Contents

Complejidad “naive”de Dinitz

depende de cuantos networks auxiliares tengamos que construir.

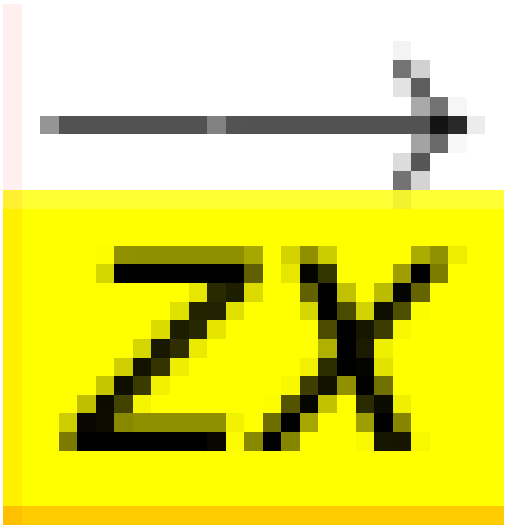
La construcción de cada uno es O(m).

complejidad total de hallar un flujo bloqueante en un network auxiliar igual a O(m2).

la complejidad total de Dinica seria n.(O(m) + O(m2)) = O(nm2),

idea de la implementación de Ever:

cuando corremos DFS, si llegamos a un vértice x que no tiene vecinos, debemos hacer un backtrack, borrando a x de la pila y usando el vértice anterior a x en el camino para seguir buscando. Esa información de que es inútil seguir buscando por x **no deberiamos perderla** y hay que “guardarla” para futuras corridas de DFS. La forma que tiene Ever de “guardar”esa información es simplemente borrar x, o bien, si hacemos backtrack desde x a z,



borrar el lado $- \rightarrow zx$.

Diferencia entre la version rusa y la occidental de Dinitz

La diferencia entre Dinitz y Dinic-Even es en cómo y cuando se actualiza el network a medida que encontramos nuevos caminos.

Ever

borra varios lados (o vértices) extras mientras corre DFS, cada vez que tiene que hacer un backtrack.

Dinitz,

el network auxiliar se construye de forma tal que **DFS nunca tenga que hacer backtrack.**

Y cada vez que se encuentra un camino entre s y t y se cambia el flujo, tambien se cambia el network auxiliar para seguir teniendo esta propiedad.

Es decir,

en el original se usa un poco mas de tiempo actualizando el network auxiliar luego de cada camino,

mientras que en la versión de Even, no se pierde tanto tiempo actualizando el network auxiliar **entre** caminos, pero las busquedas DFS no demoran todas igual

Ever es mas “lazy” y sólo borra lados si los encuentra y se da cuenta que no los necesita, mientras que la versión original de Dinitz es mas proactiva y borra todos los lados que sabe que son inútiles aún si luego nunca los

encontraria.