Demostración complejidad Dinic

Nicolás Cagliero

24 de junio de 2024

Theorem 1. *El algoritmo Dinic en ambas versiones tiene complejidad* $O(m \cdot n^2)$

Demostración. Esta será una idea de la demostración y sus partes, los detalles y ciertas cuentas no serán escritas.

Sabemos que la distancia en Networks Auxiliares sucesivos aumenta, por ende sabes que vamos a tener $O(n)$ NA en una corrida de Dinic. Luego sabemos que $O(Dinic) = O(n) \cdot O$(compl(hallar un flujo bloqueante en un NA)+compl(construir un NA))

La complejidad de armar un NA podemos ver que está dada por la cantidad de lados que puede llegar a tener $\therefore$ compl(construir un NA) $= O(m)$

Si logramos probar que la compl(hallar un flujo bloqueante en un NA) $= O(n \cdot m)$ entonces queda demostrado el teorema.

1. Versión original:

   En esta versión siempre vamos a contar con una propiedad, la cual es que todos los vértices tienen lineas de salida. Esto hace que el encontrar un camino que aumente nuestro flujo con DFS siempre tenga complejidad $O(n) = O(\#Niveles)$, esto se debe a que no tenemos que hacer backtracking para llegar a t. En cada uno de estos caminos, se borra al menos un lado $\therefore$ hay $O(m)$ caminos.

   $\Rightarrow$ la complejidad de hallar un flujo bloqueante es $O(m \cdot n)$

   De esta forma todavía no queda demostrado el teorema pues puede suceder que para cumplir la propiedad durante toda la corrida de Dinic nos aumente la complejidad (Predict: no).

   Veamos la complejidad de mantener esta propiedad,. Lo que se debe hacer es, luego de cada corrida, revisar que todos los vértices tengan líneas de salida ($O(1)$). Si un vértice tiene línea de salida, perfecto, si no, se borra el vértice. Hay $O(n)$ vértices y $O(m)$ caminos, $\Rightarrow$ esta parte es $O(m \cdot n)$ también.

   Falta la complejidad de borrar el vértice y sus lados que llegan a él. Esto se hace a lo sumo una vez por vértice, analicemos la complejidad total sobre todos los "podar". Borrar un vértice y sus lados es $O(d(x))$, sobre todos los vértices es $\sum O(d(x)) = O(m)$

   Luego, la complejidad de hallar el flujo bloqueante manteniendo la propiedad es $O(n \cdot m)+O(n \cdot m)+O(m) = O(n \cdot m)$

2. Versión Occidental:

Para hallar flujo bloqueante en un NA vimos un algoritmo que consta de 3 principales partes: A (avanzar), R (retroceder) e I (incrementar) y analizando el código se puede ver que una corrida de Dinic es una "palabra"de la forma $AA..AIAA...ARA..AAR.....$

Miraremos subpalabras de la forma $AA...AX$ donde X es R o I. Notar que tanto la complejidad de avanzar como retroceder es constante, mientras que la de Incrementar es recorrer un camino de longitud n dos veces $\therefore$ la complejidad I es $O(n)$

Veamos cuántas A hay en $AA...AX$, cada avanzar mueve el pivote x a un nivel superior, $\therefore$ hay $O(n)A's$ en cada $AAA..AX$.

$\Rightarrow$ la complejidad de $A..AX = O(n)$

Veamos cuántas subpalabras hay, cada R borra un lado y cada I borra al menos un lado $\Rightarrow$ hay $O(m)$ palabras $AA..AX$ $\therefore$ la complejidad total es $O(n \cdot m)$

□