Contents

1 Algoritmos **2**

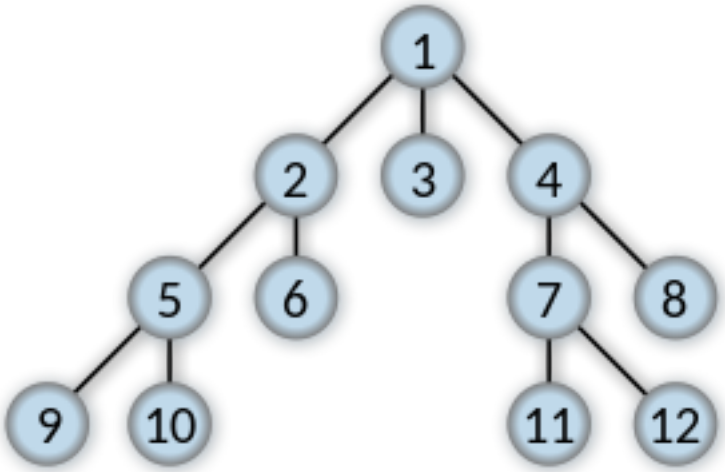
1.1 BFS . . . 2
1.1.1 Que hace? . . . 2
1.1.2 Que tener en cuenta al usarlo? . . . 2
1.2 EK . . . 2
1.2.1 Idea . . . 2
1.2.2 Notacion . . . 2
1.2.3 Armar camino aumentante . . . 3
1.2.3.1 A tener en cuenta . . . 3
1.2.4 Pasos . . . 3
1.2.5 Verficar . . . 4
1.2.6 Cosas a tener en cuenta . . . 4
1.3 Dinic . . . 4
1.3.1 Armar network auxiliar . . . 4
1.3.2 Pasos . . . 5
1.4 Wave . . . 5
1.4.1 Pasos . . . 6

1 Algoritmos

1.1 BFS

1.1.1 Que hace?

BFS arma un arbol de la forma:



1.1.2 Que tener en cuenta al usarlo?

- Solo añadir nodos que esten al mismo nivel
- No volver añadir nodos ya recorridos

1.2 EK

1.2.1 Idea

EK puede ser pensado como un algoritmo para encontrar el menor camino aumentante de s hasta t

1.2.2 Notacion

Para resolver EK usamos la siguiente notacion:

```
sB C A  D t
 s B C- A D
 9 9 7  5 5
```

1) La primer columna son los vertices que vamos agregando
2) La segunda columna es quien los agrego, si es un lado backward lleva exponente "-"
3) La tercer columna es el flujo que puede ser mandado

1.2.3 Armar camino aumentante

Primer nodo descubierto: s

Por cada nodo descubierto:

1. Añadir vecinos del nodo al camino aumentante
    - Añadir si:
        - Es vecino forward
        - Es vecino backward
        - El lado aun tiene capacidad disponible
        - El nodo aun no fue descubierto
    - Se escribe: Nombre, padre y capacidad real
2. Tachar nodo
3. Avanzar con el siguiente nodo en la lista de nodos descubiertos y volver a 1

1.2.3.1 A tener en cuenta

- No olvidar agregar nodos backward!

1.2.4 Pasos

1) Armar primer camino aumentante (BFS) con notacion adecuada
2) Reconstruir camino empezando desde t (leyendo segunda fila)
3) Actualizar tabla con las capacidades nuevas
    - Solo cambia si pertenece al camino aumentante formado
4) Armar siguiente camino aumentante

1.2.5 Verficar

Para verificar se puede calcular si el valor del flujo es igual a la capacidad del corte minimal

Una vez llegado al punto que no se pueda armar otro camino:

1. Definir este camino incompleto como un corte S
2. Calcular valor de f ($v(f)$)
    - Ver cuanto flujo sale de s y sumarlo
3. Calcular capacidad de S
    - Sumar capacidades de los lados que salen del corte
    - Osea, todos los lados xy donde x pertenezca a S, pero y no
4. Chequear que $v(f) = C(S)$
5. Concluir que f es flujo maximal y S corte minimal

1.2.6 Cosas a tener en cuenta

- El BFS de cada camino aumentante es independiente del anterior
    - Tampoco toma caminos saturados
    - Si está saturado es como que se elimina ese lado
- En la notacion de camino aumentante solo poner la cantidad que de verdad puede ser mandada, no la total
- Los lados backward tambien agregan vecinos!

1.3 Dinic

1.3.1 Armar network auxiliar

- Solo se añaden si estan a mayor distancia de s que el padre
- Se pueden añadir lados a nodos ya descubiertos
- Apenas se llega a t se para de añadir nodos
- Tambien se añaden lados backward al network auxiliar

1.3.2 Pasos

1. Armar network auxiliar por niveles (BFS)

- A partir de las segunda iteracion se toman solo los que tienen capacidad mayor a 0

1. Hacer DFS desde s hasta t hasta que se llegue a un flujo bloqueante

- Si se llega a un camino sin salida hacer backtrack y seguir hasta llegar a t

1. Repetir pasos anteriores

- Flujo bloqueante: cuando no se pueden encontrar mas caminos desde s hasta t porque todos los vertices han sido saturados
- Cuando hay un flujo backward se toma la capacidad que se está usando al calcular el valor del camino

1.4 Wave

1. Hacer lista de vertices ordenados por niveles y alfabeticamente
    - Se va armando una tablita (2D) con lo que manda cada vertice

Ola forward:

1. Cada vertice manda todo lo que puede a sus vecinos
2. Marcar los vertices que quedan bloqueados (no pueden mandar el flujo que tienen)
3. Una vez se termina de mandar todo se hace una linea al final de la tablita y se hace una flechita hacia la derecha en algun lado de la parte de arriba, como para denotar que es forward

Ola backward:

1. Arrancar desde los vertices bloqueados
2. Devuelve todo lo que puede
    - Para devolver usar orden antialfabetico (solo se puede devolver a alguien que manda algo)

Se hace ola forward de nuevo

1. Se manda hacia adelante en los vertices desbalanceados, pero no se manda a los bloqueados

Se hace de nuevo lado backward

1.4.1 Pasos

1. Armar tablita de nodos
2. s manda a sus hijos todo el flujo que tiene
3. Los hijos de s mandan todo lo que pueden a sus hijos
    - Sin sobrepasar lo que les mandó s
    - Se actualiza en la tabla el flujo disponible en el padre
    - Si no se manda todo lo que se pudo mandar se encierra en un cuadrado el numero
    - Una vez se bloquean no se desbloquean más

Ola backward:

1. Toma solo los nodos bloqueados
2. Cada nodo bloqueado devuelve al que le mandó

- Devolver en orden antialfabetico

Ola forward:

- No se manda a los bloqueados