoritmos de Kahn o una adaptacion del DFS.

El orden topológico es una secuencia lineal de los vértices de un DAG tal que, para cada arista dirigida (u,v), el vértice u aparece antes que v en la secuencia.

Camino minimo en DAG (desde s al resto de nodos):

1. Obtenemos el orden topologico del DAG.
2. Procesamos d(u) en el orden cronologico utilizando programacion dinamica.

Procesar d(u):
$d(s) = 0$
$d(v) = min_u \{w((u \to v)) + d(u)\}$

Complejidad: O(N+M)

DIJKSTRA

Inicialización:

- Se elige un nodo de origen y se establece su distancia a 0 (distancia desde el nodo de origen a sí mismo).
- Se establece la distancia de todos los demás nodos a infinito (∞).
- Se crea un conjunto de nodos no visitados.

Selección del Nodo Más Cercano:

- Se selecciona el nodo no visitado con la distancia más corta desde el nodo de origen (inicialmente, este será el nodo de origen).

Actualización de Distancias:

- Para el nodo seleccionado, se consideran todos sus vecinos no visitados.
- Para cada vecino, se calcula la distancia desde el nodo de origen pasando por el nodo seleccionado.
- Si esta nueva distancia es menor que la distancia conocida actualmente para ese vecino, se actualiza la distancia.

Marcado del Nodo como Visitado:

- Una vez que se han considerado todos los vecinos del nodo seleccionado, se marca este nodo como visitado y se elimina del conjunto de nodos no visitados.

Repetición:

- Se repiten los pasos 2-4 hasta que todos los nodos hayan sido visitados.

2. Un 1-´arbol es un ´arbol con una arista agregada. Determinar cu´ales de las siguientes afirmaciones son verdaderas.

a) El grafo tiene n aristas. V
b) Existe un ´unico camino entre cada par de v´ertices. F
c) El grafo puede contener dos ciclos. F
d) Todos los v´ertices tienen m´as de un vecino. F
e) Si asignamos longitudes/pesos a las aristas de un grafo G, de manera tal que todas aristas tienen distintas longitudes, entonces G tiene un ´unico 1-´arbol generador minimo. V

4. Sea G un grafo conexo de n v´ertices y al menos n aristas con costos asociados. Sea emin una arista de mınimo costo, y emax una arista de costo m´aximo.

a) ¿Es cierto que emin pertenece a alg´un ´arbol generador m´ınimo?

Se

b) ¿Es cierto que emin pertenece a todo ´arbol generador m´ınimo?

No, no explicita que todas las aristas tengan distinta longitud entonces puede pasar que haya dos aristas minimas y una de ellas se quede afuera.

c) ¿Es cierto que no existe un ´arbol generador m´ınimo tal que emax pertenezca a dicho ´arbol?

No, puede pasar que haya dos aristas de long max y una tenga que entrar si o si. Ademas capaz incluirla reduce la suma de todas las aristas del AG.

El algoritmo de Dijkstra, se puede aplicar tanto para grafos orientados como para grafos no orientados con longitudes/pesos en sus aristas. En el caso de grafos conexos no orientados, las aristas elegidas por el algoritmo para generar los caminos m´ınimos, forman un ´arbol generador (podemos llamarlo como un ´arbol generador de Dijkstra) del grafo en cuesti´on. Para el contexto de un grafo conexo G, responder los siguientes puntos.

a) ¿Es cierto que todo ´arbol generador de Dijkstra sea m´ınimo?

No

b) ¿Es cierto que todo ´arbol generador m´ınimo sea de Dijkstra?

No.

c) Mostrar un ´arbol generador de Dijkstra que sea tambi´en m´ınimo.

d) ¿Es cierto que siempre existe un ´arbol generador de Dijkstra q sea m´ınimo?

¿Todo dígrafo tiene un orden topológico?

- No, solo los dígrafos sin ciclos (DAG) tienen un orden topológico.

¿Todo dígrafo que admita un orden topológico es un DAG?

- Sí, si un dígrafo tiene un orden topológico, entonces es un DAG.

¿Todo DAG tiene un orden topológico? ¿Es único ese orden topológico?

- Sí, todo DAG tiene al menos un orden topológico.
- No, el orden topológico no es necesariamente único.

¿Cuál es la máxima cantidad de aristas que puede tener un DAG de nnn nodos?

- La máxima cantidad de aristas en un DAG de n nodos es $n(n-1)/2$

Sí, en un grafo con longitudes de aristas de distinto peso, es cierto que en todo árbol generador mínimo (MST, por sus siglas en inglés) está incluida la arista con el peso mínimo. Esta afirmación es válida y se fundamenta en los principios de los algoritmos que se utilizan para encontrar el MST, como el algoritmo de Kruskal y el algoritmo de Prim.