‣ **Max-flow and min-cut problems**

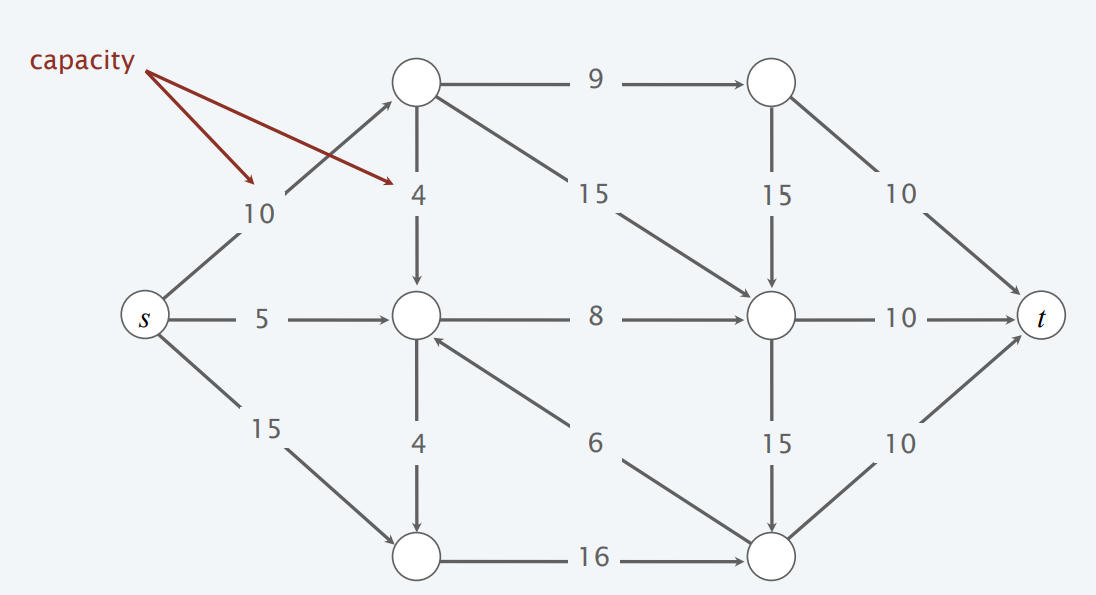
***Rete di flusso***

**Una rete di flusso è una tupla G = (V, E, s, t, c).**

**・Digrafo (V, E) con sorgente s ∈ V e pozzo t ∈ V.**

**・Capacità c(e) ≥ 0 per ogni e ∈ E.**

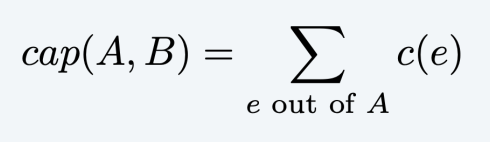
**Intuizione. Materiale che scorre attraverso una rete di trasporto; il materiale ha origine alla sorgente e viene inviato al pozzo.**

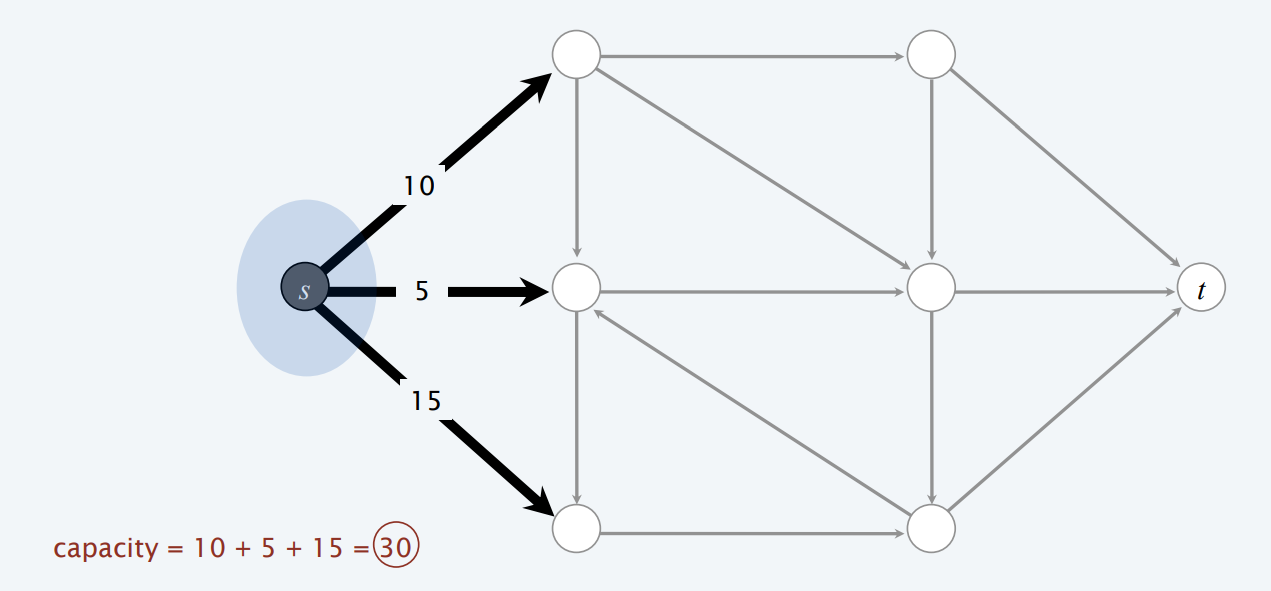
****

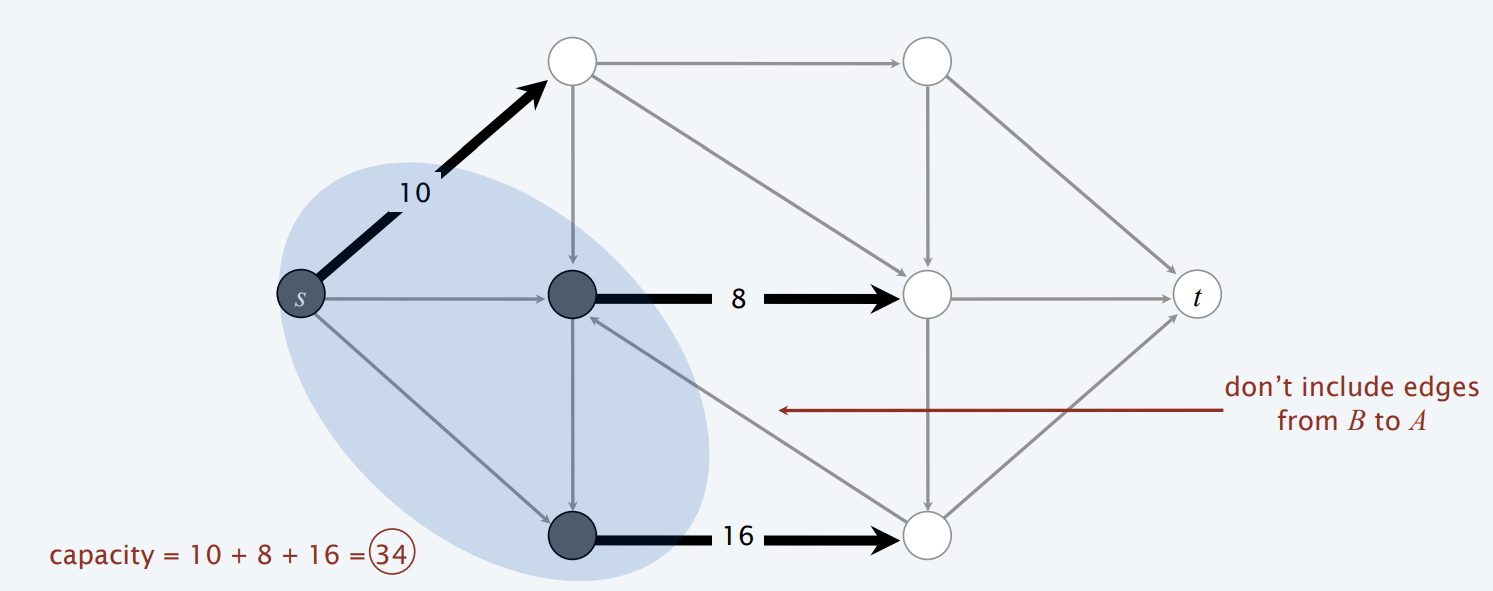
**Minimum-cut problem**

**Def. Un taglio (st-cut) è una partizione (A, B) dei nodi con s ∈ A e t ∈ B.**

**Def. La sua capacità è la somma delle capacità dei bordi da A a B.**

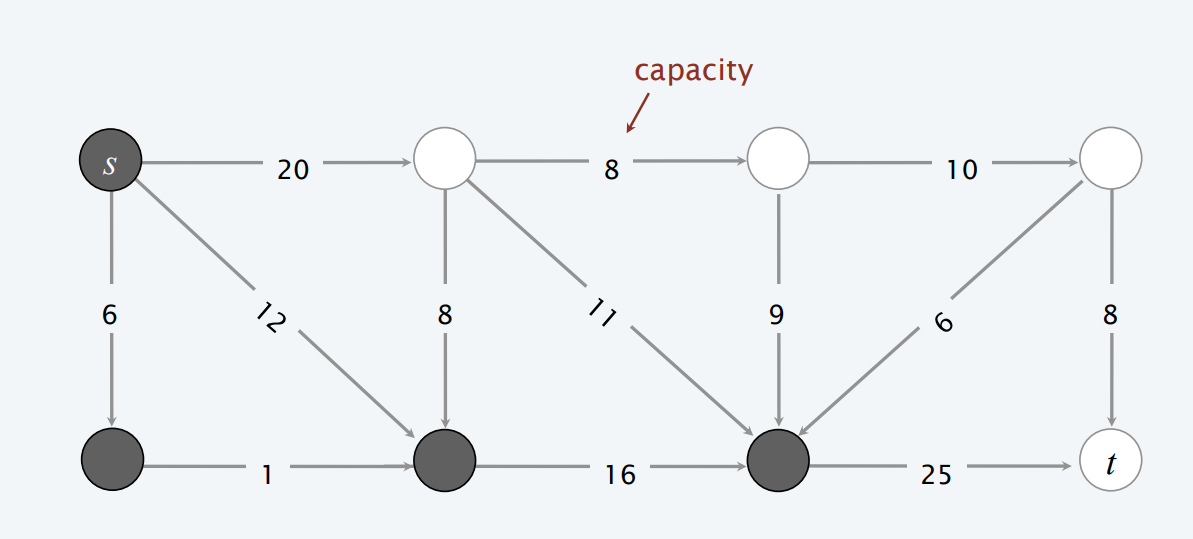
****

****

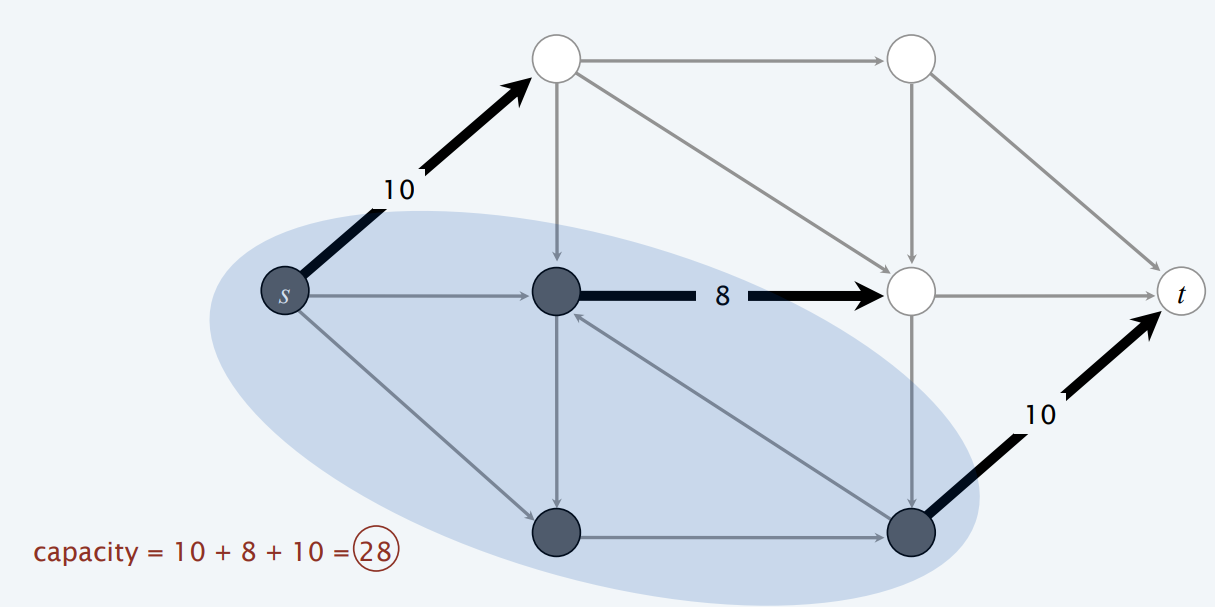
****

**ulteriore esempio:**

****

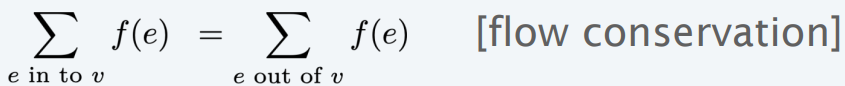
****

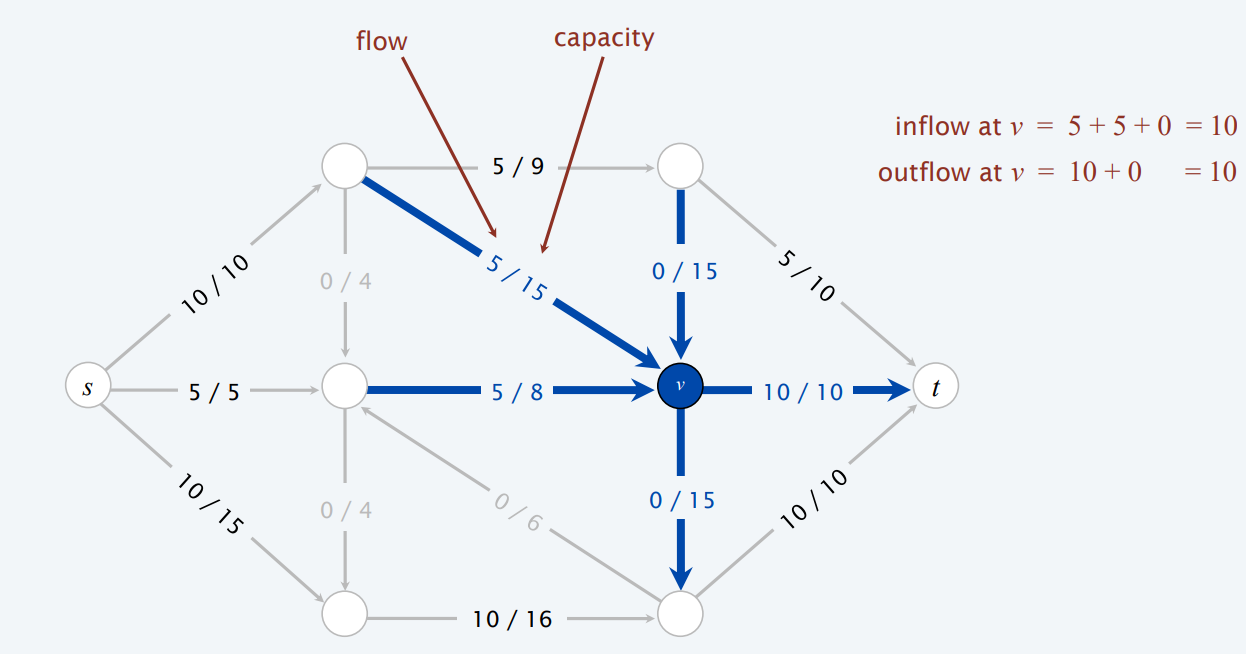
**Problema del taglio minimo. Trovare un taglio di capacità minima.**

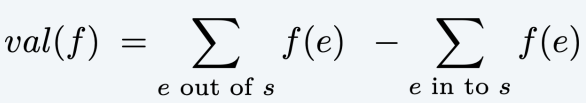
****

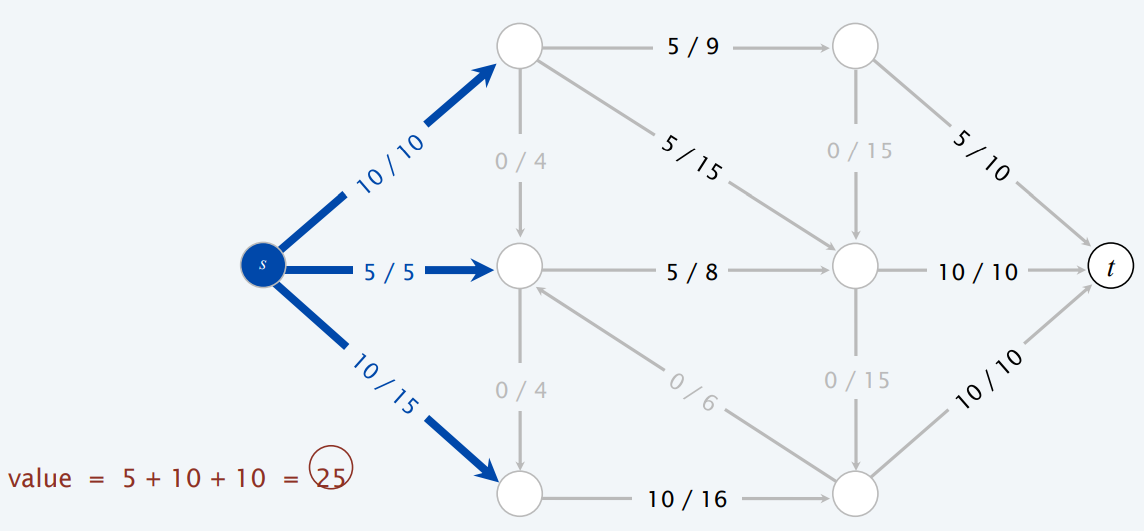
***Maximum-flow problem***

**Definizione. Un flusso st (flusso) f è una funzione che soddisfa:**

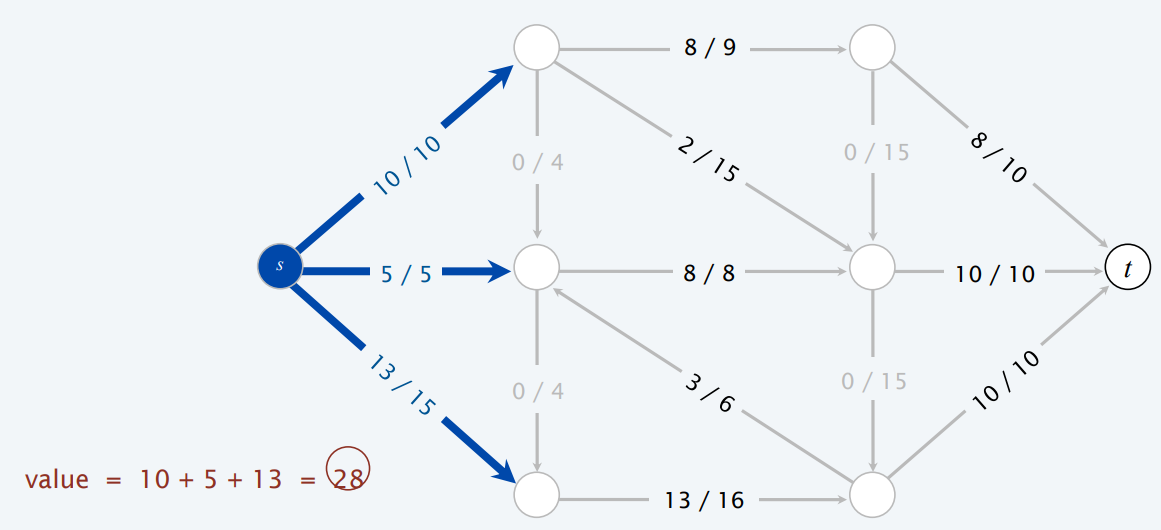
* **Per ogni e ∈ E : **
* **Per ogni v ∈ V - {s, t} :**

****

* **Il valore di un flusso f è:**

****

**Problema del flusso massimo. Trovare un flusso di valore massimo.**

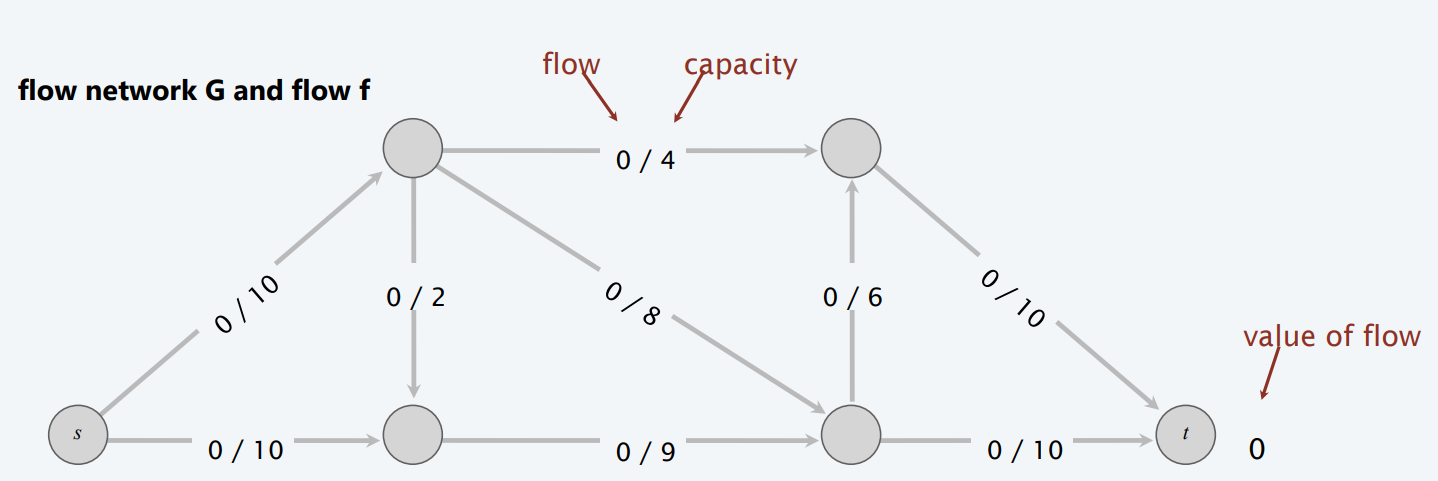
****

**‣ Ford–Fulkerson algorithm**

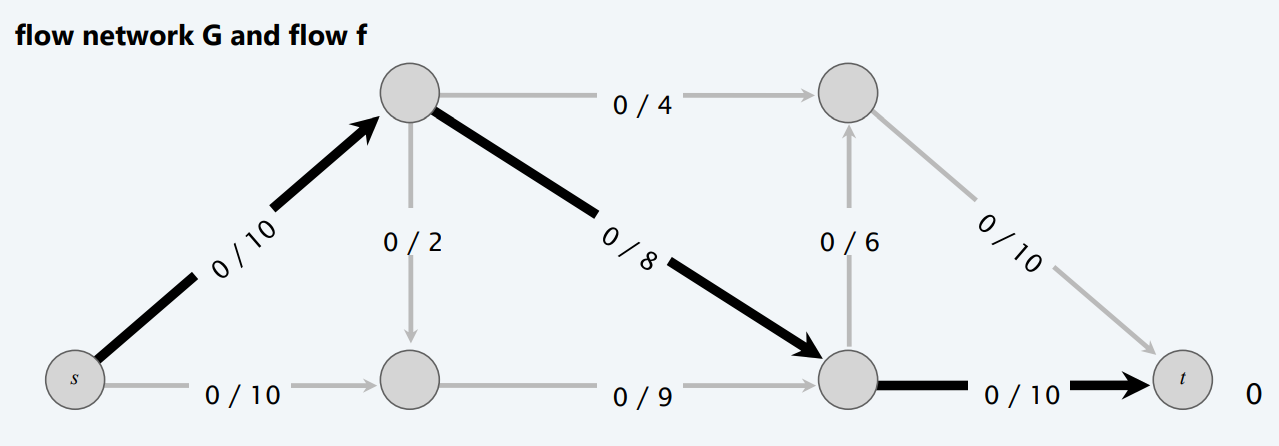
**Verso un algoritmo a flusso massimo.**

**Algoritmo greedy.**

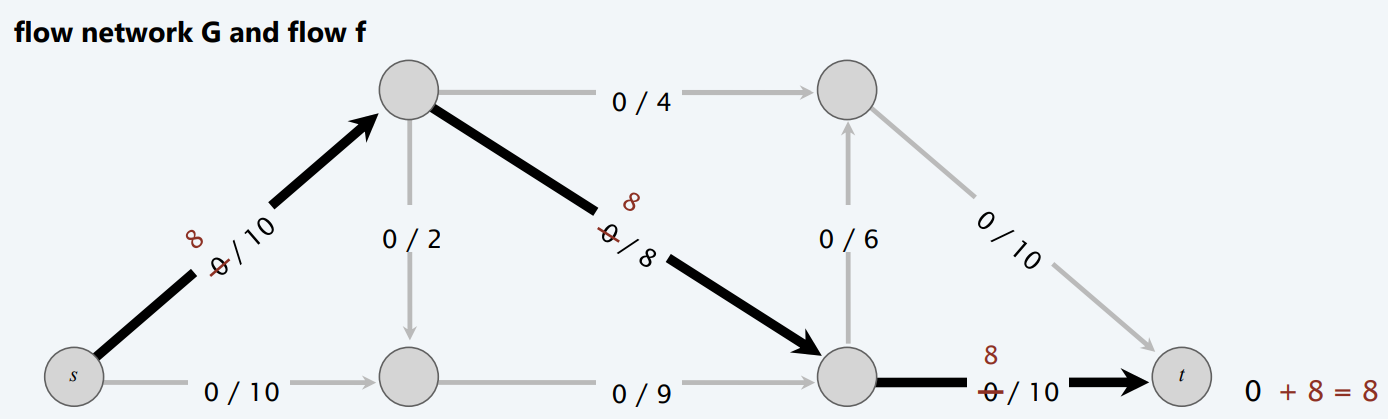
* **Inizia con f (e) = 0 per ogni arco e ∈ E.**

****

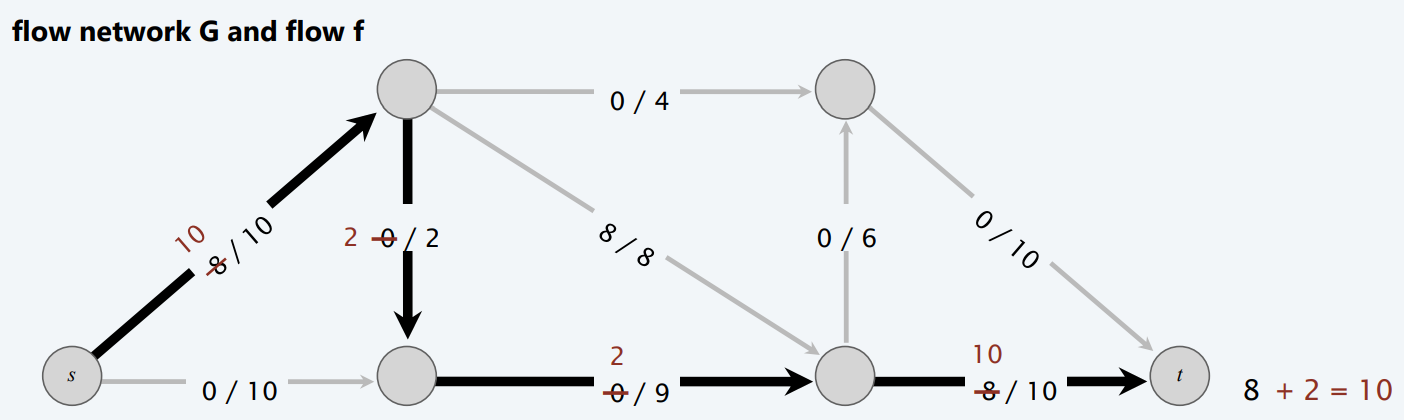
* **Trovare un percorso s↝t P in cui ogni arco ha f (e) < c(e).**

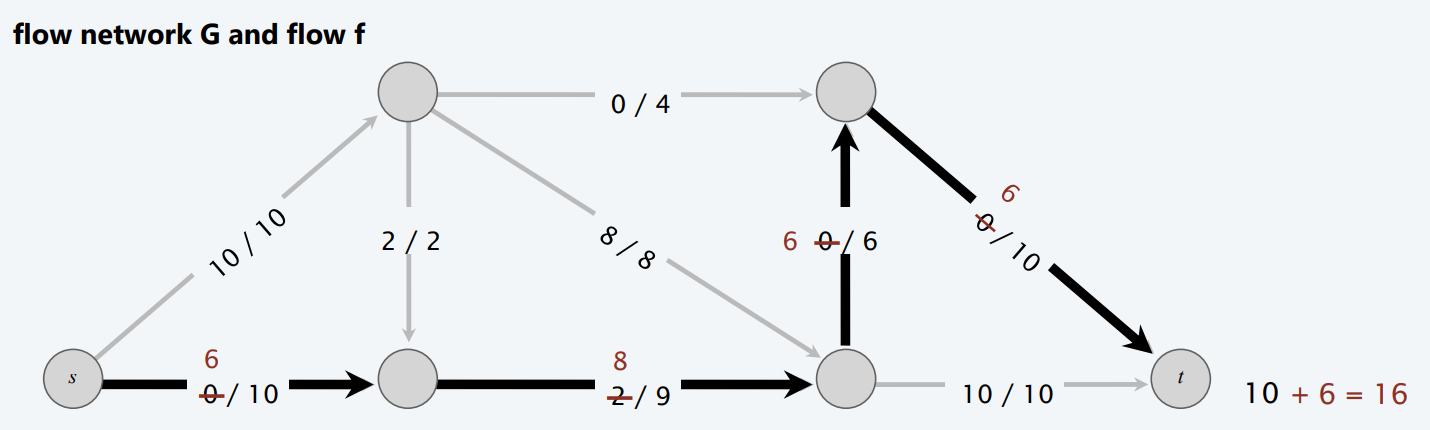
****

* **Aumentare il flusso lungo il percorso P.**

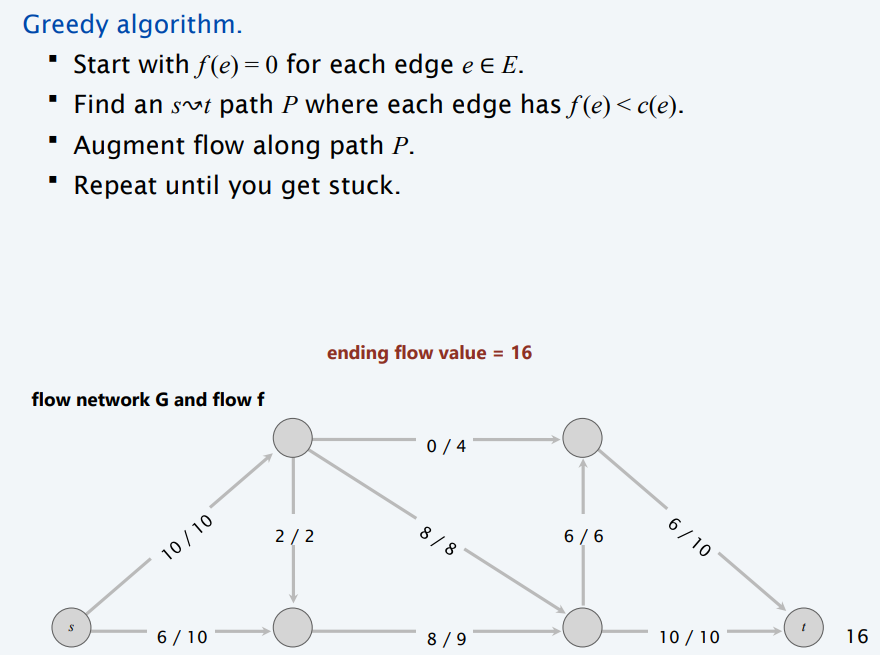
****

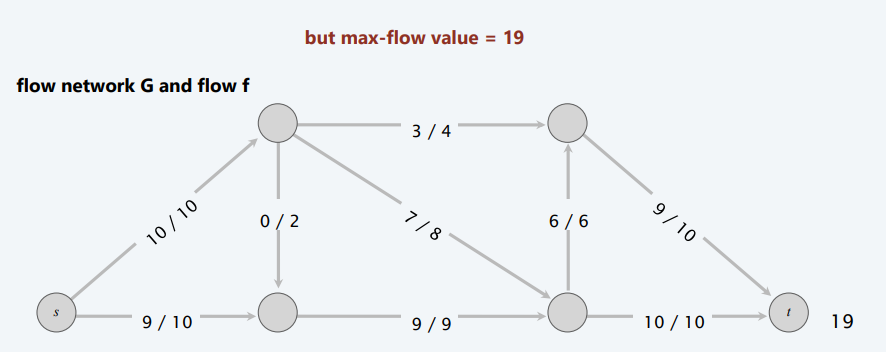
* **Ripetere fino a quando non ci si blocca.**

****

****

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

****

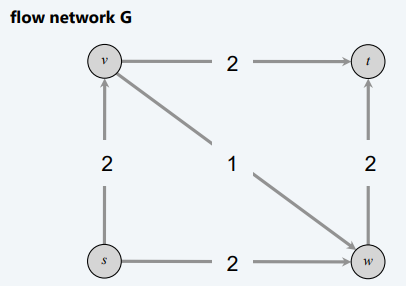
****

**Perché l'algoritmo greedy fallisce**

**D. Perché l'algoritmo greedy fallisce?**

**R. Una volta che l'algoritmo greedy aumenta il flusso su un bordo, non lo diminuisce mai.**

**Es. Consideriamo la rete di flussi G .**

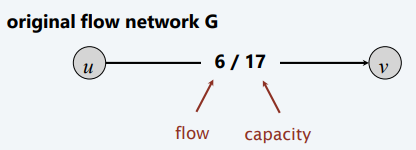
****

**L'unico flusso massimo f\* ha f\*(v, w) = 0.**

**L'algoritmo di tipo "greedy" potrebbe scegliere s→v→w→t come primo percorso.**

**Bottom Line. Serve un meccanismo per "annullare" una decisione sbagliata.**

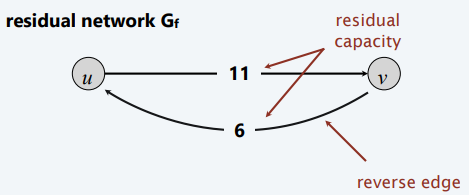
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Residual network**

**Arco originale. e = (u, v) ∈ E.**

**・Flusso f (e).**

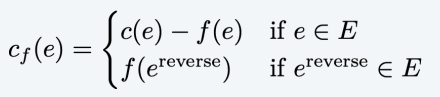
**・Capacità c(e).**



**Arco inverso. = (v, u).**

**Flusso "annullamento" inviato**

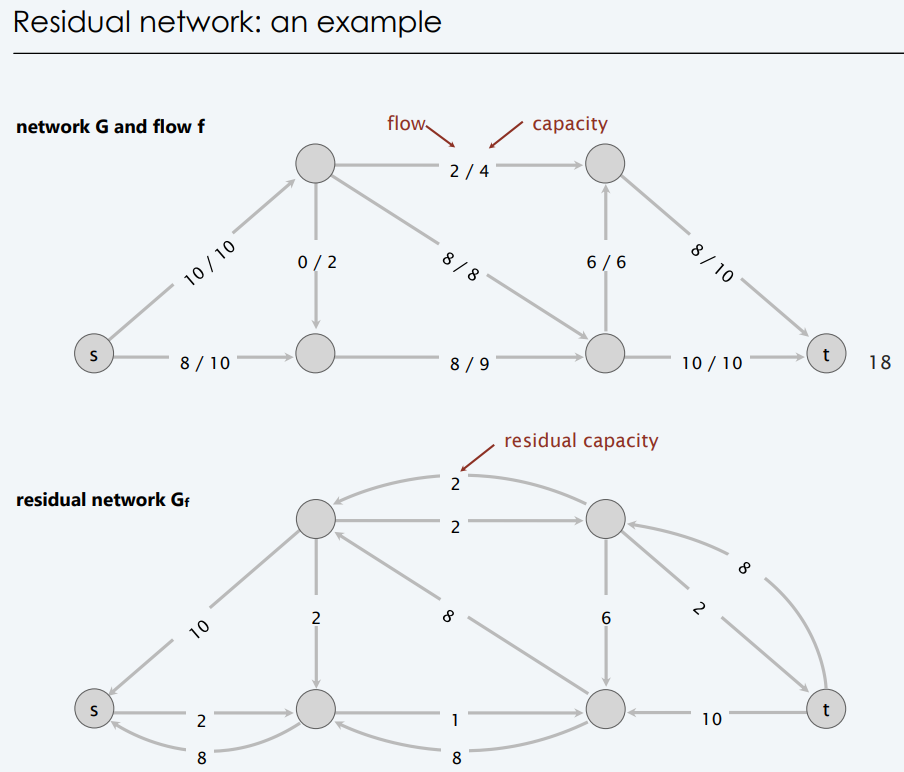
**Capacità residua.**

****

**Rete residua. Gf = (V, Ef, s, t, cf).**

**・Ef = {e : f (e) < c(e)} ∪ {e : f () > 0}.**

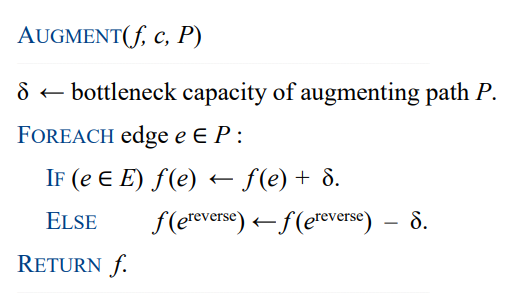
**Proprietà fondamentale: f ′ è un flusso in Gf iff f + f ′ è un flusso in G.**

****

***Percorso di ampliamento***

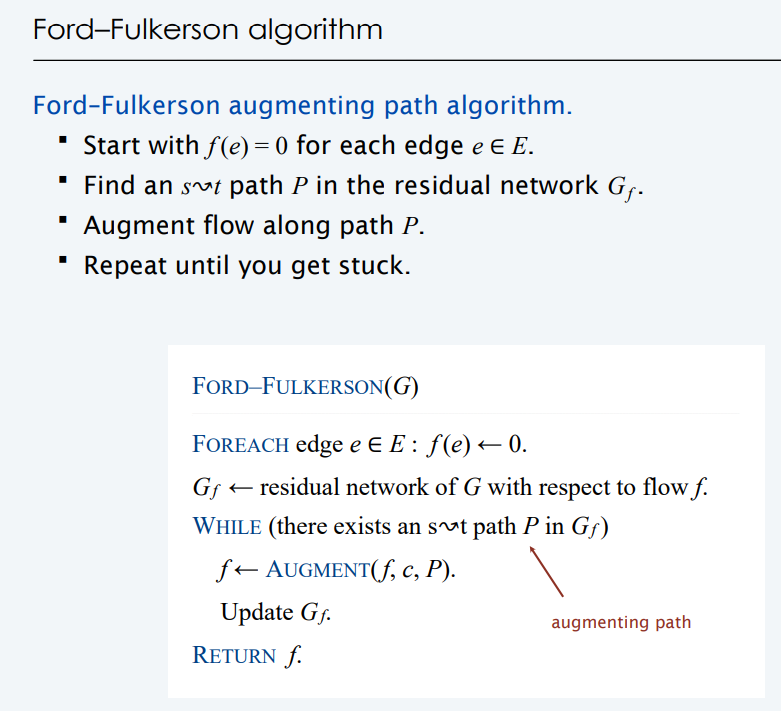
**Definizione. Un percorso di incremento è un percorso semplice s↝t nella rete residua Gf.**

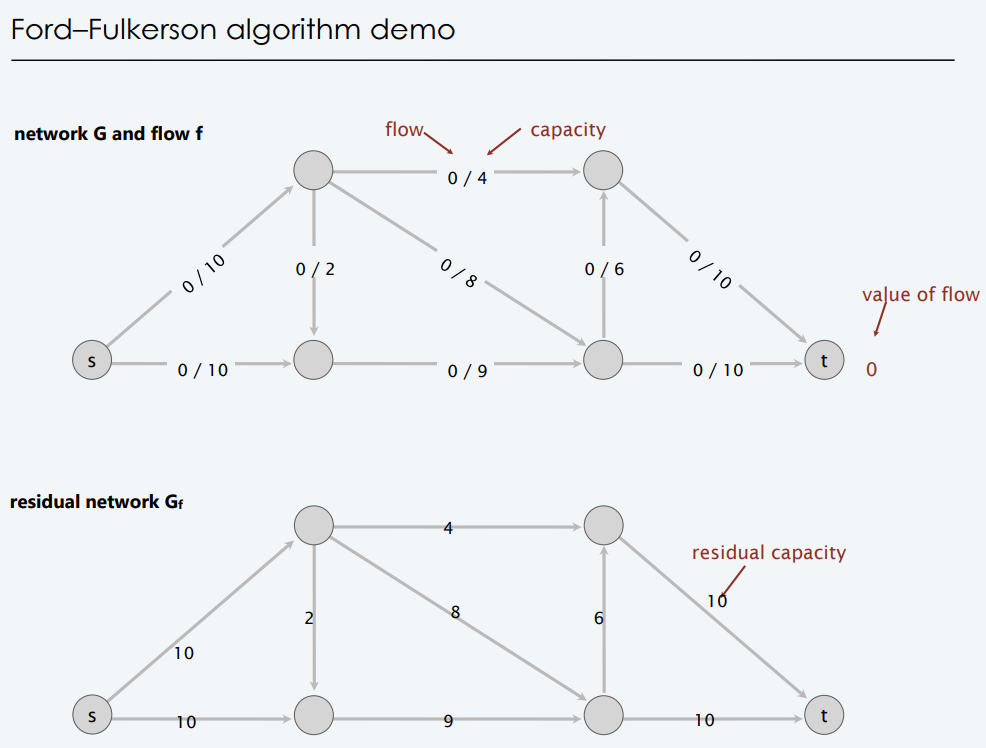
**Def. La capacità del collo di bottiglia (bottleneck) di un cammino accrescitivo P è la capacità residua minima di qualsiasi arco in P.**

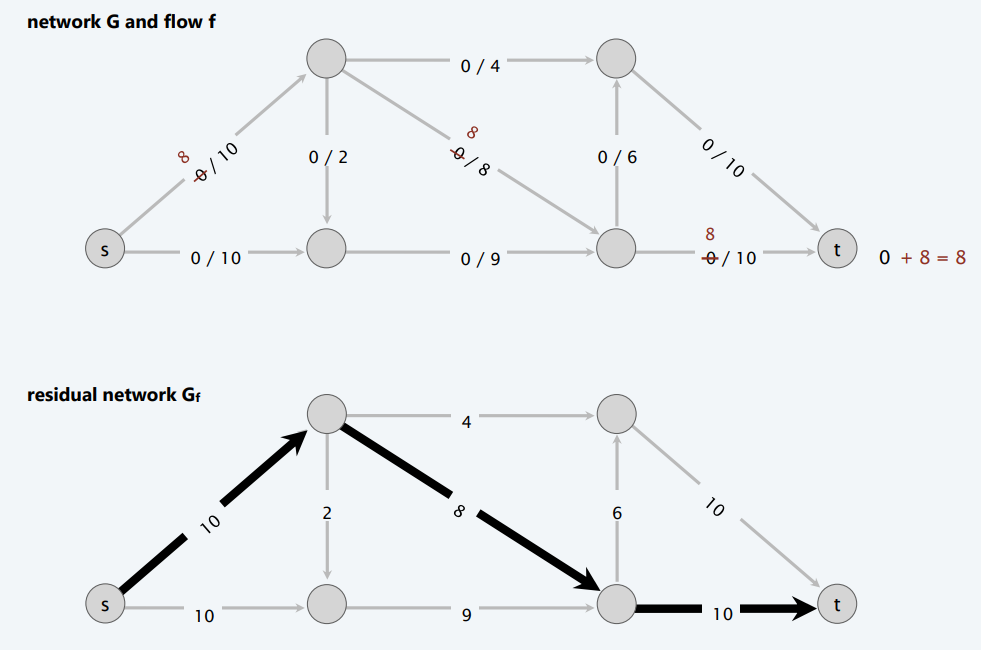
****

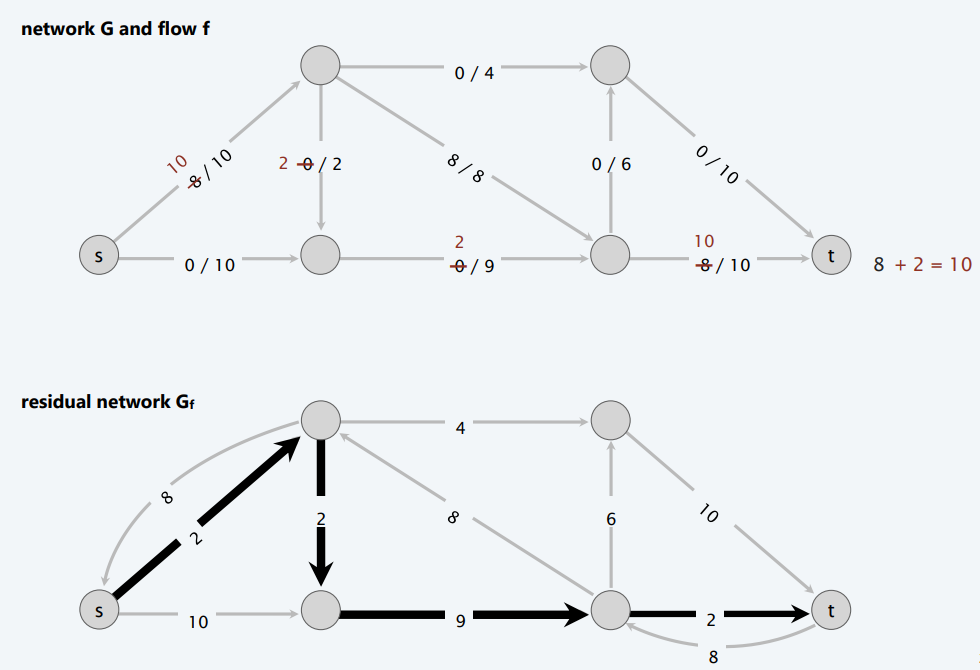
**Proprietà fondamentale. Sia f un flusso e sia P un cammino accrescitivo in Gf .**

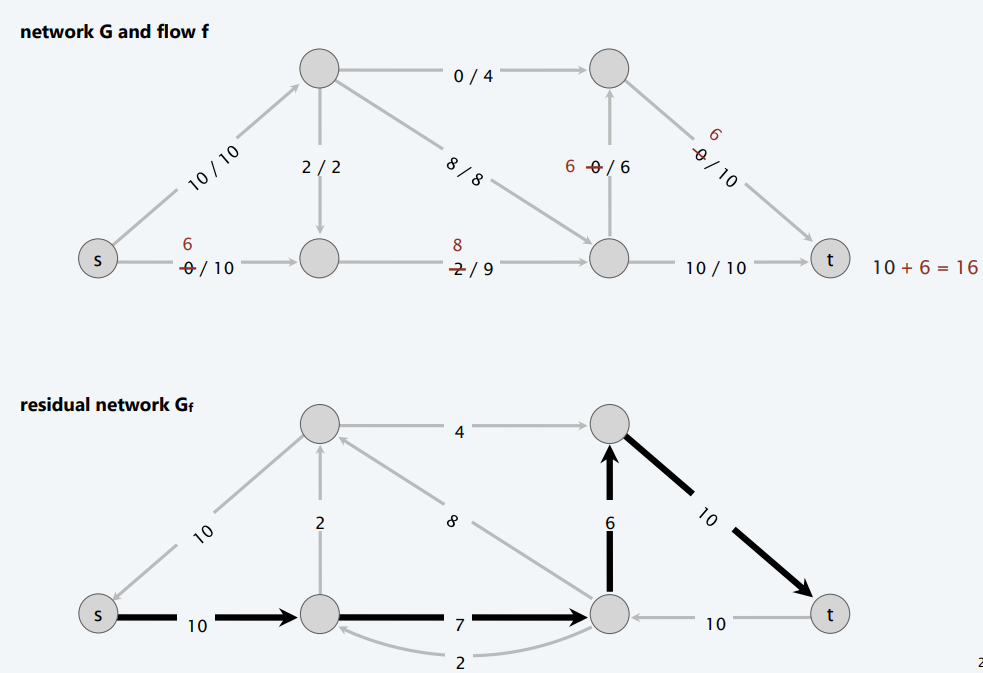
**Allora, dopo aver chiamato f ′ ← AUGMENT(f, c, P), la risultante f ′ è un flusso e val(f ′) = val(f ) + bottleneck(Gf, P).**

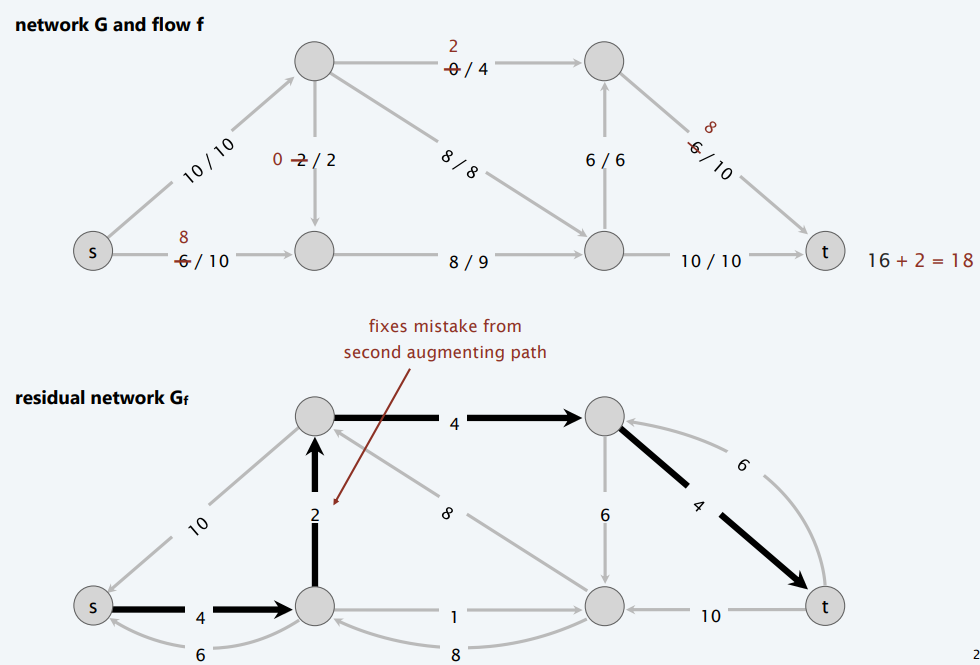
****

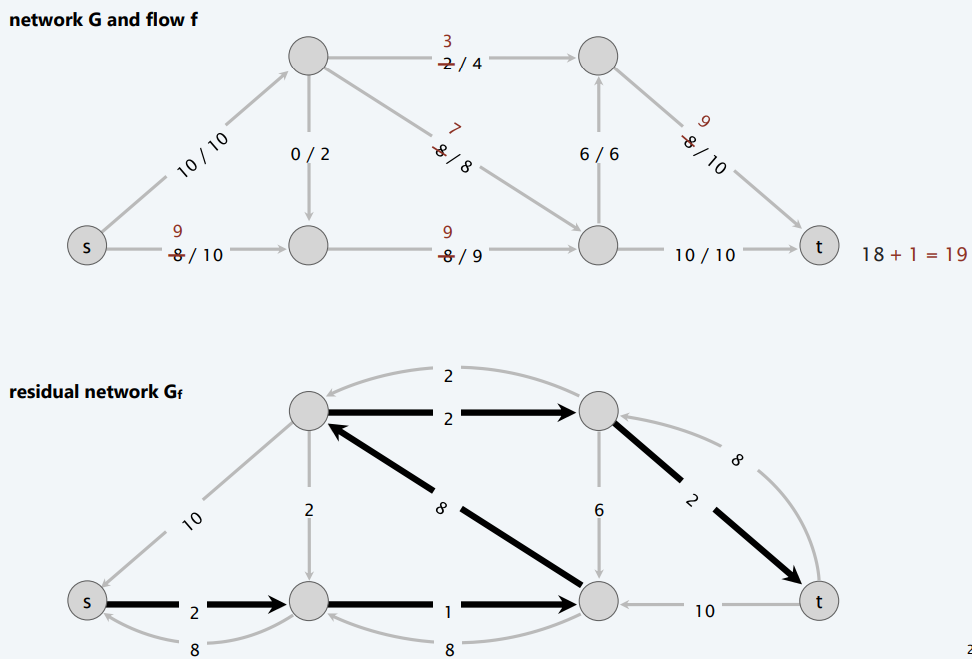
****

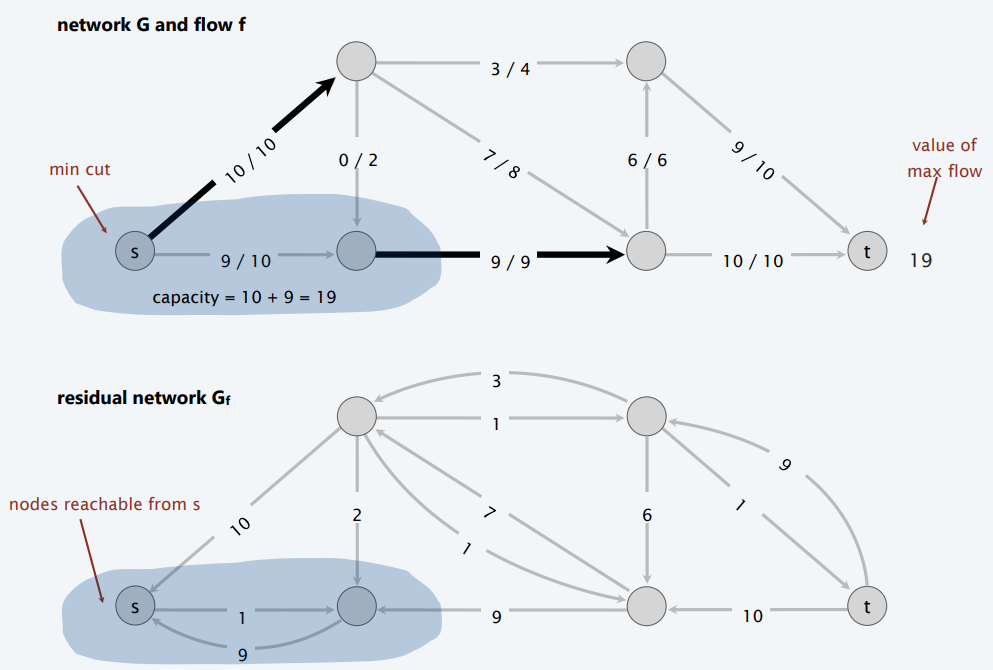
****

****

****

****

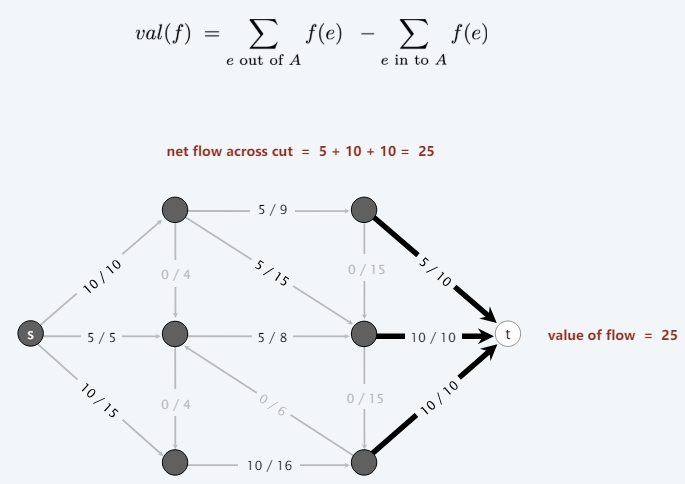
****

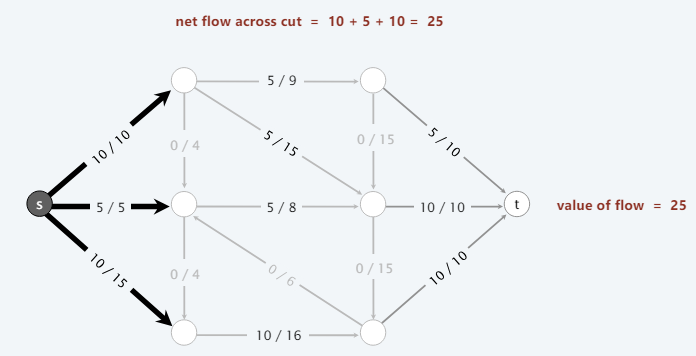
****

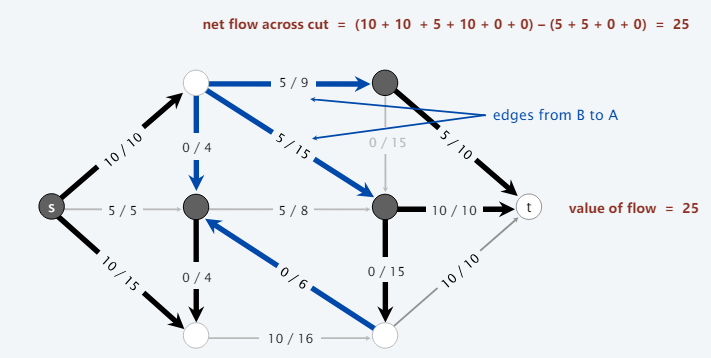
**‣ max-flow min-cut theorem**

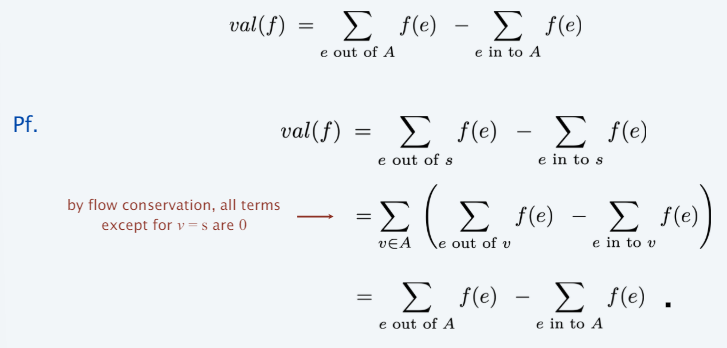
***Relazione tra flussi e tagli***

**Lemma del valore del flusso. Sia f un flusso qualsiasi e sia (A, B) un taglio qualsiasi. Allora, il valore del flusso f è uguale al flusso netto attraverso il taglio (A, B)**

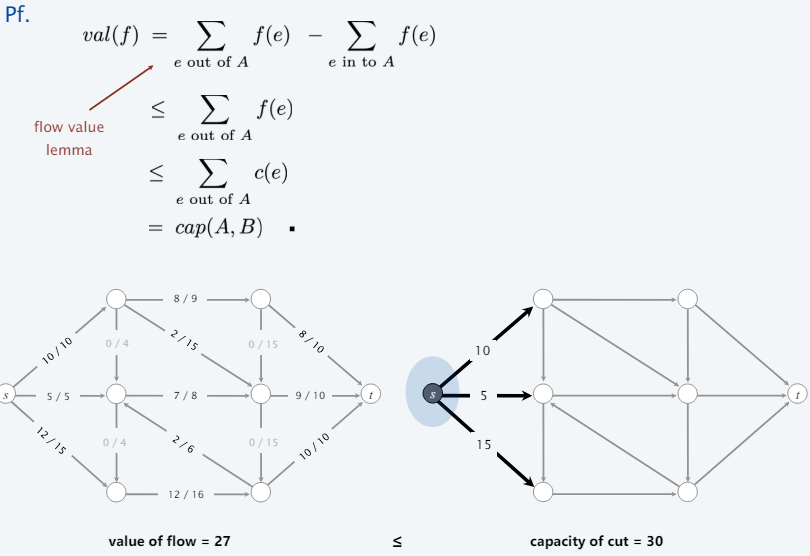
****

****

****

****

**Dualità debole. Sia f un flusso qualsiasi e (A, B) un taglio qualsiasi. Allora val(f) ≤ cap(A, B).**

****

***Certificato di ottimalità***

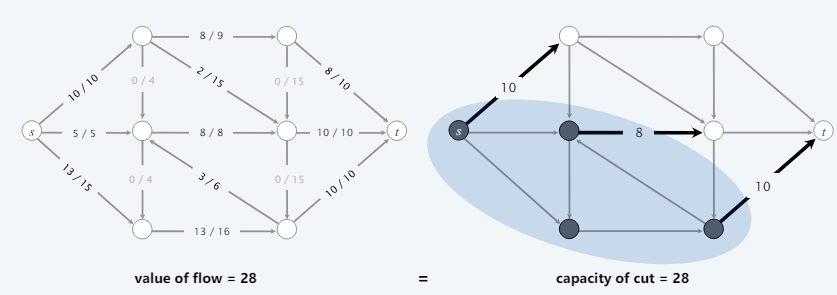
**Corollario. Sia f un flusso e sia (A, B) un taglio qualsiasi.**

**Se val(f) = cap(A, B), allora f è un flusso massimo e (A, B) è un taglio minimo.**

**Pf.**

**・Per qualsiasi flusso f ′: val(f ′) ≤ cap(A, B) = val(f).**

**・Per qualsiasi taglio (A′, B′): cap(A′, B′) ≥ val(f) = cap(A, B). ▪**

****

***Teorema del taglio minimo del flusso massimo***

**Teorema del flusso massimo e del taglio minimo. Valore di un flusso massimo = capacità di un taglio minimo.**

**Teorema dei percorsi di incremento. Un flusso f è un flusso massimo se non ha percorsi di incremento.**

**Pf.**

**Le tre condizioni seguenti sono equivalenti per qualsiasi flusso f :**

**i. Esiste un taglio (A, B) tale che cap(A, B) = val(f).**

**ii. f è un flusso massimo.**

**iii. Non esiste un percorso accrescitivo rispetto a f.**

**[ i ⇒ ii ]**

**・Questo è il corollario di dualità debole. ▪**

**[ ii ⇒ iii ] Dimostriamo il controsenso: ¬ iii ⇒ ¬ ii.**

**・Supponiamo che esista un percorso aumentativo rispetto a f.**

**・Può migliorare il flusso f inviando il flusso lungo questo percorso.**

**・Quindi, f non è un flusso massimo. ▪**

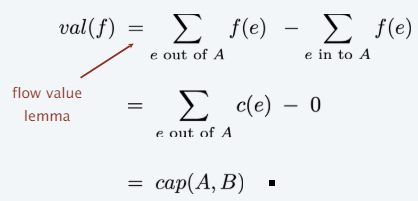
**[ iii ⇒ i ]**

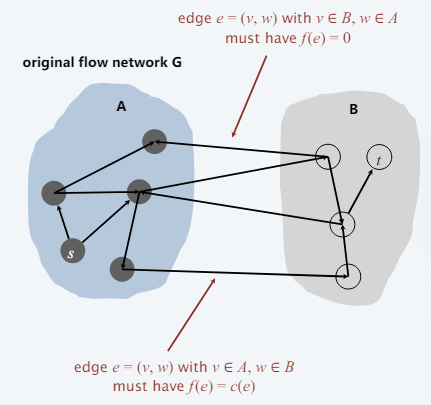
**・Lasciare che f sia un flusso senza percorsi di incremento.**

**・Sia A = insieme dei nodi raggiungibili da s nella rete residua Gf.**

**・Per definizione di A: s ∈ A.**

**・Per definizione del flusso f: t ∉ A.**

****

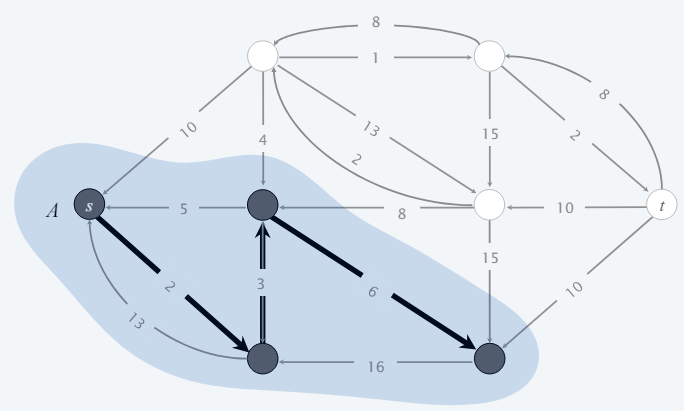
****

***Calcolo di un taglio minimo da un flusso massimo***

**Teorema. Dato un qualsiasi flusso massimo f , è possibile calcolare un taglio minimo (A, B) in tempo O(m).**

**Pf.**

**Sia A = insieme dei nodi raggiungibili da s nella rete residua Gf . ▪**

****

***Analisi dell'algoritmo di Ford-Fulkerson (per capacità integrali)***

**Assunzione. Ogni capacità di bordo c(e) è un intero compreso tra 1 e C.**

**Invariante di integrità. In tutto l'algoritmo di Ford-Fulkerson, ogni flusso di bordo f (e) e capacità residua cf (e) è un intero.**

**Pf. Per induzione sul numero di percorsi di incremento. ▪**

**Teorema. Ford-Fulkerson termina dopo al massimo val(f \*) ≤ nC percorsi di incremento, dove f \* è un flusso massimo.**

**Pf. Ogni incremento aumenta il valore del flusso di almeno 1. ▪**

**Corollario. Il tempo di esecuzione di Ford-Fulkerson è O(m val(f \*))=O(m nC).**

**Pf. Si può usare BFS o DFS per trovare un percorso di incremento in tempo O(m). ▪**

**Teorema di integrità. Esiste un flusso massimo integrale f\*.**

**Pf. Poiché Ford-Fulkerson termina, il teorema segue dall'invariante dell'integrità (e dal teorema del percorso di incremento).**

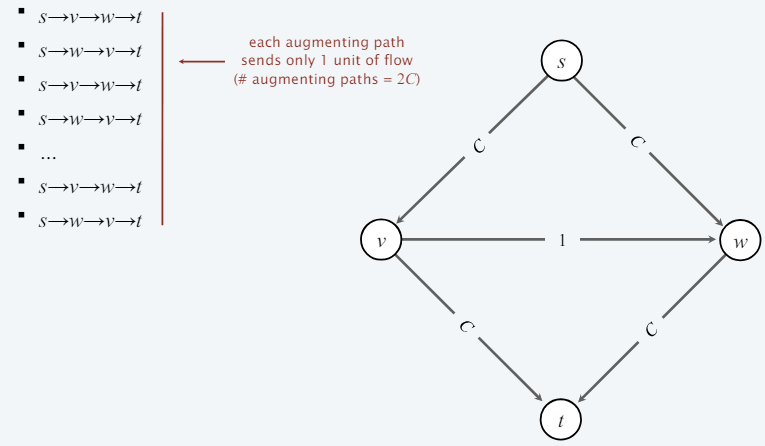
**(e dal teorema del cammino di accrescimento). ▪**

***Ford-Fulkerson: esempio esponenziale***

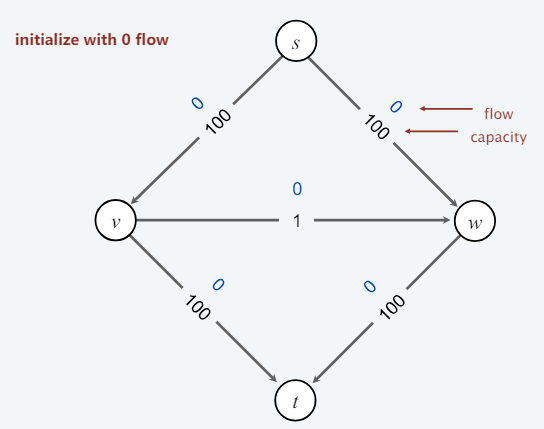
**Q. L'algoritmo generico di Ford-Fulkerson è poli-tempo nella dimensione dell'input?**

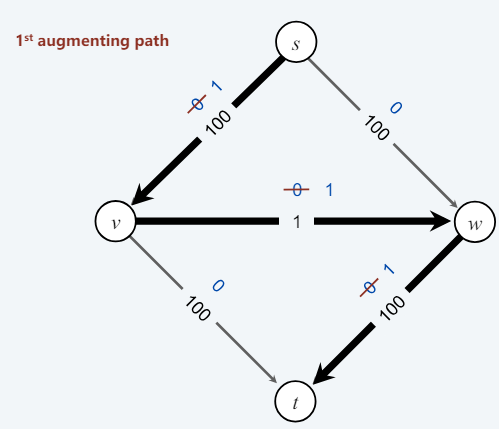
**A. No. È pseudo-polinomiale.**

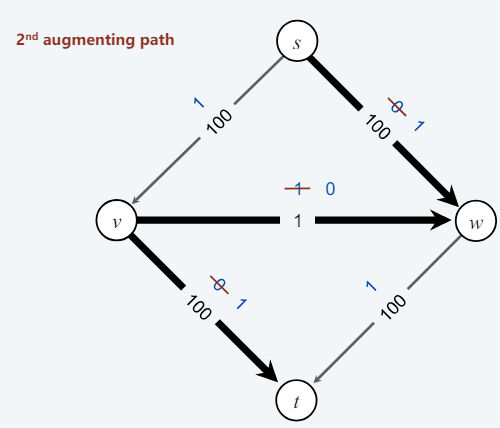
**Se la capacità massima è C, l'algoritmo può richiedere ≥ C iterazioni.**

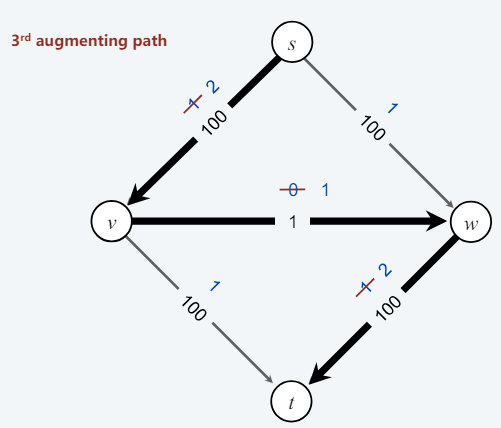
****

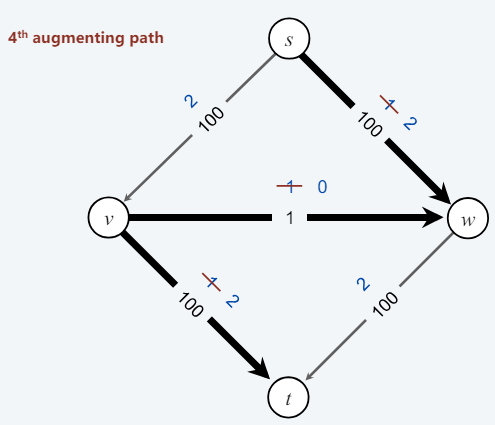
**Cattive notizie. Il numero di percorsi di incremento può essere esponenziale rispetto alle dimensioni dell'input.**

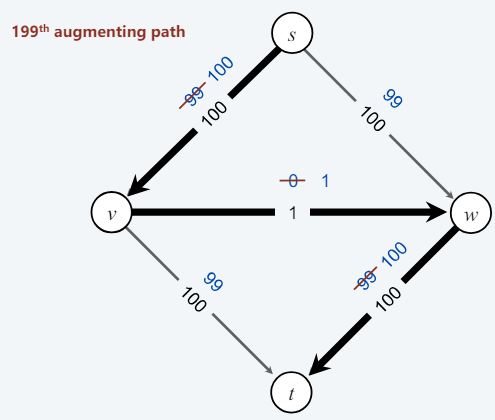
****

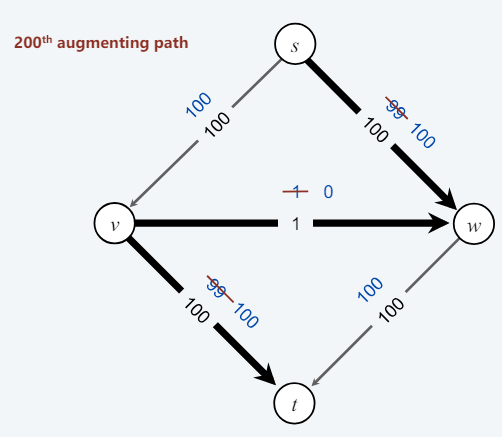
****

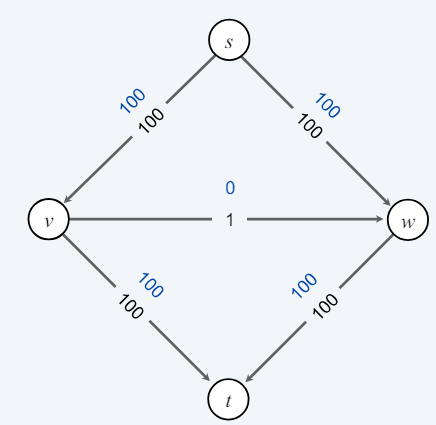
****

****

****

****

****

****

**‣ choosing good augmenting paths**

***Scegliere buoni percorsi di accrescimento***

**Prestare attenzione nella scelta dei percorsi di incremento.**

**・Alcune scelte portano ad algoritmi esponenziali.**

**・Le scelte più oculate portano ad algoritmi polinomiali.**

**Patologia. Quando le capacità dei bordi possono essere irrazionali, non è garantito che Ford-Fulkerson termini (o converga a un flusso massimo)!**

**Obiettivo. Scegliere percorsi di incremento in modo che:**

**・possa trovare percorsi di incremento in modo efficiente.**

**・Poche iterazioni.**

**Scegliere percorsi di incremento con:**

**・Capacità del collo di bottiglia sufficientemente grande.**

**・Meno archi.**

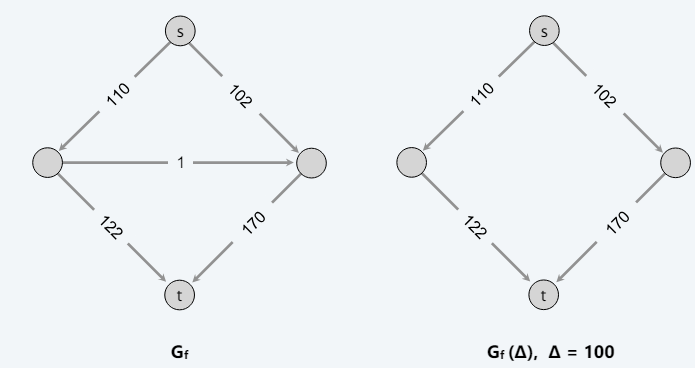
***Algoritmo di scalatura della capacità***

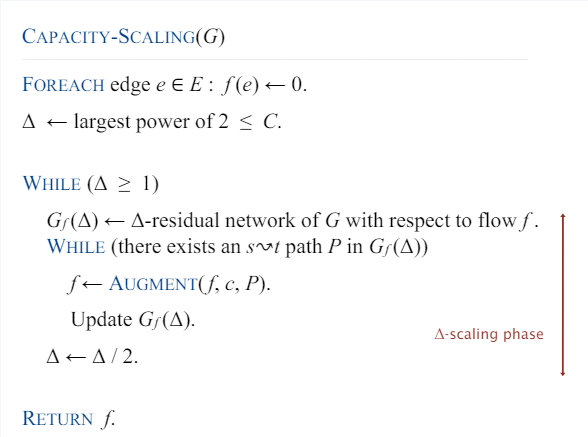
**Panoramica. Scelta di percorsi di incremento con capacità di collo di bottiglia “grande”.**

**・Mantenere il parametro di scala Δ.**

**・Lasciare che Gf (Δ) sia la parte della rete residua contenente solo i bordi con capacità ≥ Δ.**

**Ogni percorso di incremento in Gf (Δ) ha capacità di collo di bottiglia ≥ Δ.**

****

****

**Algoritmo di scalata della capacità: analisi del tempo di esecuzione (sketch)**

**Si può dimostrare quanto segue:**

**Lemma 1. Esistono 1 + ⎣log2 C⎦ fasi di scalatura.**

**Lemma 2. Ci sono ≤ 2m incrementi per fase di scaling.**

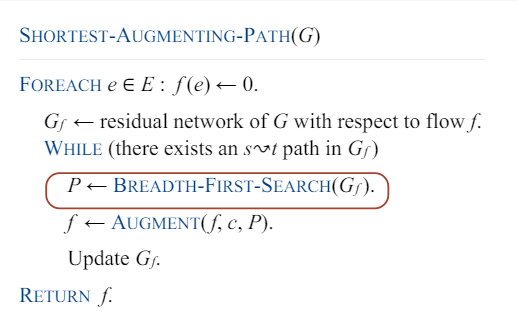
**→ Numero totale di incrementi: O(m log C)**

**Teorema. L'algoritmo di scalatura della capacità richiede un tempo O(log C).**

***Il percorso di incremento più breve***

**Q. Come scegliere il prossimo percorso di incremento in Ford-Fulkerson?**

**A. Scegliere quello che utilizza il minor numero di bordi( BFS ).**

****

**Percorso incrementale più breve: tempo di esecuzione**

**Si può dimostrare quanto segue:**

**Lemma 1. Il numero totale di incrementi è al massimo m n.**

**Teorema. L'algoritmo del percorso più breve richiede un tempo O(n).**