Tabla Complejidades Algorítmicas

NOMBRE	USOS	COMPLEJIDAD
BFS	Distancias en (di)grafos no pesados	$O(m+n)$
DFS	Detección de componentes y ciclos	$O(m+n)$
Primm	Encuentra árbol generador mínimo(máximo)	$O(n^2)$ matriz $O(m\log(n))$ binary heap $O((m+n)\log(n))$
Kruskal	Encuentra árbol generador mínimo(máximo)	$O(m\log(n))$ [1]
Dijkstra	Distancias en (di)grafos pesados uno a todos. No	$O((n+m)*log(n))$ cola de prio $O(n^2)$ vector
Bellman-Ford	Distancias en (di)grafos pesados uno a todos. Puede encontrar ciclos de costo negativo(con modificación)	$O(n*m)$
DAGs	Distancias en dags(directed acyclical graphs)	$O(n+m)$
Floyd-Warshall	Distancias en (di)grafos pesados de todos contra todos	$O(n^3)$
Johnson (NO DEBERÍA ENTRAR)	Distancias en (di)grafos pesados de todos contra todos	$O(n^2\log(n)+mn)$ $O(n*m*log(n))$
Ford-Fulkerson	Encuentra flujo maximo para una red(Con capacidades racionales)	$O(nmU)$
Edmonds-Karp	Modificación al algoritmo de Ford-Fulkerson	$O(nm^2)$
Cancelacion de Ciclos	Dado un flujo, encuentra otro de mismo valor de flujo y de costo minimo.	$O(nm^2CU)$ $O(\min\{nm\lg(nC),\ nm^2\lg(n)\})$
Succesive Shortest Paths	Encontrar un flujo maximo de costo minimo	$O(m\lg(U)*\min\{n^2, m\lg(n)\})$ $O(mn^2U)$

Propiedades

Caminos minimos y Relajaciones

```
RELAX( u, v, w):
    if(v.d > u.d + w(u,v)):
        v.d = u.d + w(u,v)
        v.pred = u
```

1. Una arista $v \to w$ pertenece a un camino minimo entre s y t si $d(s,v) + c(v \to w) + d(w,t) = d(s,t)$
2. **Desigualdad triangular:** Para toda arista $(u,v) \in E$, tenemos que $\delta(s,v) \leq \delta(s,u) + w(u,v)$
3. **Cota superior:** La distancia real es menor o igual a la estimada(.d) $v.d \geq \delta(s,v)$ para todo vertice, y cuando $v.d == \delta(s,v)$ no vuelve a cambiar
4. **No hay camino:** Si no hay camino entre s y v entonces $v.d == \delta(s,v) == \infty$
5. **Convergencia:** Si $s \to \ldots \to u \to v$ es un camino minimo en G para algun $u, v \in V$, y si $u.d = \delta(s,u)$ en algun momento antes de relajar el edge (u,v); entonces despues de relajar el edge $v.d = \delta(s,v)$ para siempre.
6. **Relajación de camino mínimo:** si $p = \langle s = v_0, v_1, \ldots, v_k \rangle$ es un camino minimo desde s a v_k y relajamos los edges de p en orden, entonces despues de todos las relajaciones $v_k.d = \delta(s, v_k)$. Esto pasa incluso si hay relajaciones en el medio que afecten a los nodos en p.
7. **Subgrafo de predecesores:** una vez que $v.d = \delta(s,v)\ \forall\ v \in V$ el subgrafo predecesor es un árbol de caminos mínimos enraizados en s.

Difference Constraints(SRD)

Se pueden reescribir ecuaciones para que queden de la forma $x_i - x_j \leq c$

1. $x_i - x_j \geq c \iff -x_i + x_j \leq -c$
2. $x_i - x_j = c \iff x_i - x_j \leq c \ \wedge\ x_i - x_j \geq c$
3. $x_i + x_j \leq c$ agregamos una variable $y_j = -x_j \implies x_i - y_j \leq c$
4. $x_i \leq c$ agregamos una variable z y las restricciones $x_i - z \leq c \ \wedge\ z - c_i \leq -1$

Si tenemos una solucion para un SRD $x_1, \ldots, x_n$ entonces $x_1 + d, \ldots, x_n + d$ tambien es solución.

Flujos

1. Conservación: Salvo s y t, el flujo que entra es igual al que sale
2. Restricción capacidad
3. Red residual G_x del flujo x reemplazando cada arco $ij \in A$ por dos arcos ij y ji cumpliendo:
    1. El arco ij tien costo c_{ij} y capacidad residual $r_{ij} = u_{ij} - x_{ij}$
    2. El arco ji tiene costo $-c_{ij}$ y capacidad residual $r_{ji} = x_{ij}$
4. Capacidad de un corte es la suma de las aristas que salen
5. El valor del flujo máximo es igual al corte con capacidad mínima.
6. Dado un flujo maximo podemos obtener el corte minimo: (practica/gfg)

1. Obtengo red residual en O(m)
2. Corro DFS desde s O(n+m)
3. Las aristas alcanzables son parte del corte minimo.

P vs NP

Es un problema de desición Π,

1. Instancia Positiva: Aquellas para las que la respuesta a Π es SI
2. Instancia Negativa: Aquellas para las que la respuesta a Π es NO

Problema complemento: el problema $\bar{\Pi}$ donde la respuesta a las instancias positvas de Π es NO, y a las negativas de Π es SI

Reducción polinomial: transformación f de cualquier instancia de Π_1 a una instancia construida por nosotros del problema Π_2 en tiempo polinomial, tal que una instancia I es positiva de Π_1 si y solo si $f(I)$ es una instancia positiva de Π_2. ($\Pi_1 \leq_p \Pi_2$). Π_2 no es polinomialmente más dificil que Π_1.

Clases:

1. **P:** problemas que se pueden resolver en tiempo polinomial en una maquina de Turing deterministica
2. **NP:** problemas que se pueden resolver en tiempo polinomial en una maquina de Turing no determistica, o que se puede certificar en tiempo polinomial una respuesta positiva en una maquina deterministica.
3. **CoNP:** Problemas que tienen un problema certificante negativo que se puede resolver en tiempo polinomial.

Relaciones:

1. P $\subseteq$ NP, P $\subseteq$ CoNP
2. $\Pi \in$ P $\iff$ $\bar{\Pi} \in$ P
3. $\Pi \in$ NP $\iff$ $\bar{\Pi} \in$ CoNP

NP-Hard:
Un problema igual o más dificil que NP. Para todos los problemas en NP se pueden reducir polinomialmente a este.

NP-Complete:
Un problema que pertenece a NP y es NP-hard

---

1. Tambien puede expresarse $O(m\log(m))$ pero como $m \leq n^2 \implies \log(m) \leq 2\log(n)$ ↩