



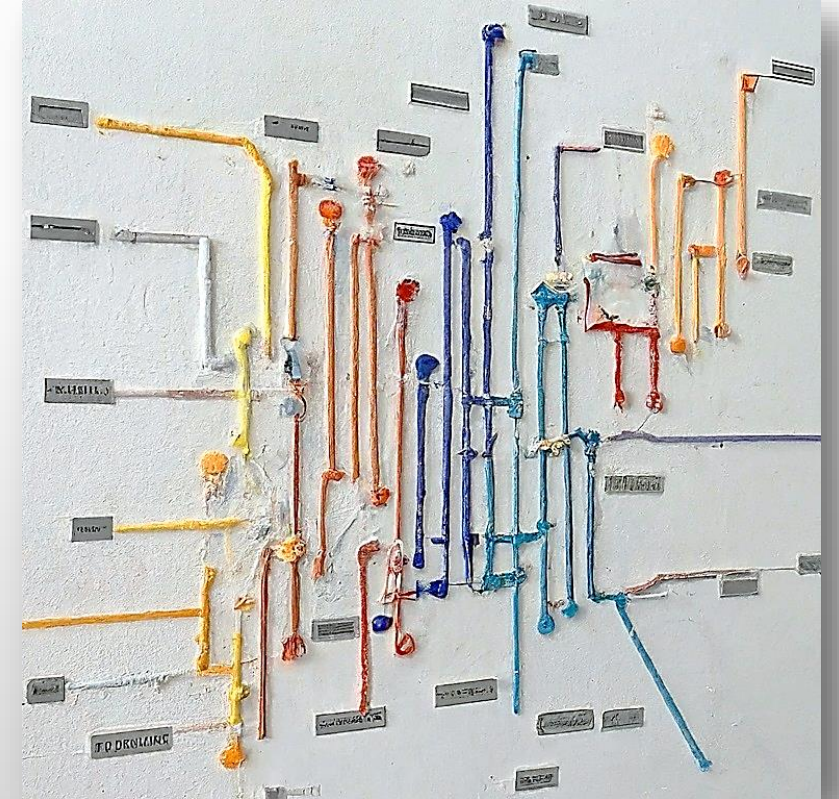
Bölüm 4: Çizge Algoritmaları

Algoritmalar



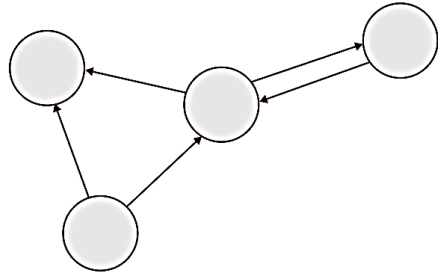
Çizge Algoritmaları

- Dünya aslında bir ağ gibidir.
 - Şehirler yollarla,
 - İnsanlar ilişkilerle,
 - Bilgisayarlar kablolarla birbirine bağlıdır.
- Çizge algoritmaları bu ağları inceler ve anlamlandırır.

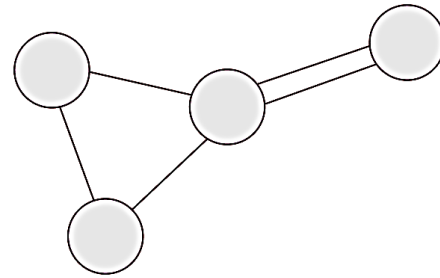




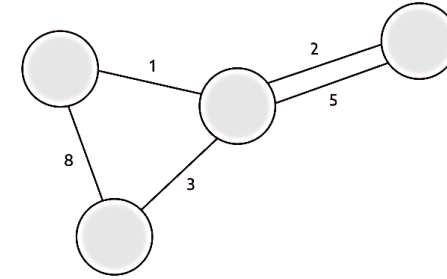
Çizge Türleri



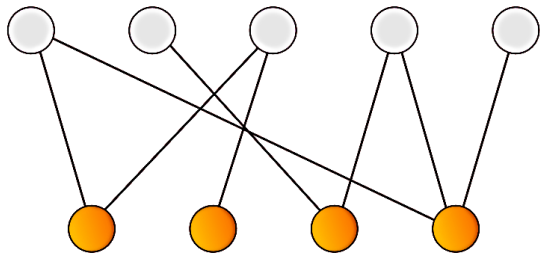
Directed graph



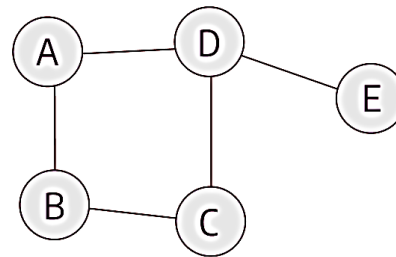
Undirected



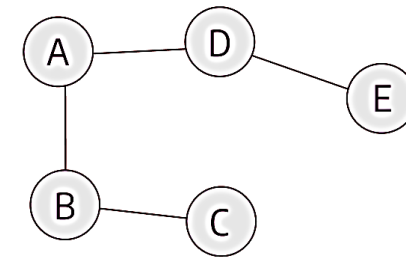
Weighted



Bipartite graph



Cyclic graph



Acyclic graph



Çizge Algoritmaları

- Birbirine bağlı noktalar (düğüm) ve bu noktaları birleştiren çizgiler (kenar) ile temsil edilen ağ yapılarını inceler.
- Ağlarda en kısa yolu hesaplama, gruplama gibi işlemleri gerçekleştirir.
- Sosyal ağlar, harita uygulamaları, navigasyon gibi birçok alanda kullanılır.



Çizge Algoritmalarının Çeşitleri

- Farklı çizge algoritmaları, farklı işlemler için kullanılır.
- Derinlik Öncelikli Arama (DFS):
 - Bir düğümden başlar, dallanarak tüm ağı gezer.
- Genişlik Öncelikli Arama (BFS):
 - Bir düğümden başlar, katman katman tüm ağı gezer.
- Dijkstra Algoritması:
 - Başlangıç düğümünden diğer düğümlere en kısa yolları bulur.
- Kruskal Algoritması:
 - Bir ağı minimum maliyetle birbirine bağlayan kenarları seçer.



Çizge Algoritmaları

- DFS bir labirentten çıkış yolu ararken kullanılabilir.
- BFS bir haberin tüm şehire yayılma sürecini modelleyebilir.
- Dijkstra en kısa sürede teslimat yapmak için kullanılabilir.





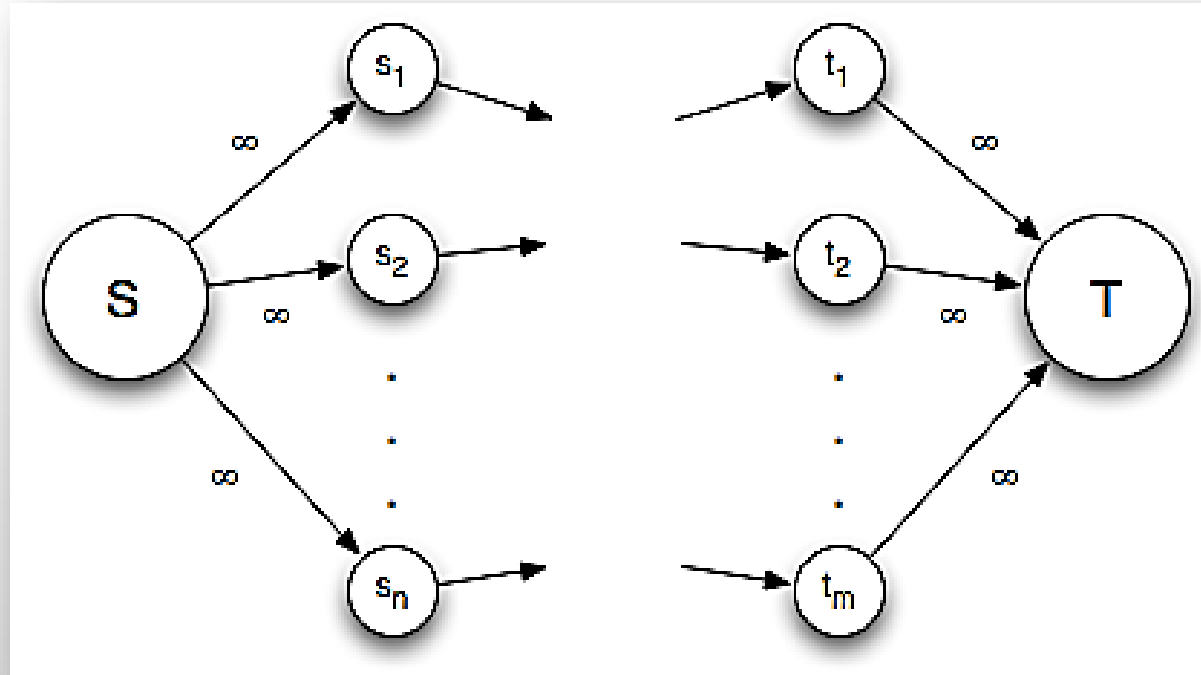
Çizge Algoritmaları

- Çizge gezinme algoritmaları (*Graph traversal*)
- En kısa yol algoritmaları (*Shortest path*)
- Minimum yayılan ağaç algoritmaları (*Minimum spanning tree*)
- Ağ akış algoritmaları (*Network flow*)



Maksimum Akış (Maximum Flow)

- Çizge içinde bir kaynak (source) ve bir hedef (sink) düğüm bulunur.
- Çizge üzerindeki kenarlara kapasiteler atanır.
- Kaynaktan hedefe maksimum miktarda akışı bulmak amaçlanır.





Artık Ağlar (Residual Networks)

- Akışın hedefe ulaşması için kullanılan kapasiteyi gösterir.
- Bir kenarın kapasitesinden akışın miktarı çıkarılarak bulunur.
- Eğer kenarda daha az akış varsa, kalan kapasite pozitif olur.
- Çizge üzerinde artış yollarını gösterir.



Artış Yolları (Augmenting Paths)

- Çizge üzerinde kaynaktan hedefe ek akış sağlayan yolları ifade eder.
- Bir artış yolu, kaynaktan hedefe yönlendirilen bir yol olmalıdır.
- Çizge üzerindeki kenarların kapasitelerinden daha az akış taşınmalıdır.
- Maksimum akışa ulaşmak için kullanılır.
- Genellikle BFS veya DFS algoritmaları kullanılarak bulunur.



Kesimler (Cuts)

- Bir çizgenin düğümlerini ikiye bölen bir kenar kümesidir.
- Çizgenin bağlantısını keser ve farklı bileşenlere ayırır.
- Min-cut (Minimum kesme): En az sayıda kenarı kesecek düğüm kümesi.
- Max-flow (Maksimum akış): Çizgeyi kaynak ve hedef arasında böler.



Ağ Akış Algoritmaları (Network Flow)

- Kaynaktan hedefe kısıtlamalar altında ağ akışını optimize eder.
- Amaç, kısıtlamalar altında mümkün olan en fazla akışı sağlamaktır.
- *Ford-Fulkerson*,
 - basit ve anlaşılır, çalışma süresi diğerlerine göre daha uzun.
- *Edmonds-Karp*,
 - *Ford-Fulkerson* algoritmasının gelişmiş hali, çalışma süresi daha kısa.
- *Dinic's*,
 - *Edmonds-Karp* algoritmasından hızlı, bazı durumlarda daha verimli.



Ford Fulkerson

- Ağırlıklı yönlü çizgede iki düğüm arasındaki maksimum akışı bulur.
- Maksimum akış problemi, kaynaktan hedefe belirli kapasiteye sahip yollarla maksimum suyun akışını modelleyen bir çizge problemidir.
- L.R. Ford Jr. ve D.R. Fulkerson tarafından geliştirilmiştir.
- *Ford-Fulkerson uses the DFS, Edmonds-Karp uses the BFS approach.*



Algoritma İlkeleri

- Ağ, yönlü bir çizge olarak temsil edilir.
- Her kenara bir kapasite değeri atanır.
- Başlangıçta, tüm akışlar sıfır olarak başlatılır.
- Artan yol (*augmenting path*) bulma adımları tekrarlanarak,
 - maksimum akış bulunur.



Algoritma Adımları

- Adım 1: Kaynaktan hedefe artan bir yol bulunur.
- Adım 2: Bulunan artan yol boyunca maksimum akışa izin verilir. Bu, akış ağındaki tüm kenarlarda artışa neden olur.
- Adım 3: Hedefe ulaşılan kadar Adım 1 ve Adım 2 tekrarlanır.



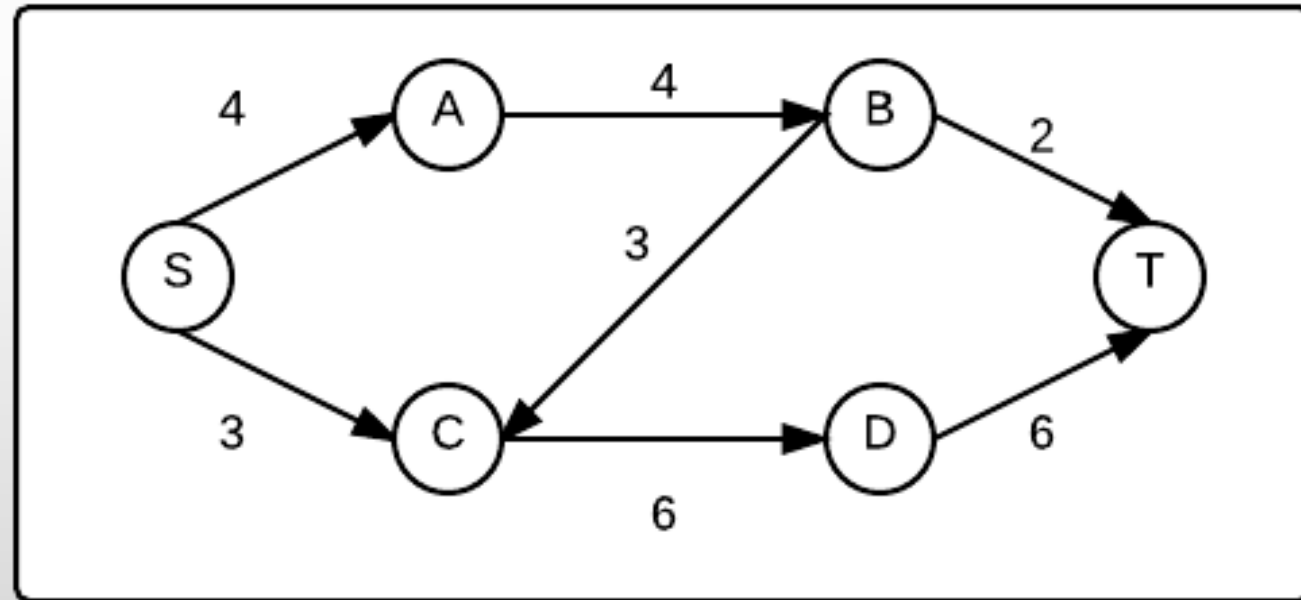
Karmaşıklık Analizi

- Akış ağının yapısına ve kapasitelerine bağlı olarak değişir.
- En kötü durumda, $O(E f)$ veya $O(V E^2)$ karmaşıklığına sahiptir.
 - E kenar sayısını, V düğüm sayısını, f maksimum akışı temsil eder.
- $O(E f)$, bir artış yolu bulunduğunda, artış yolundaki en düşük kapasiteli kenarı bulmak $O(E)$. Kenar boyunca akışı artırmak için bir kez daha $O(f)$.
- $O(V E^2)$, her artış yolu bulma aşamasında DFS kullanılır. Artış yolu üzerinde en düşük kapasiteli kenarı bulmak $O(E)$. V kez DFS yapılır.



Ford Fulkerson

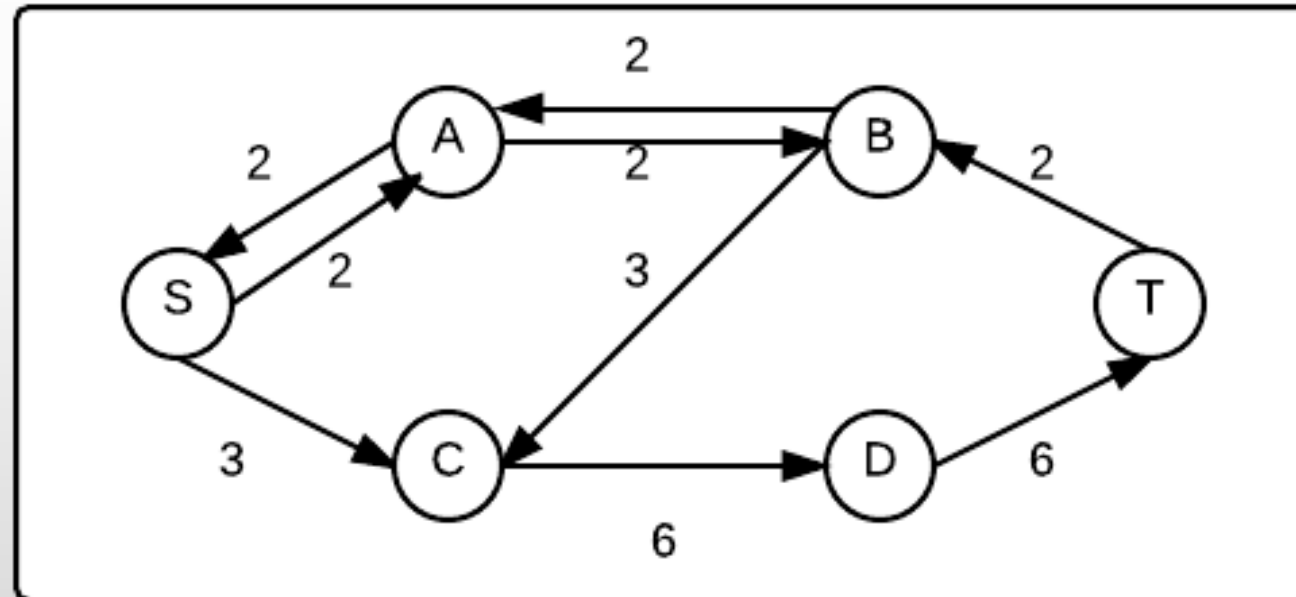
- Aşağıdaki çizge verilsin. $p=\{S,A,B,T\}$ 2 birim akış





Ford Fulkerson

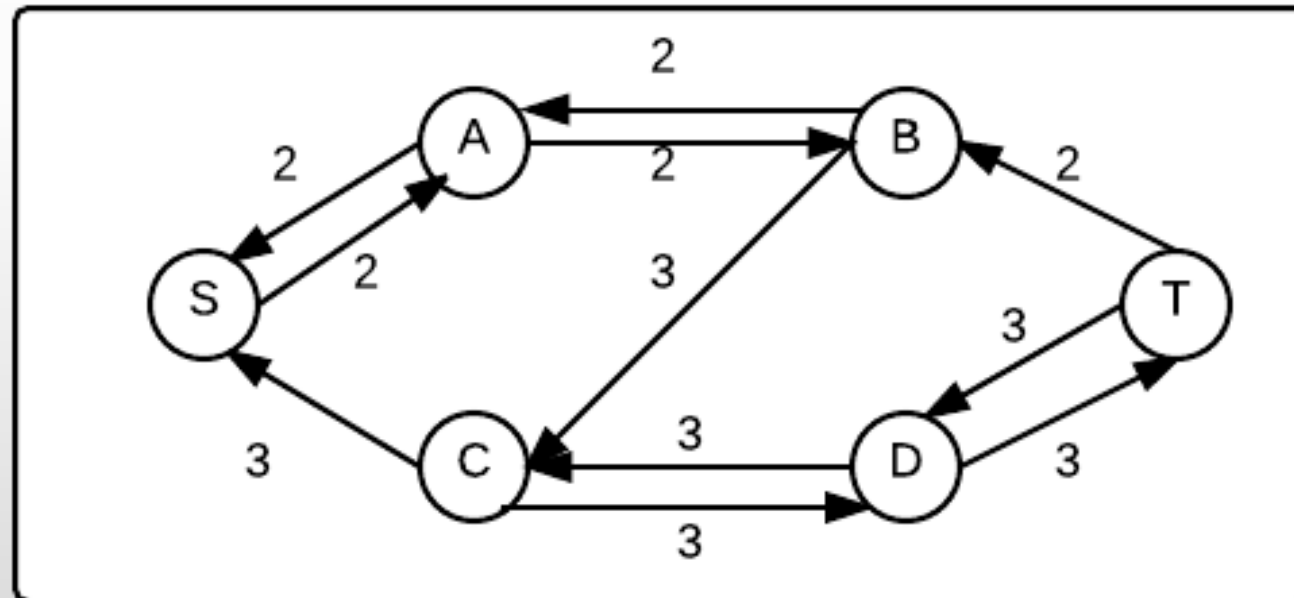
- $p=\{S,C,D,T\}$ 3 birim akış





Ford Fulkerson

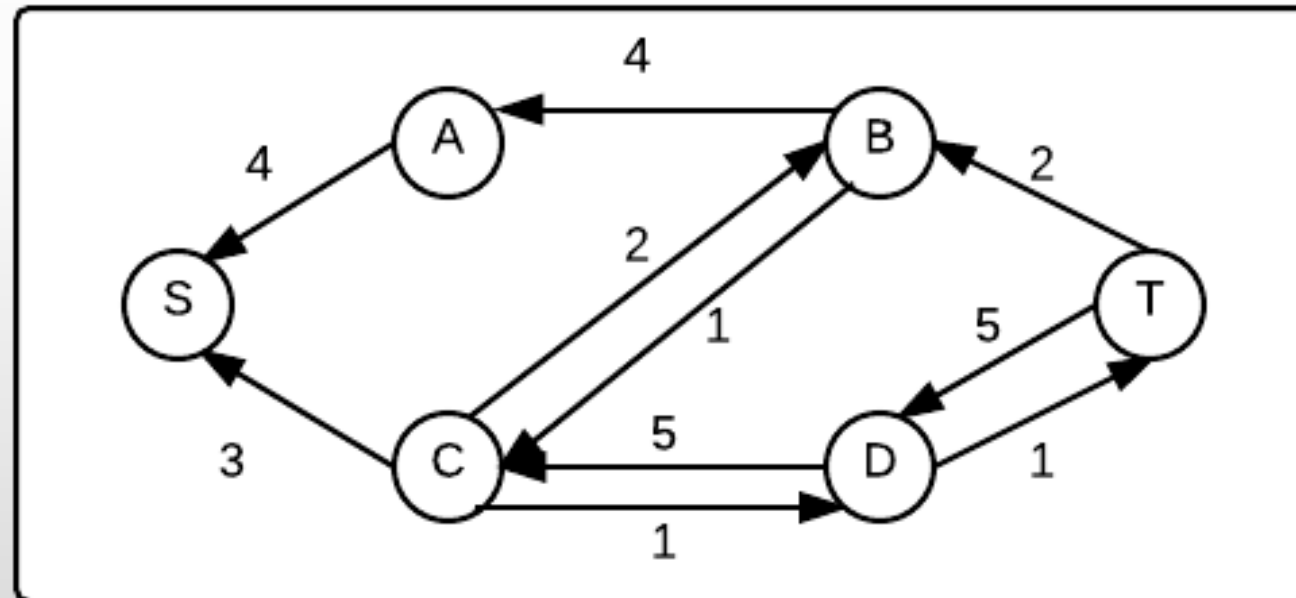
- $p=\{S,A,B,C,D,T\}$ 2 birim akış





Ford Fulkerson

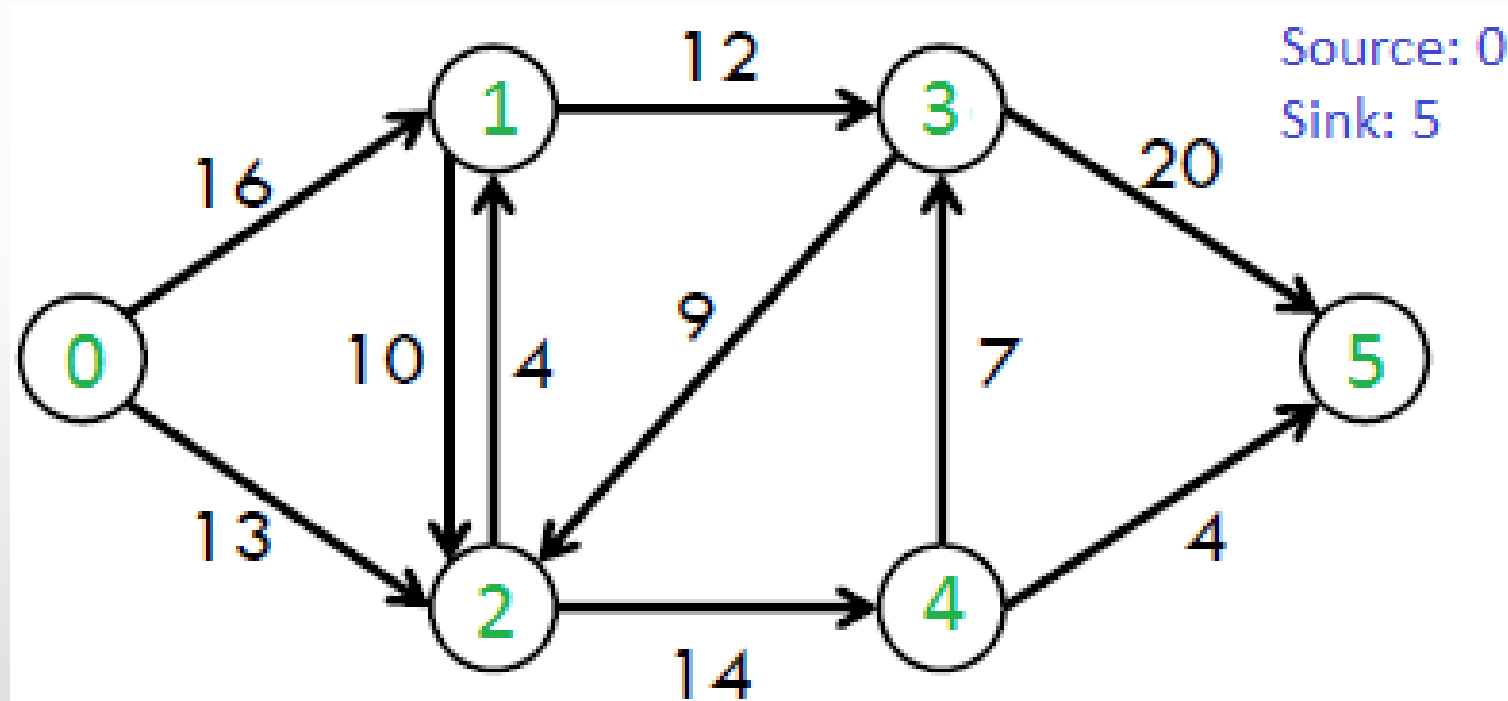
- Maksimum akış 7





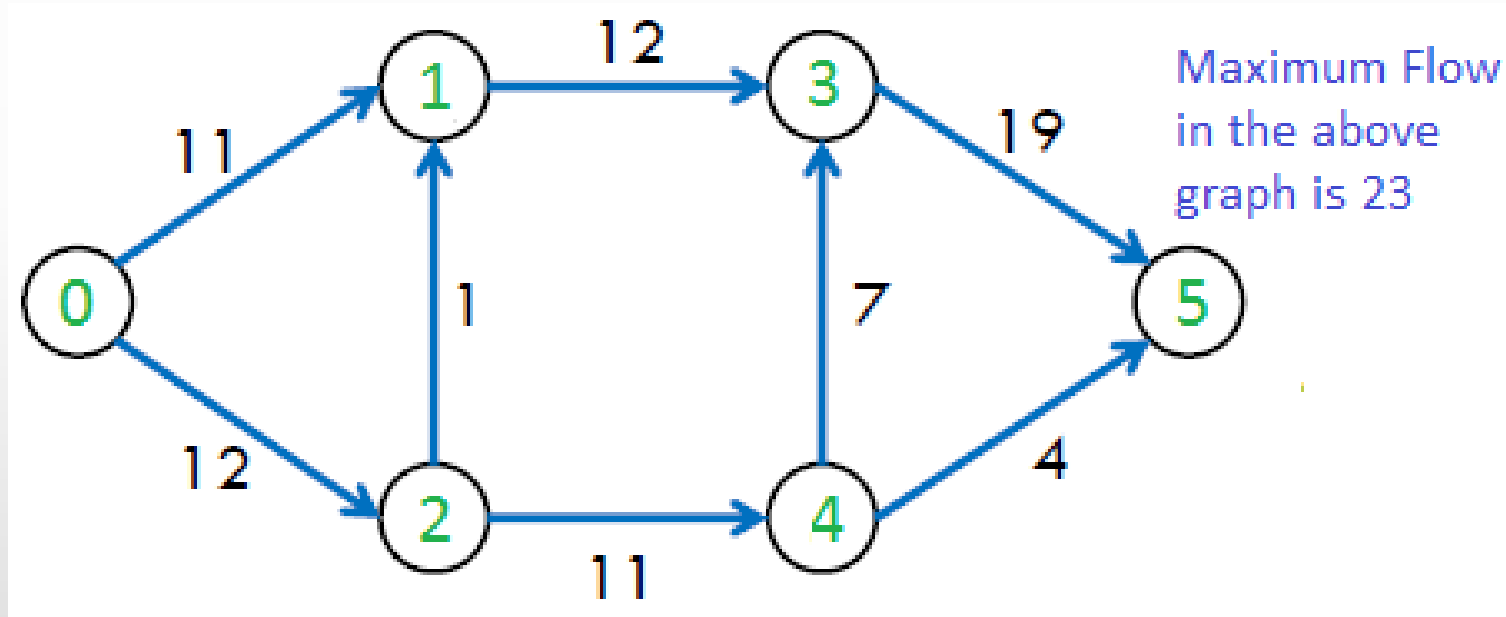


Ford Fulkerson





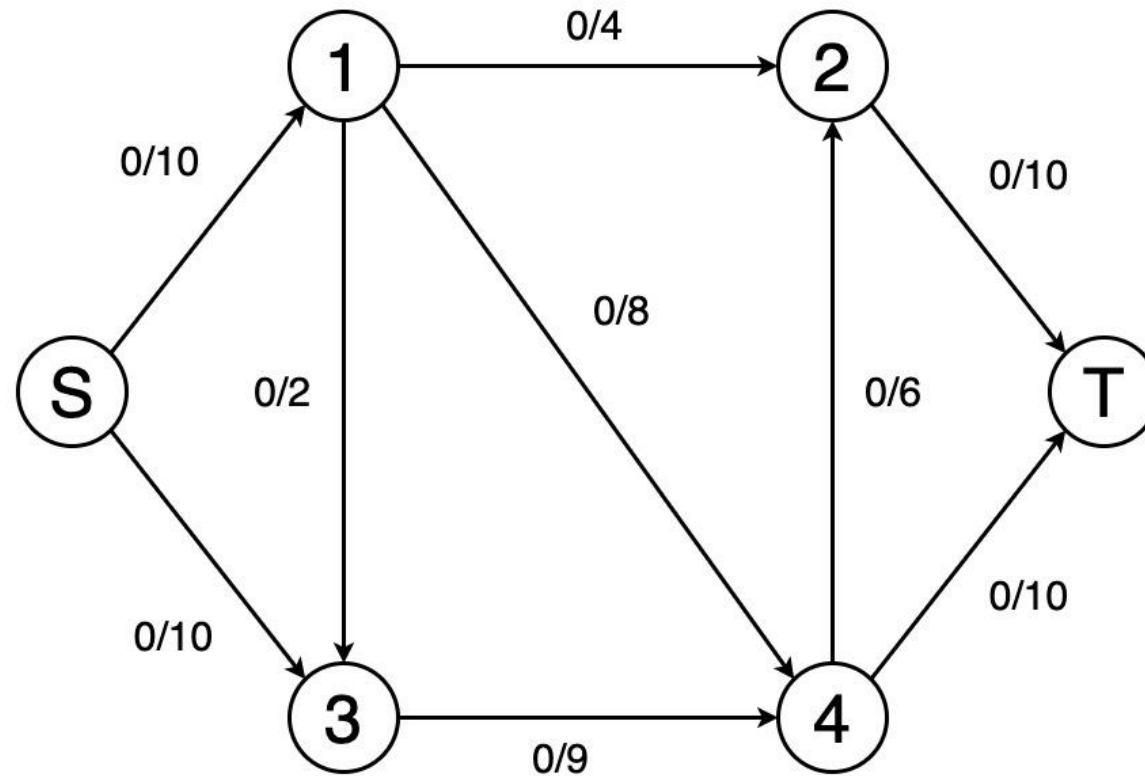
Ford Fulkerson





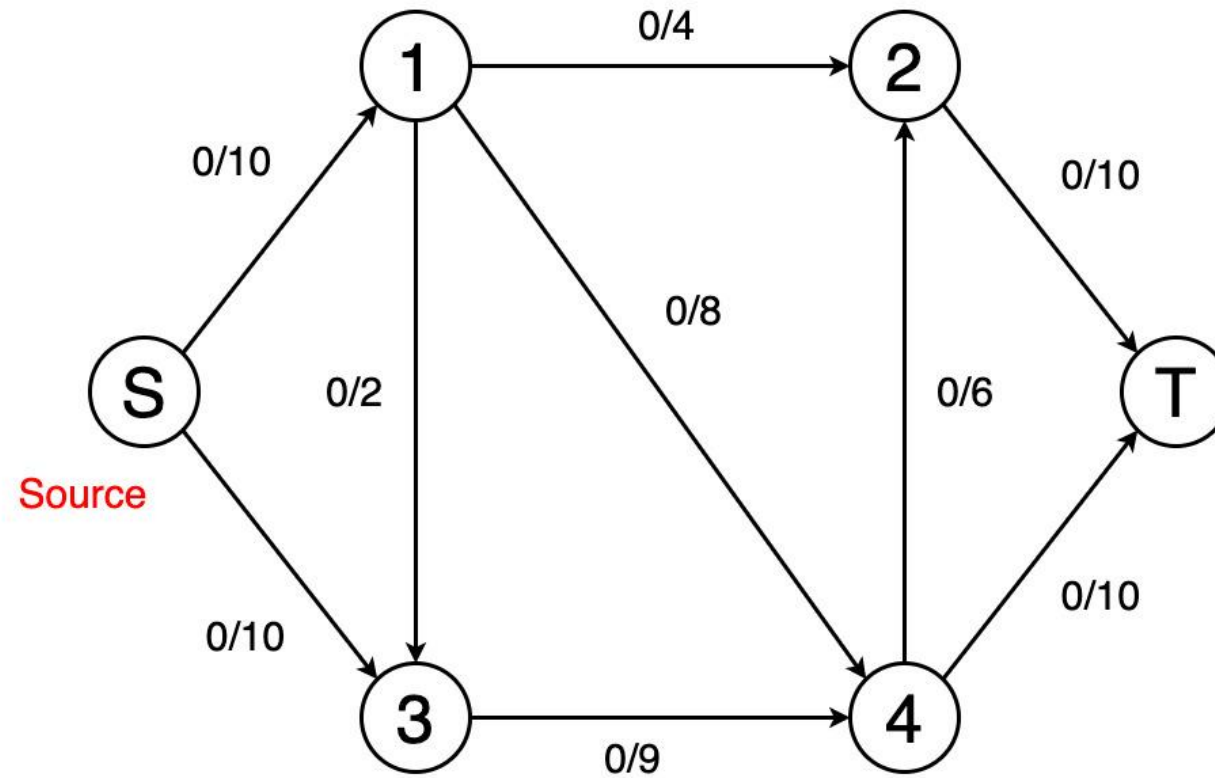


Ford Fulkerson



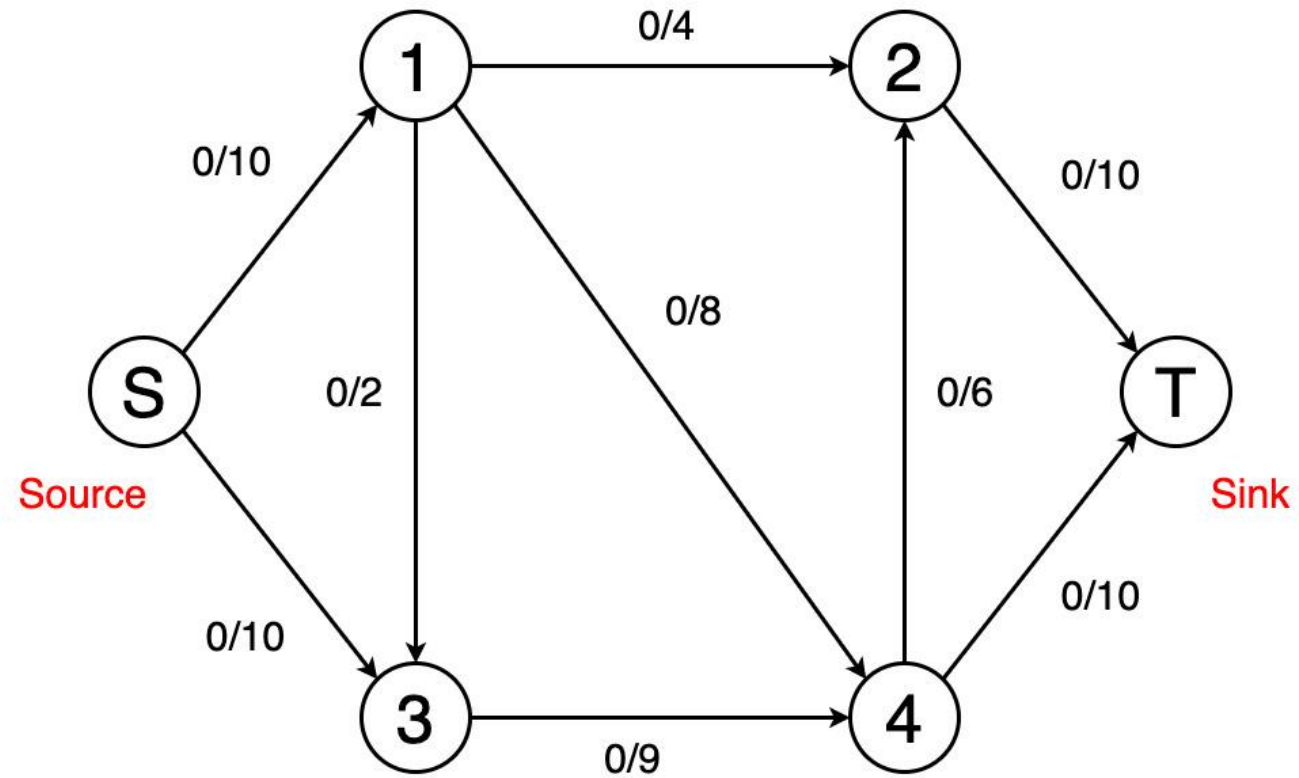


Ford Fulkerson



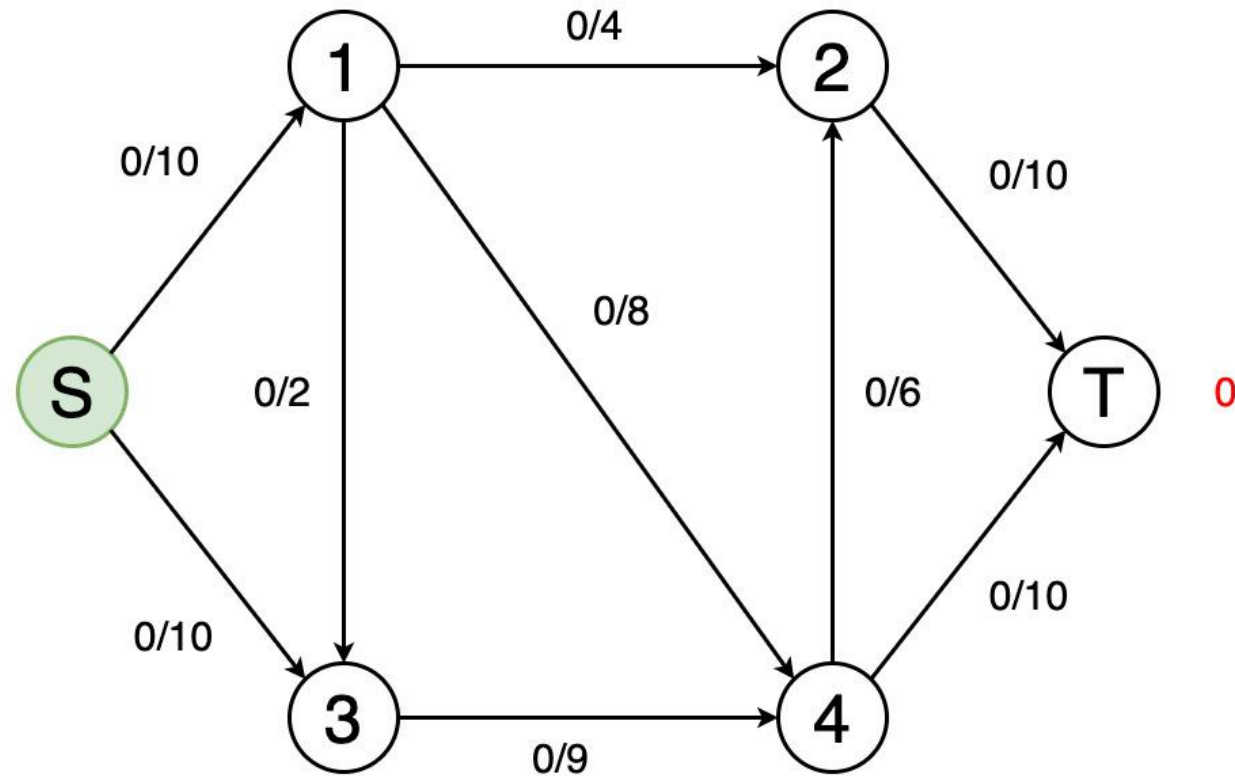


Ford Fulkerson



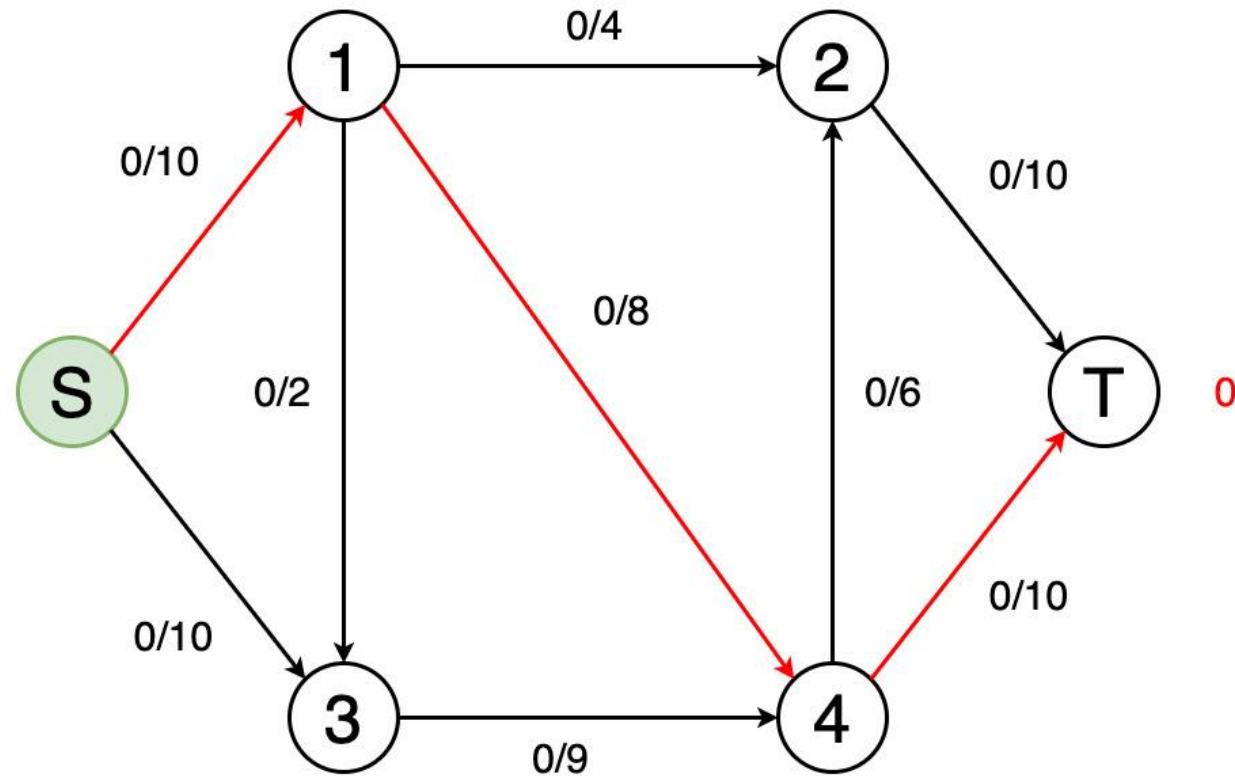


Ford Fulkerson



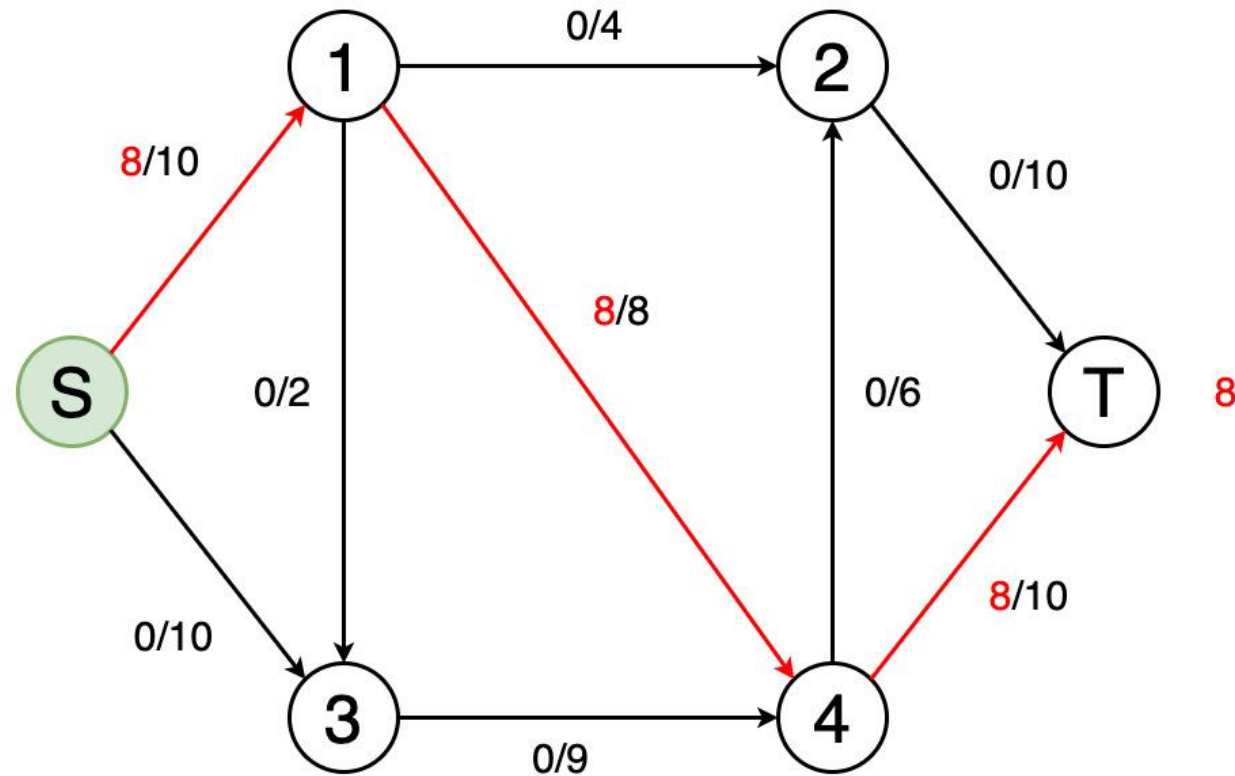


Ford Fulkerson



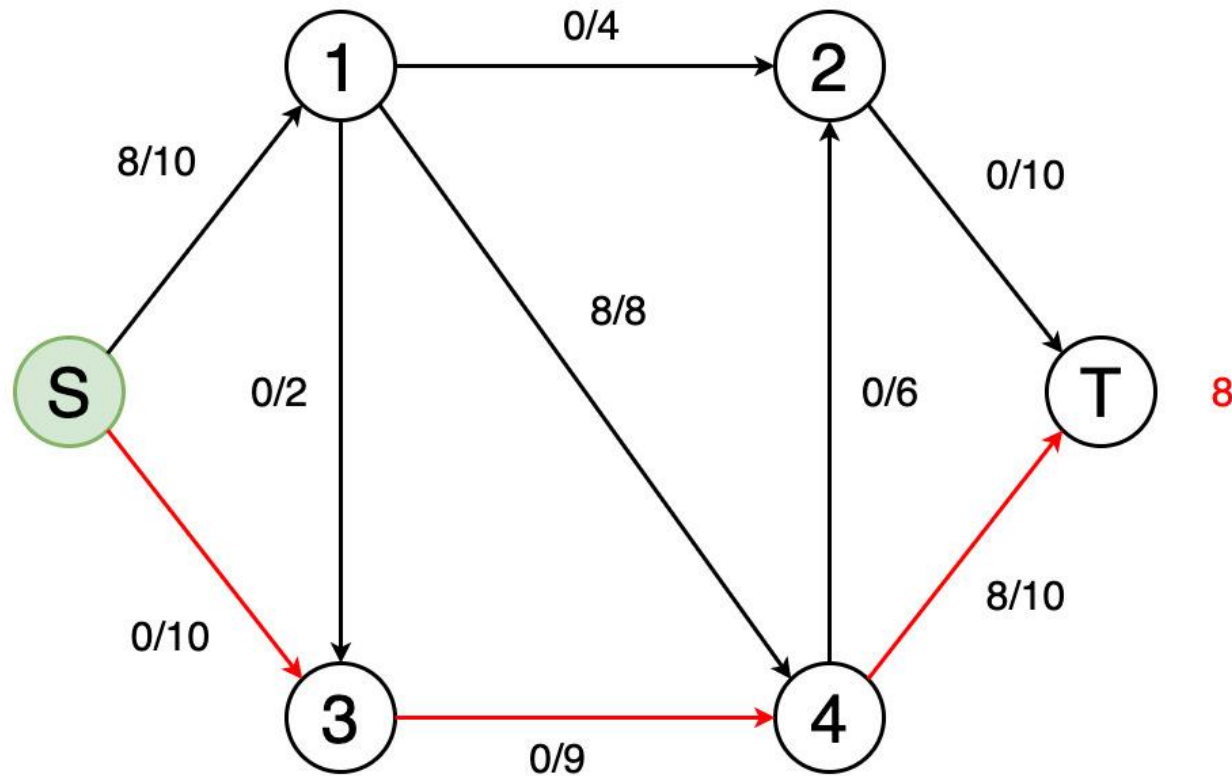


Ford Fulkerson



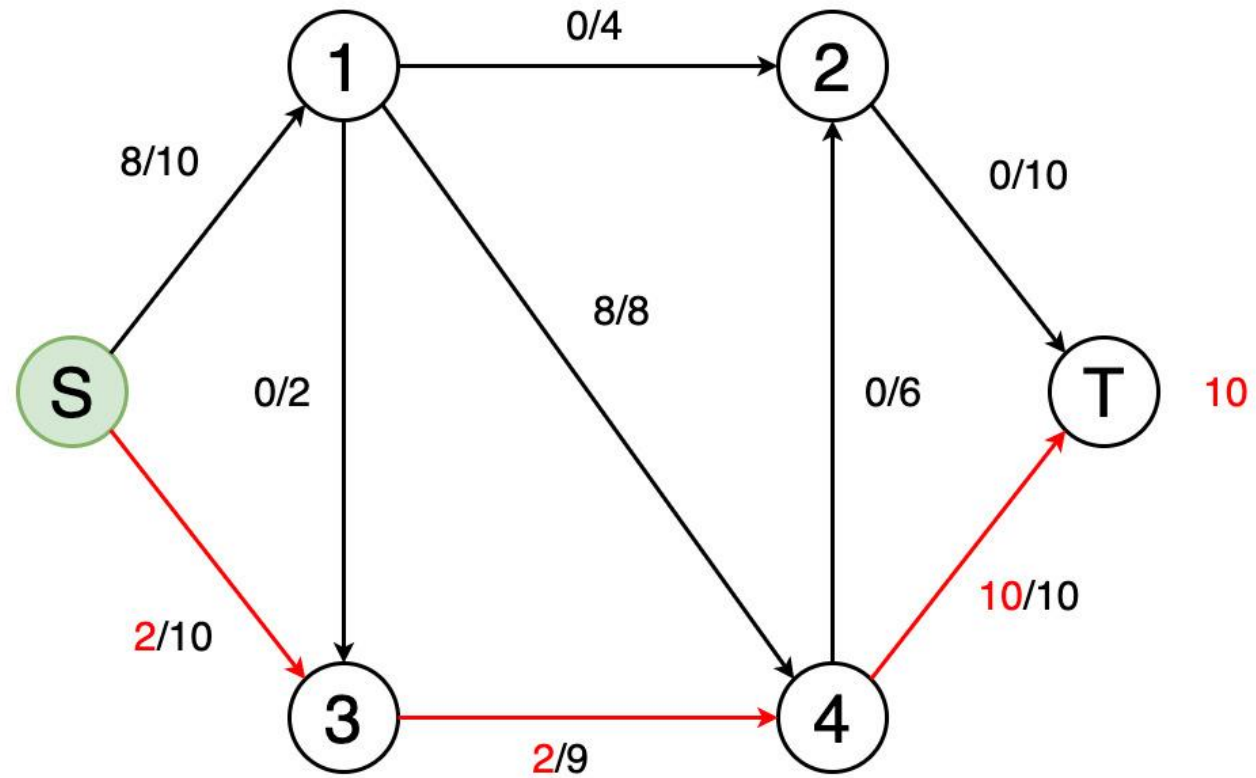


Ford Fulkerson



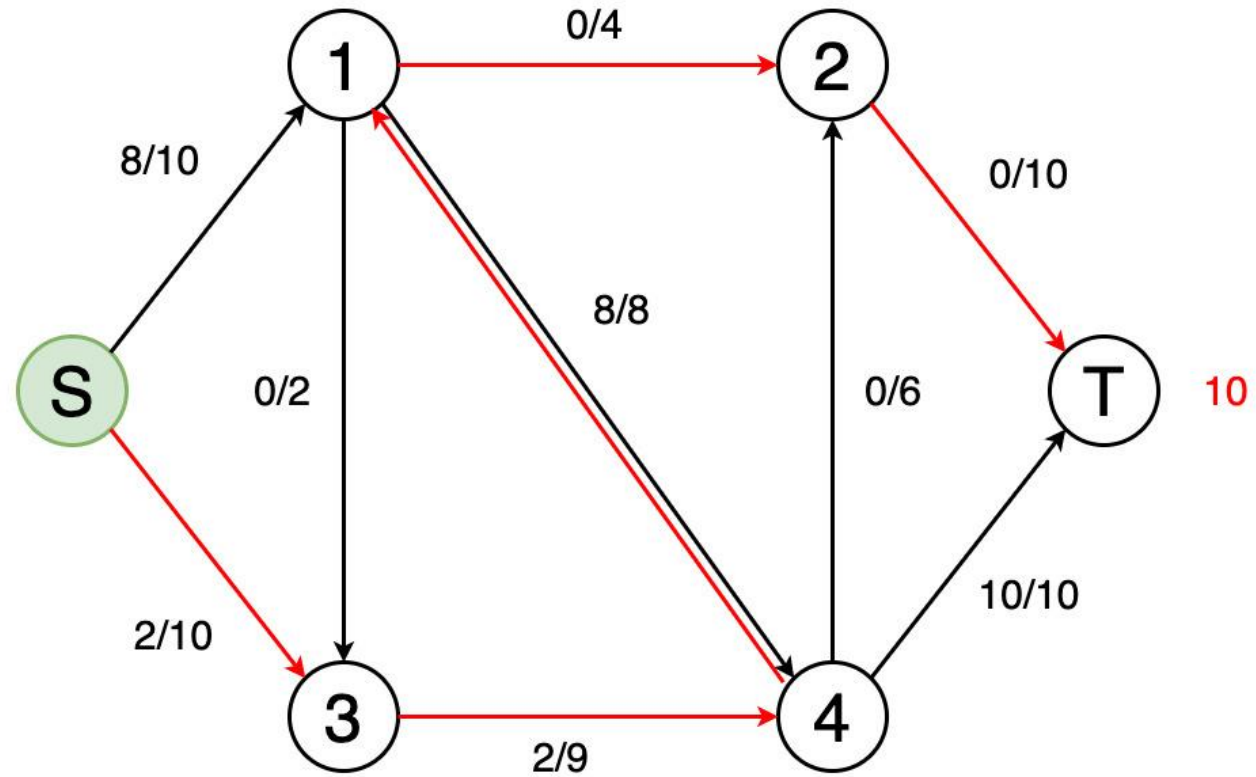


Ford Fulkerson



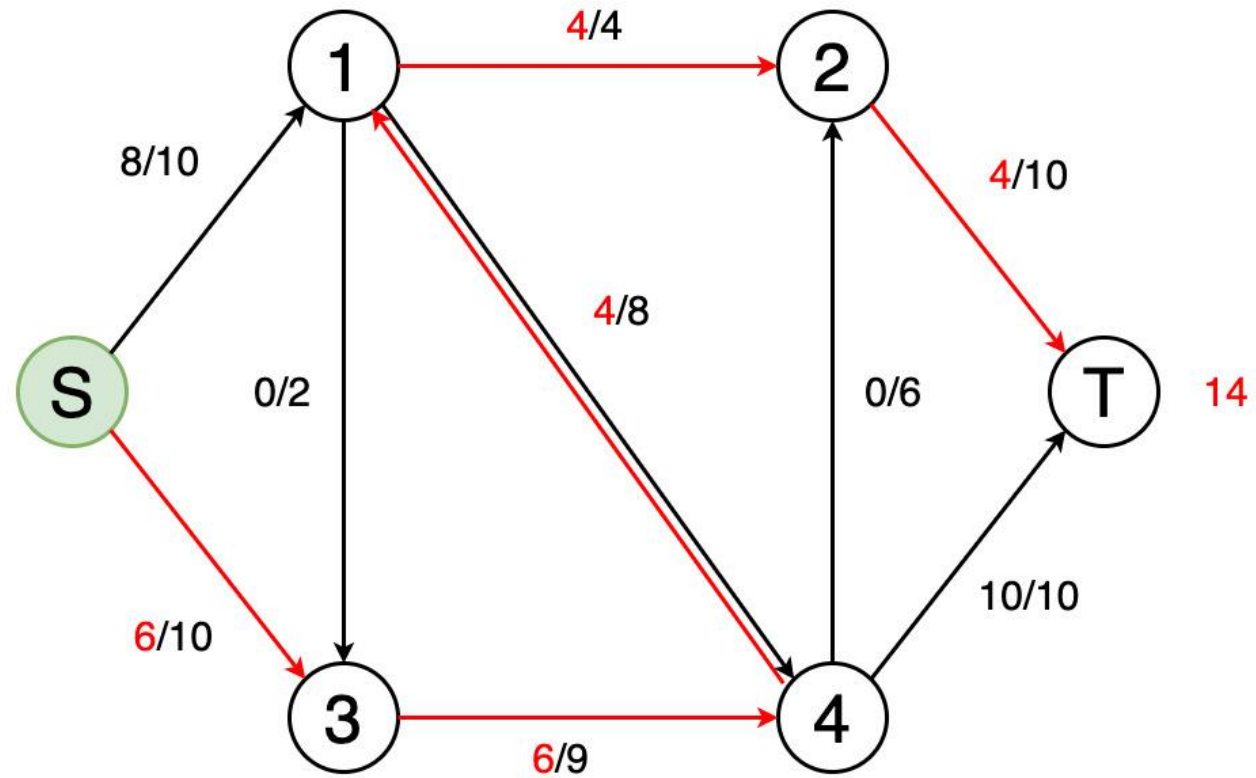


Ford Fulkerson



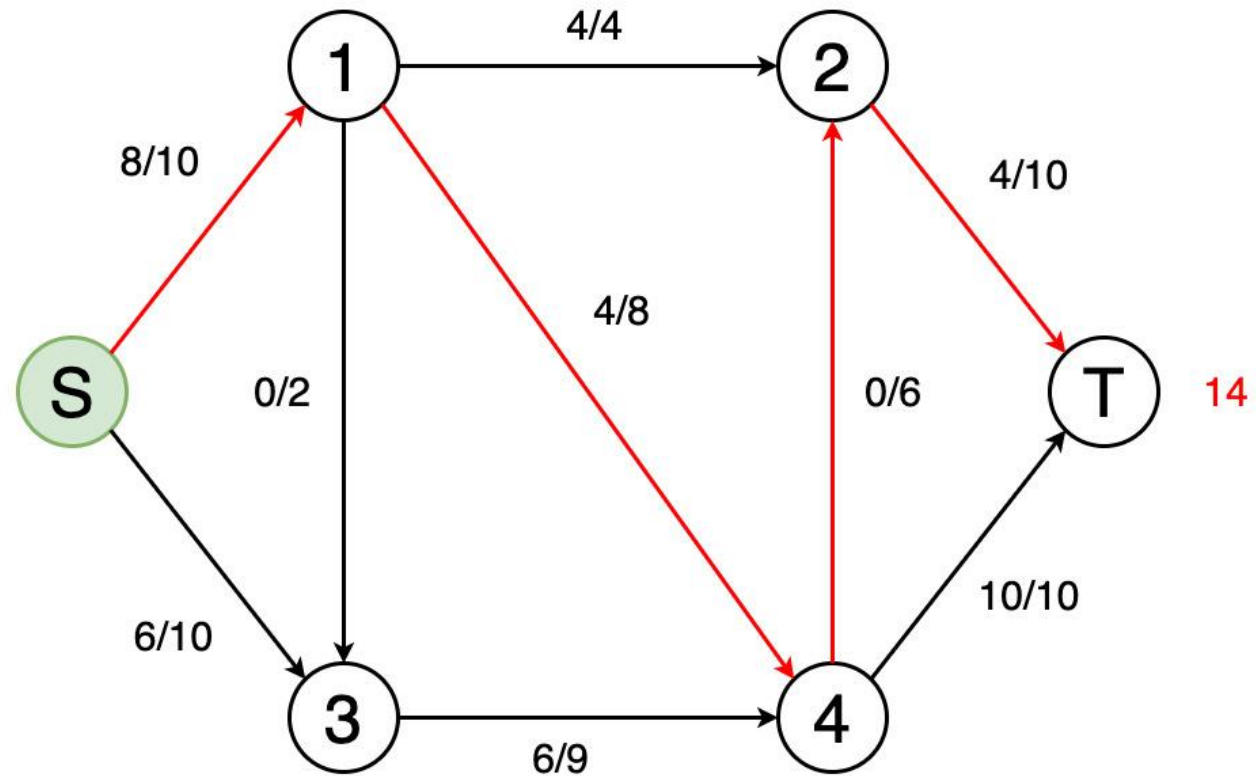


Ford Fulkerson



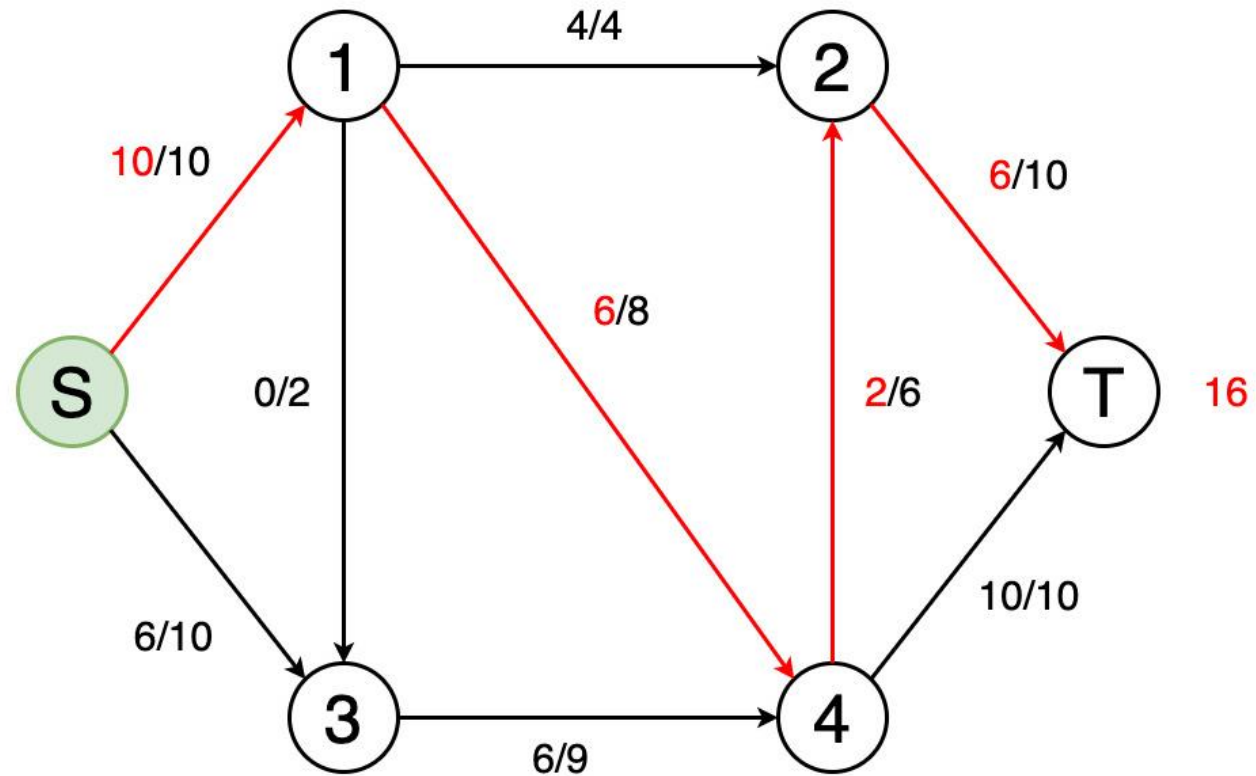


Ford Fulkerson



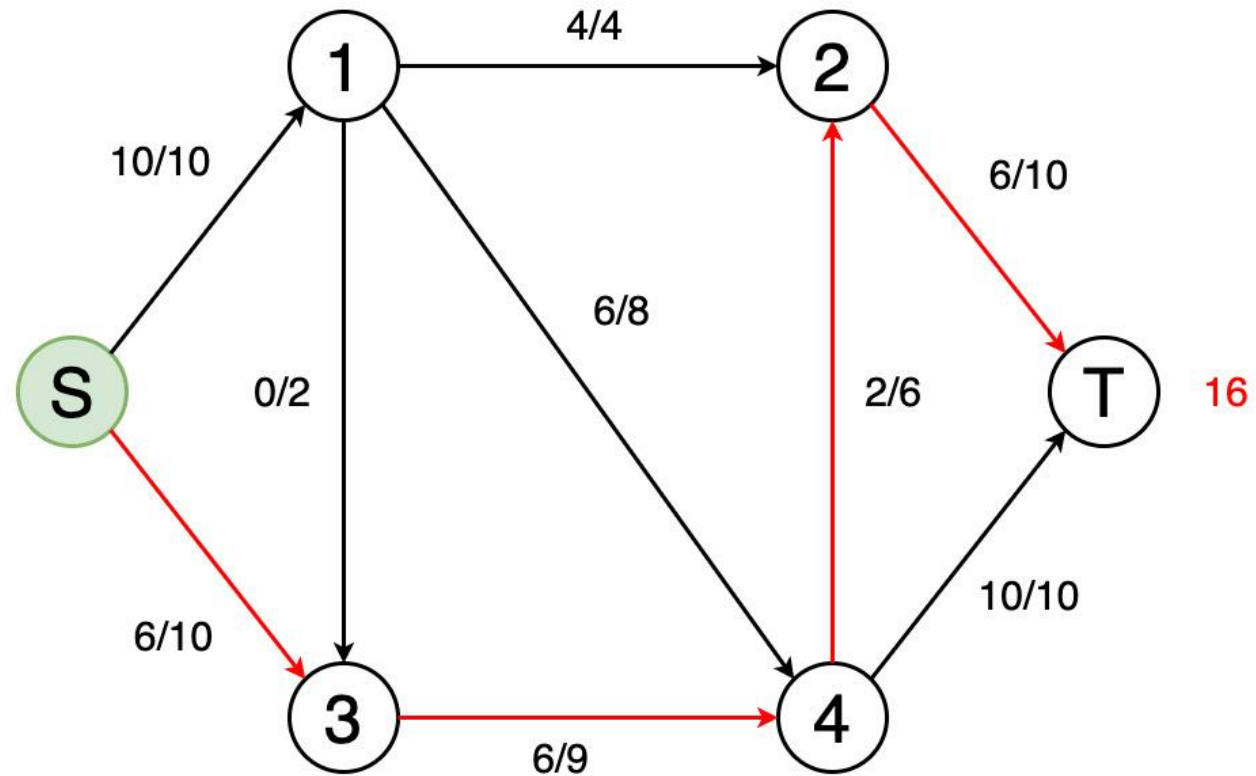


Ford Fulkerson



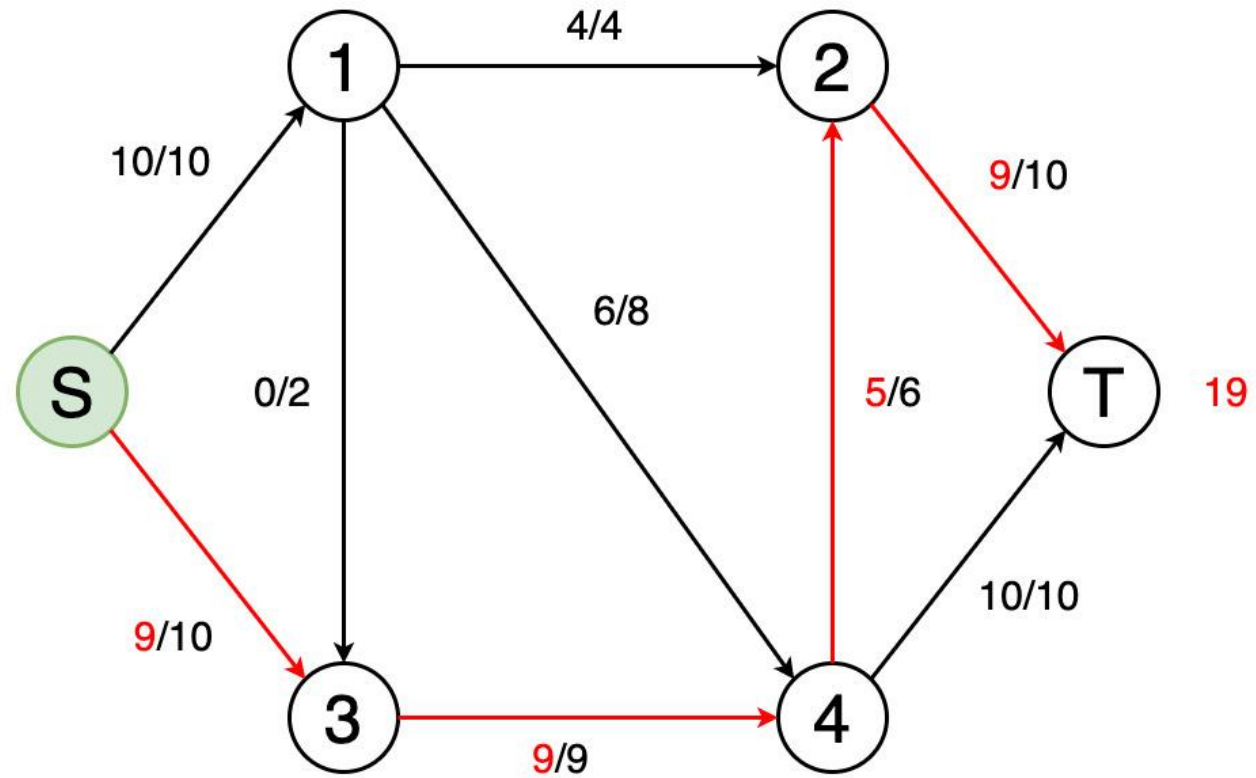


Ford Fulkerson



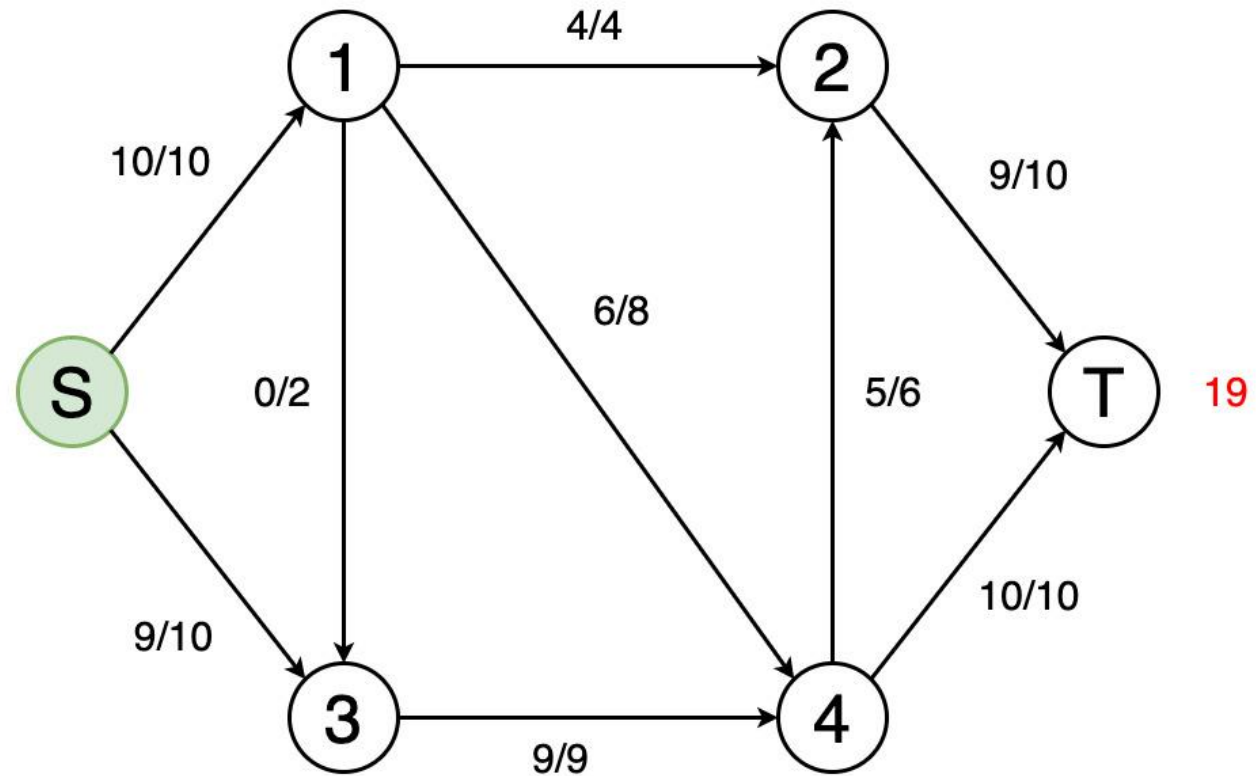


Ford Fulkerson





Ford Fulkerson







Edmonds Karp

- Ağırlıklı yönlü çizgede iki düğüm arasındaki maksimum akışı bulur.
- Ford-Fulkerson Algoritması'nın bir türevidir.
- Jack Edmonds ve Richard Karp tarafından geliştirilmiştir.
- BFS (Breadth-First Search) kullanarak artan yolları bulur ve bu yollarda maksimum akışı uygular.
- Edmond's Karp uses BFS to find an augmenting path and Dinic's uses BFS to check if more flow is possible and to construct level graph.



Algoritma Adımları

- Adım 1: BFS kullanılarak, kaynaktan hedefe olan artan yollar bulunur.
- Adım 2: Bulunan artan yollar boyunca maksimum artışa izin verilir. Bu, akış ağındaki tüm kenarlarda artışa neden olur.
- Adım 3: Hedefe ulaşılan kadar Adım 1 ve Adım 2 tekrarlanır.



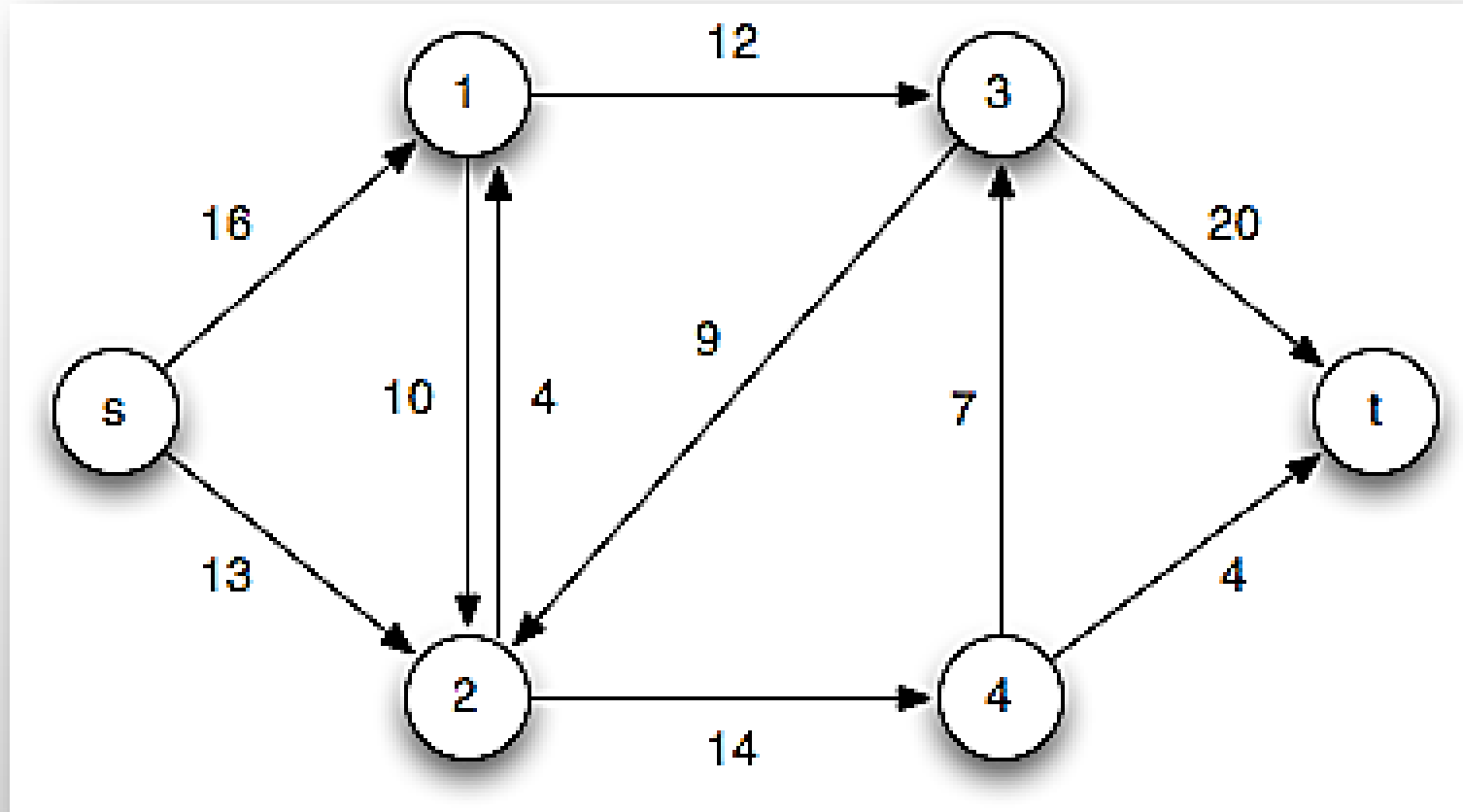
Karmaşıklık Analizi

- Algoritma karmaşıklığı,
 - BFS kullanılarak artan yolların bulunmasına dayanır ve
 - $O(V E^2)$ karmaşıklığına sahiptir.
 - E kenar sayısı
 - V düğüm sayısı.
 - Her aşamada kaynaktan hedefe artış yollarını bulmak için BFS kullanır.
 - Artış yolundaki tüm kenarların kapasitelerini artırmak için tekrar BFS.
 - En kötü durumda, bu işlemler V kez gerçekleşir.



Örnek

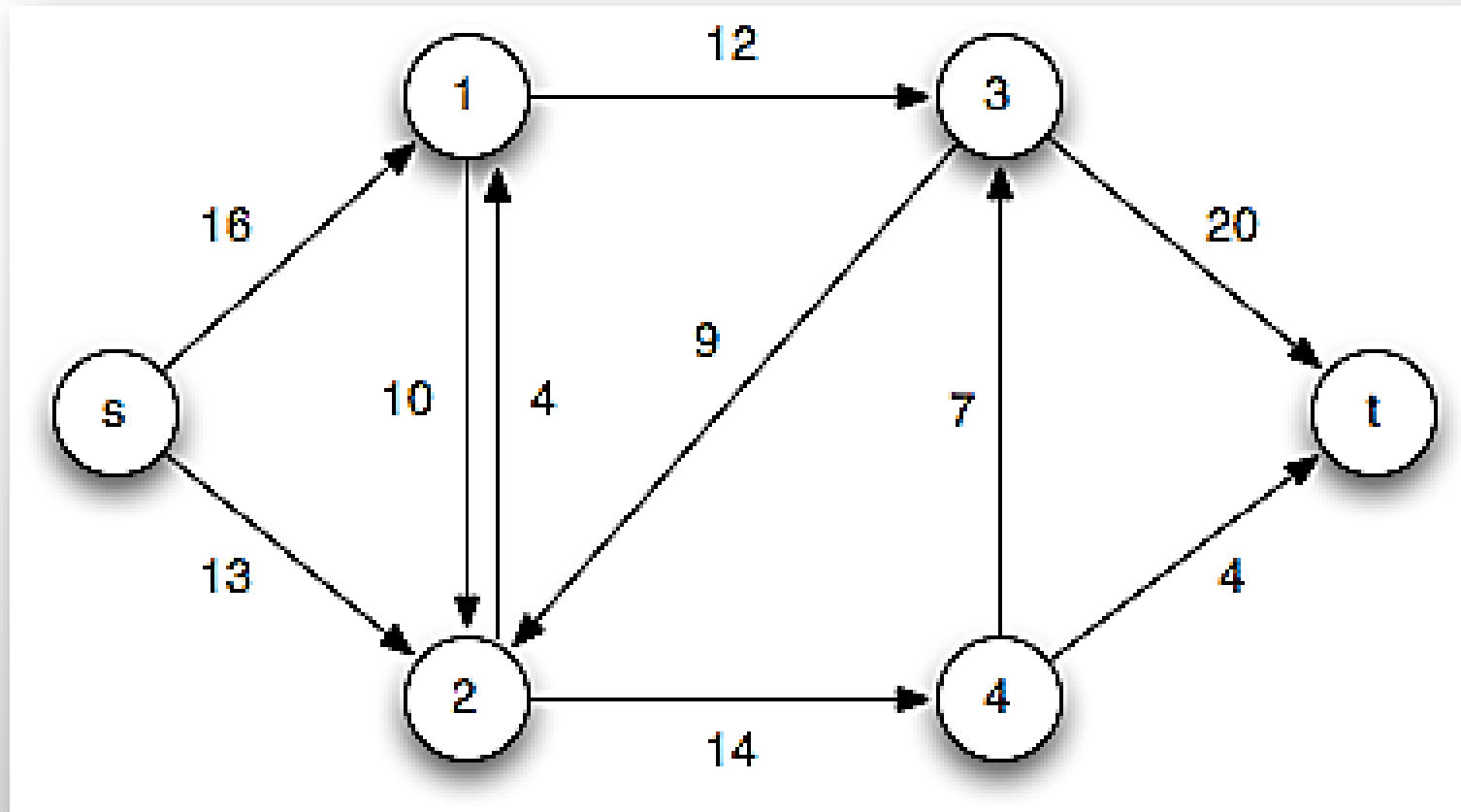
- Aşağıdaki çizge verilmiş olsun





Örnek

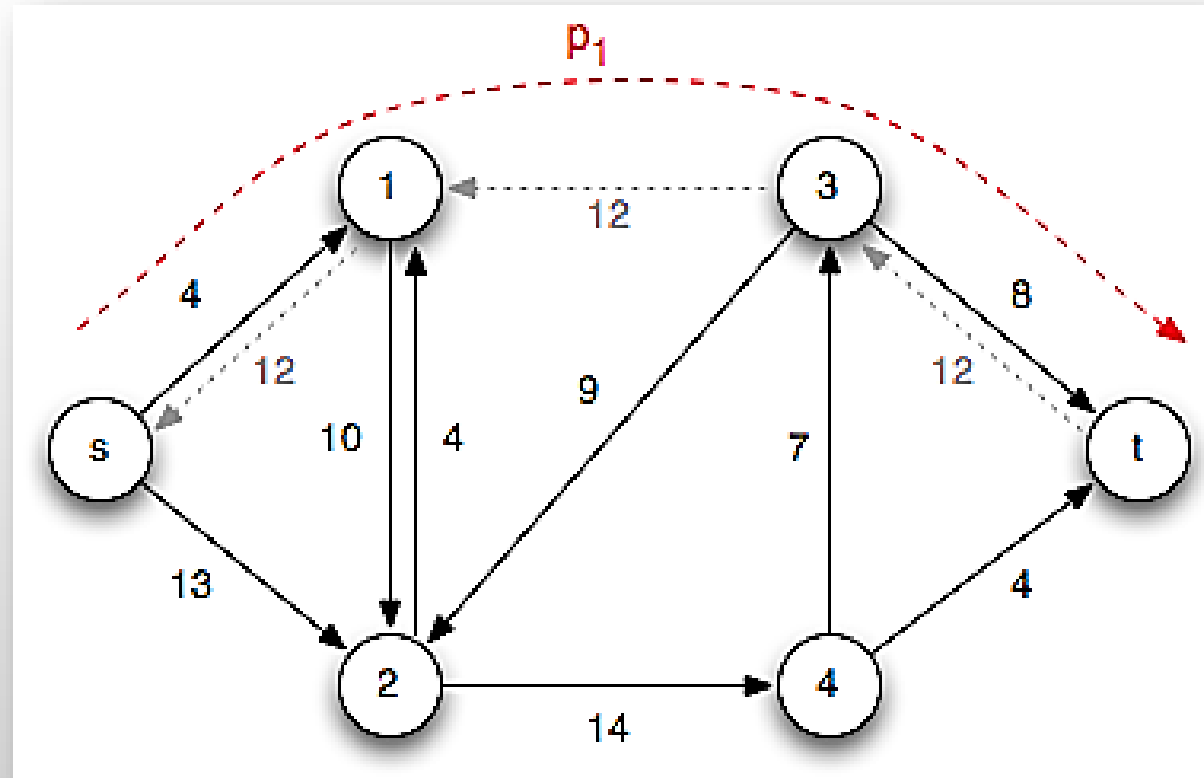
- $|f^*| \leq 24$ (s'den çıkan, ya da t'ye giren kenarların ağırlıkları toplamı)





Iteration 1

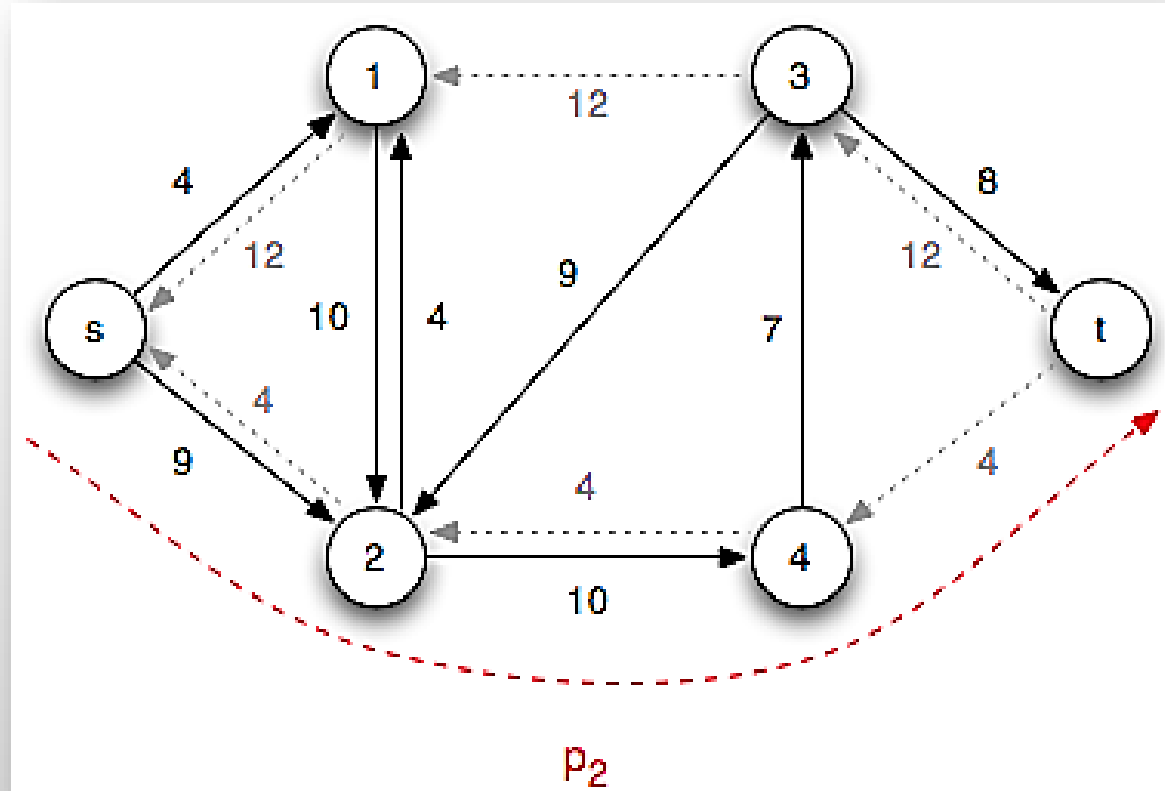
- $c(1,3)$ (*residual network*) yüzünden $cf(p_1) = 12$ 'ye sahip $p_1 = \langle s, 1, 3, t \rangle$ yolu artan yol (*augmenting path*) olarak seçilir.





Iteration 2

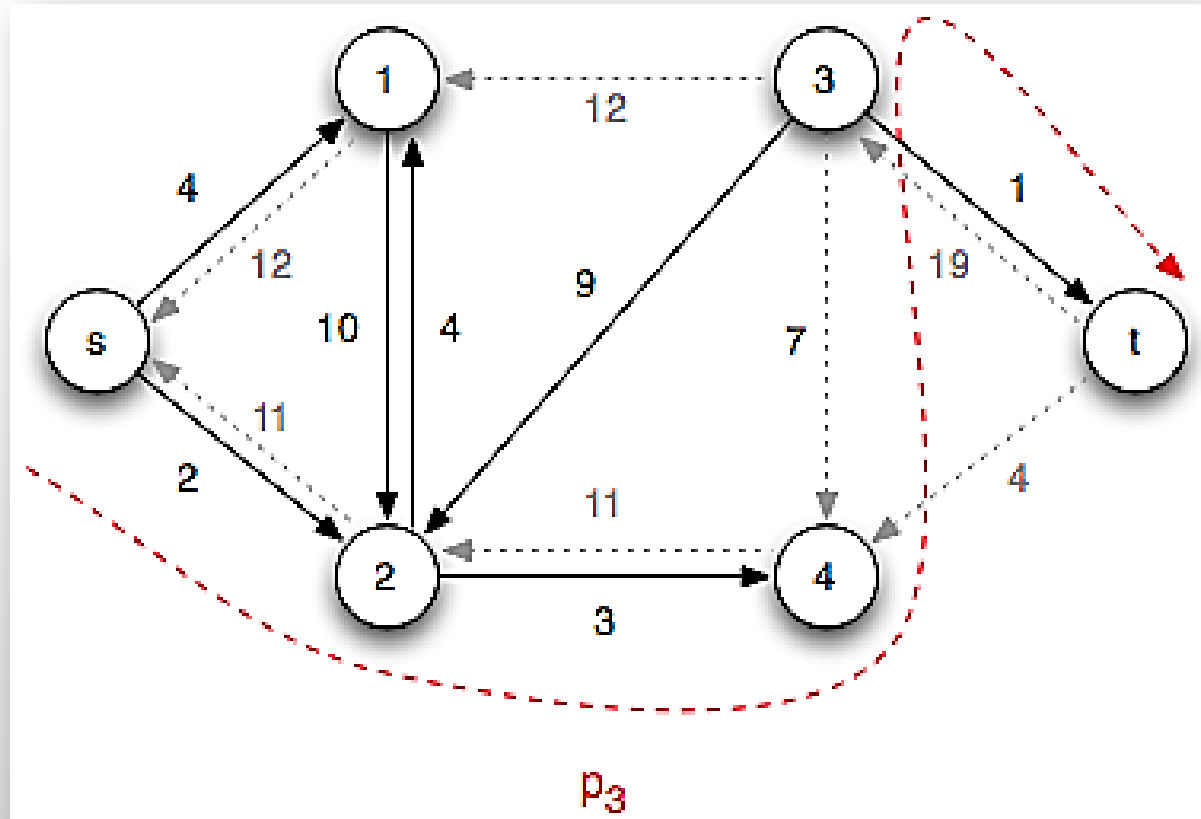
- $c(4,t)$ yüzünden $cf(p_2) = 4$ 'e sahip $p_2 = \langle s, 2, 4, t \rangle$ yolu artan yol olarak seçilir.





Iteration 3

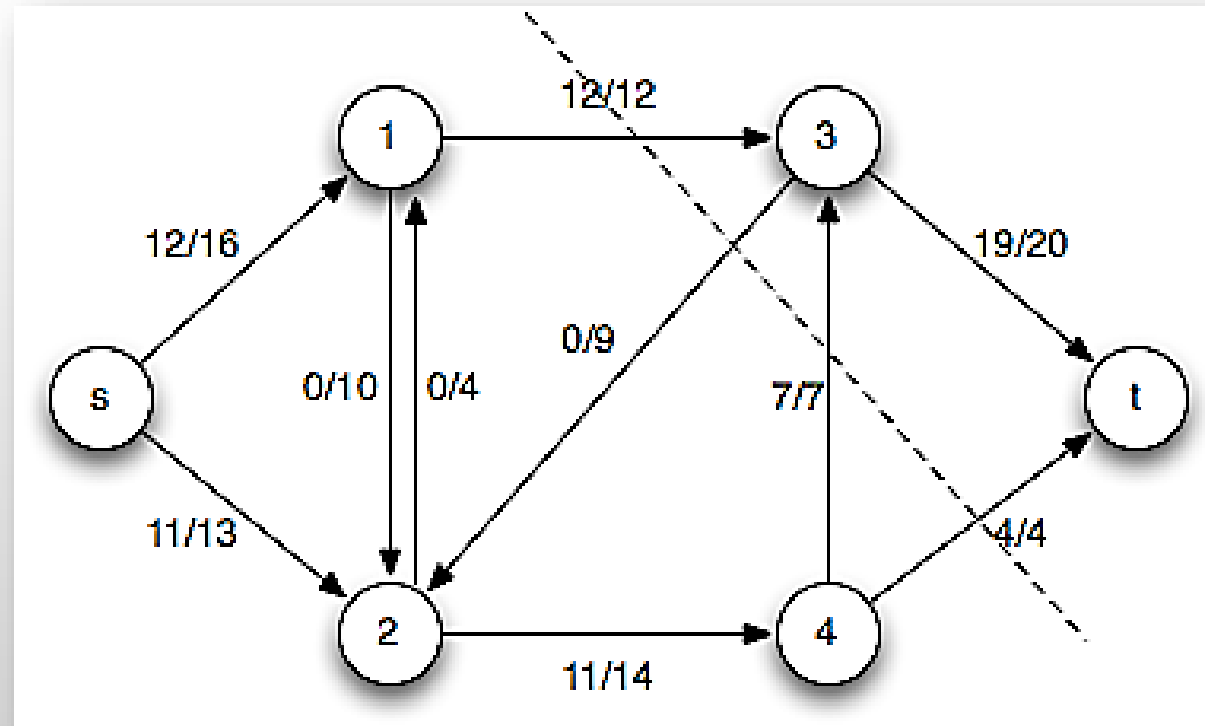
- $c(4,3)$ yüzünden $cf(p_3) = 7$ 'ye sahip $p_3 = \langle s, 2, 4, 3, t \rangle$ yolu artan yol olarak seçilir.





Son Durum

- Sadece 3 numaralı düğüm ek kapasiteye sahip. final flow network with a min-cut. $|f^*| = 19 + 4 = 12 + 11 = 23$.

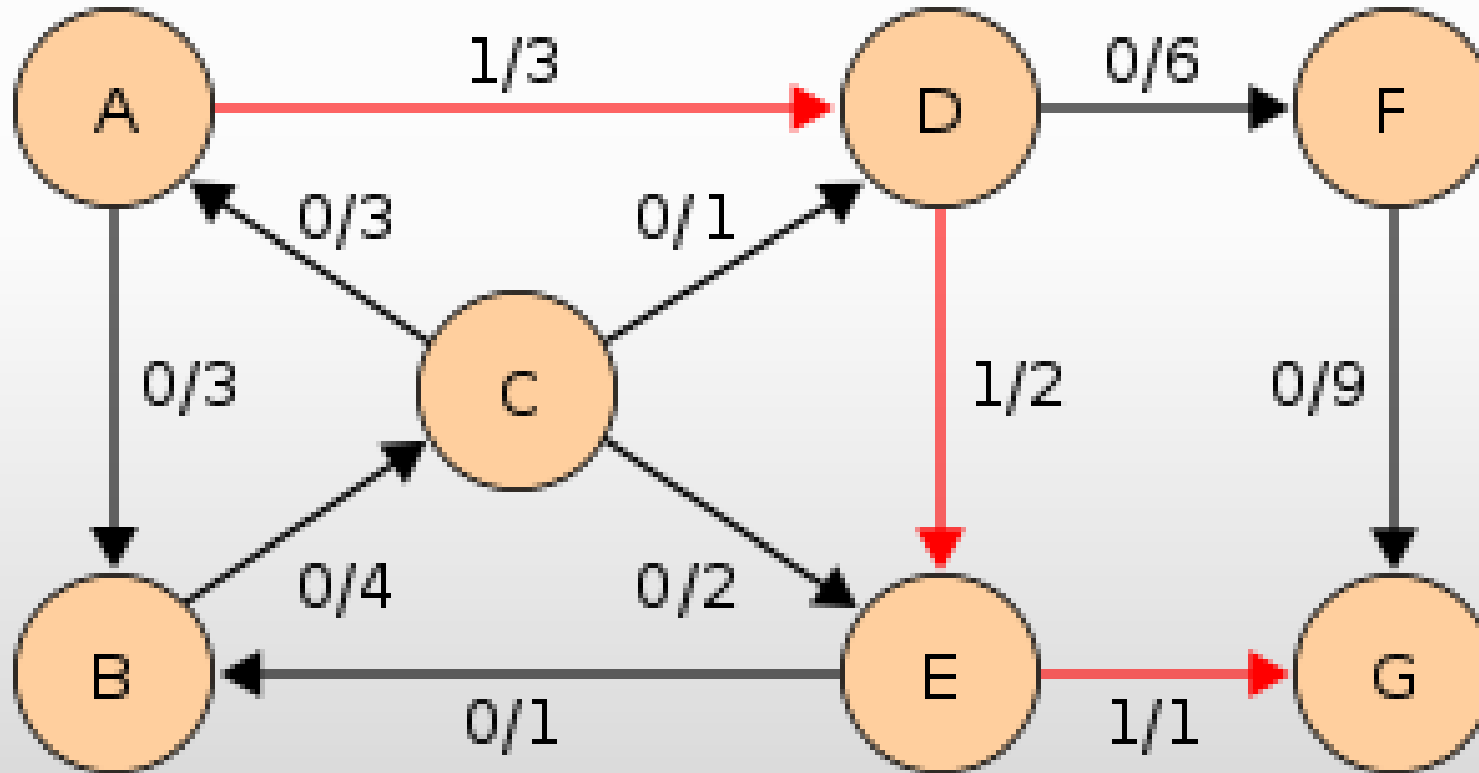






Edmonds–Karp

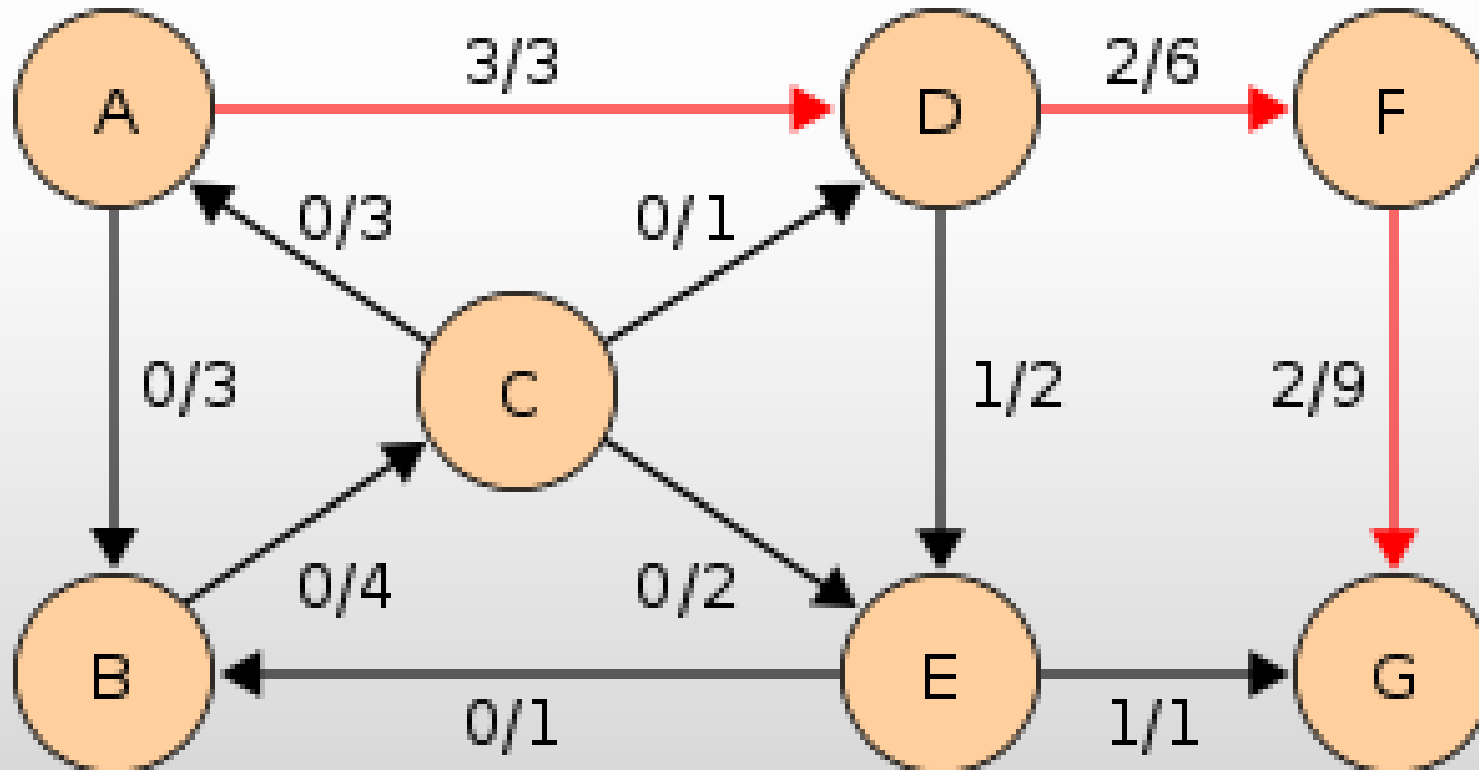
- $p = \{A, D, E, G\}$, flow = 1





Edmonds–Karp

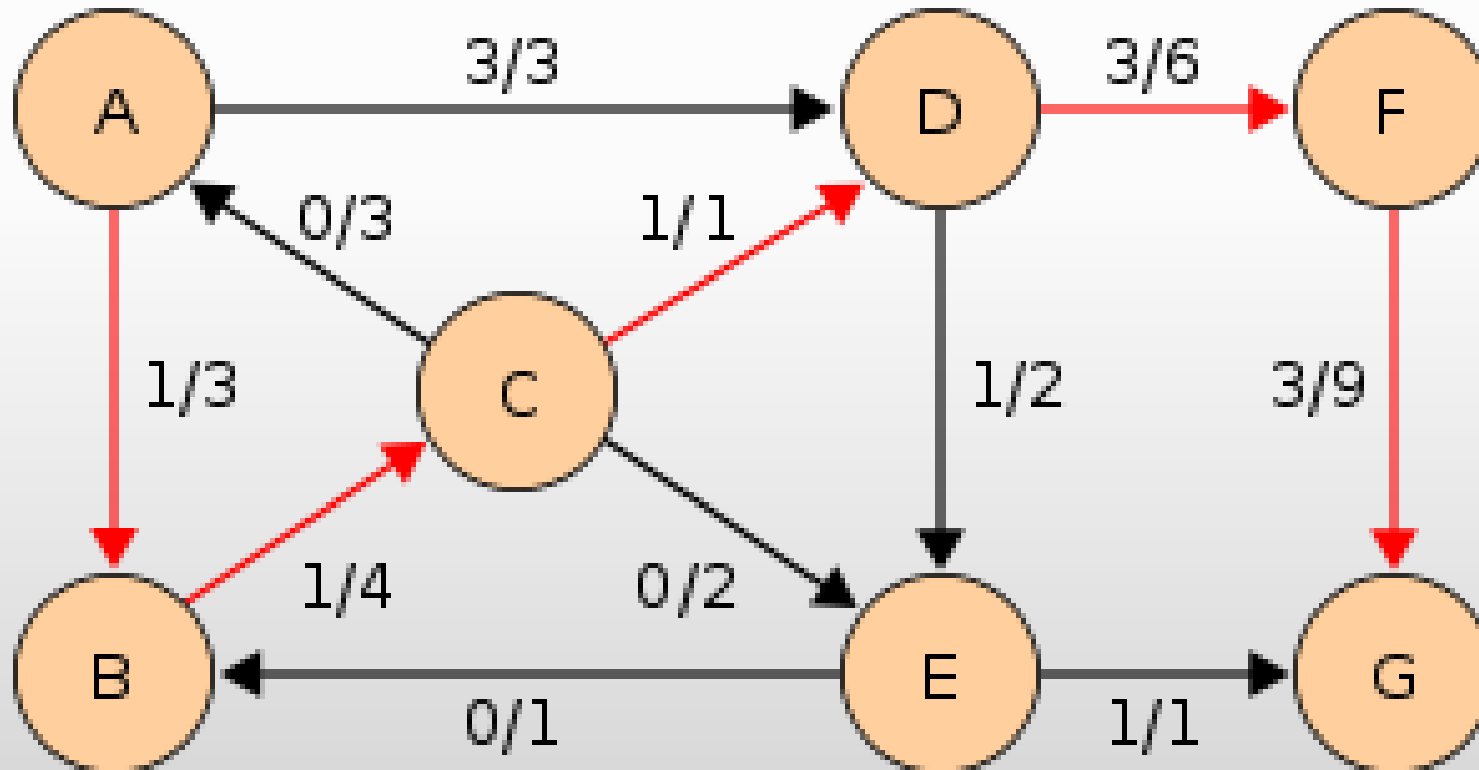
- $p = \{A, D, F, G\}$, flow = 2





Edmonds–Karp

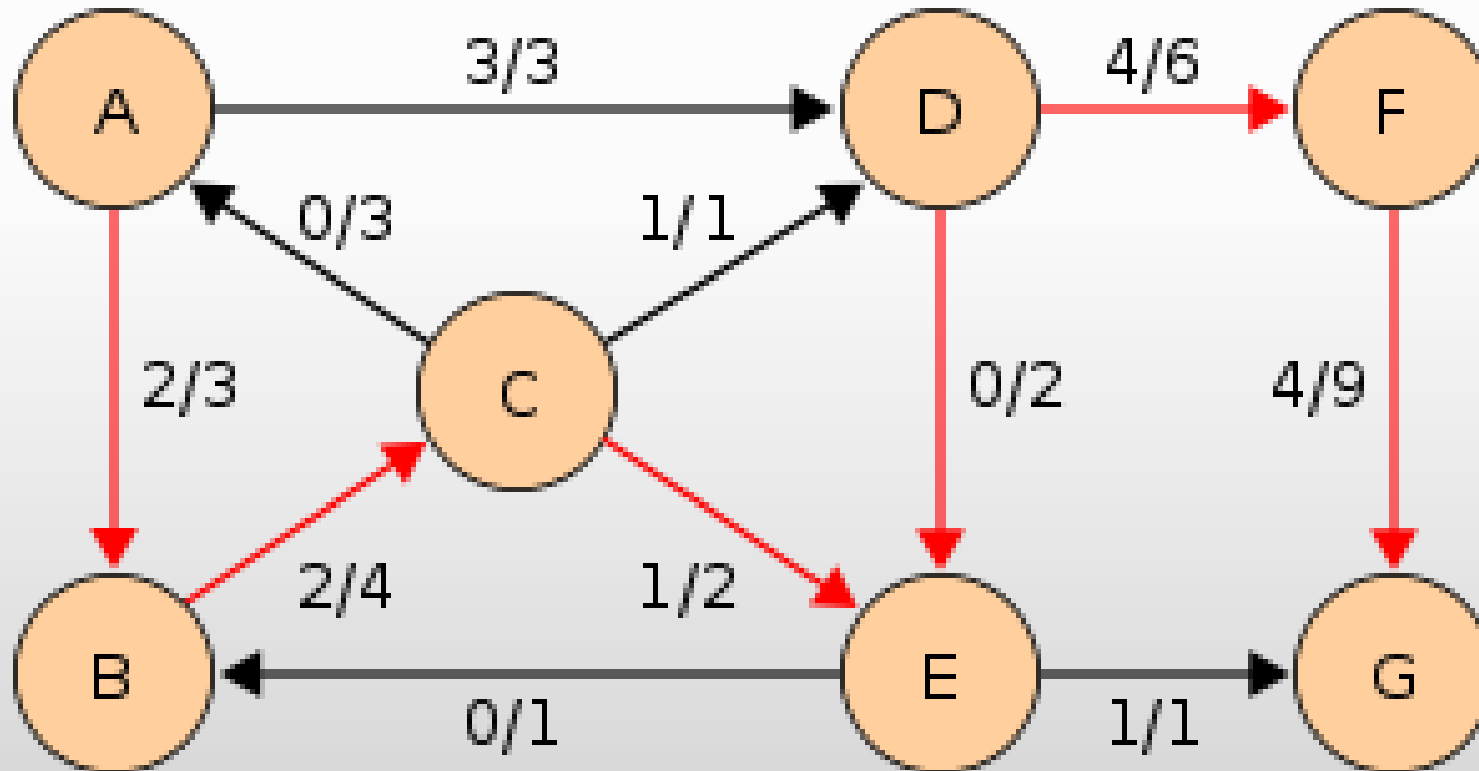
- $p = \{A, B, C, D, F, G\}$, flow = 1





Edmonds–Karp

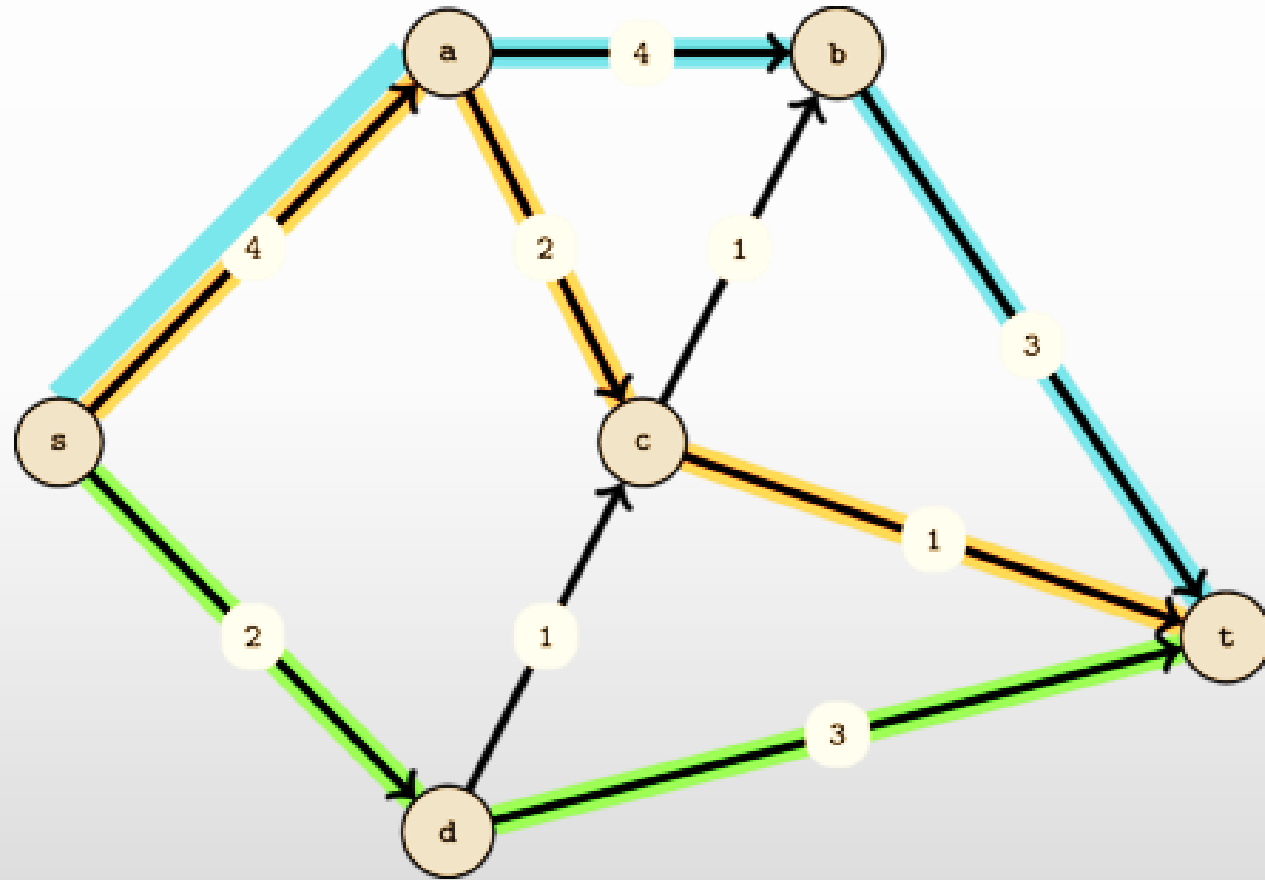
- $p = \{A, B, C, E, D, F, G\}$, flow = 1



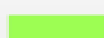






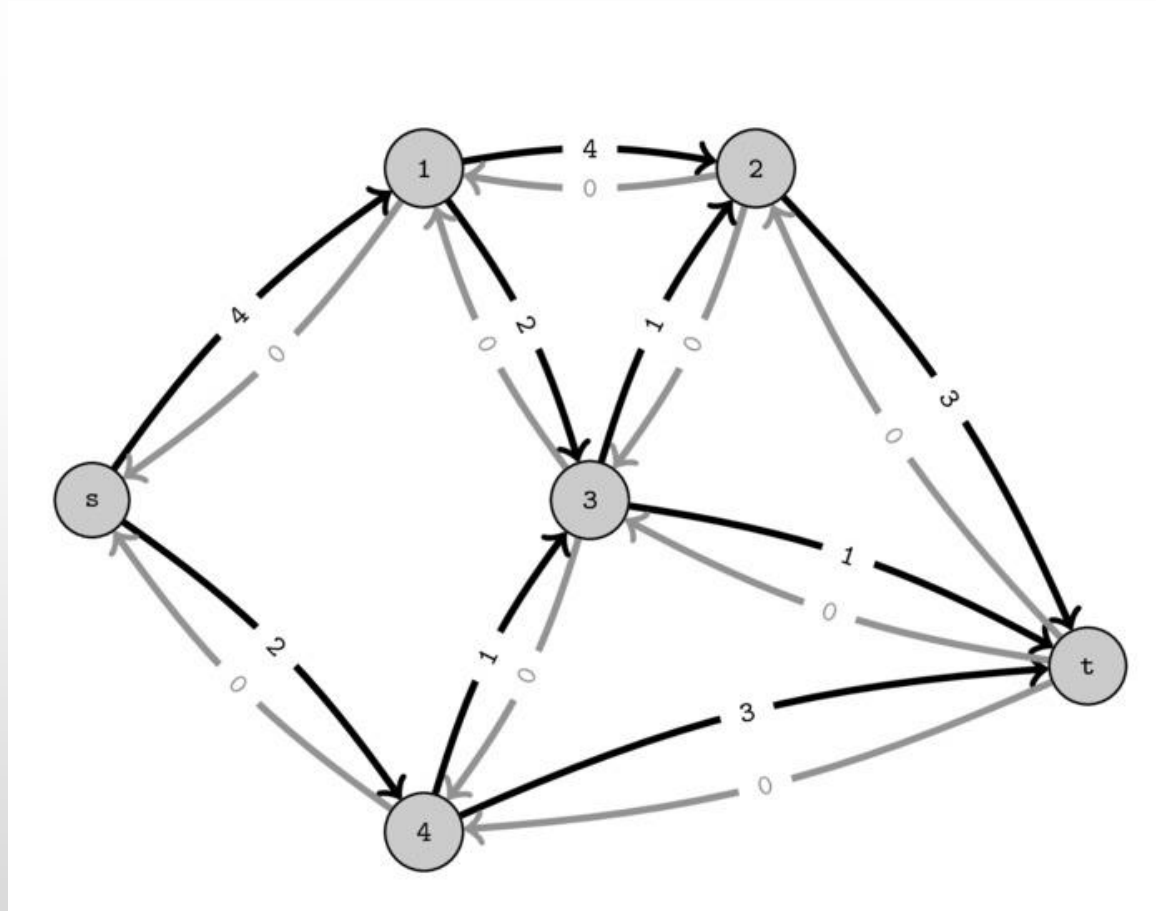
Edmonds Karp



-  3 units of flow
-  1 unit of flow
-  2 units of flow

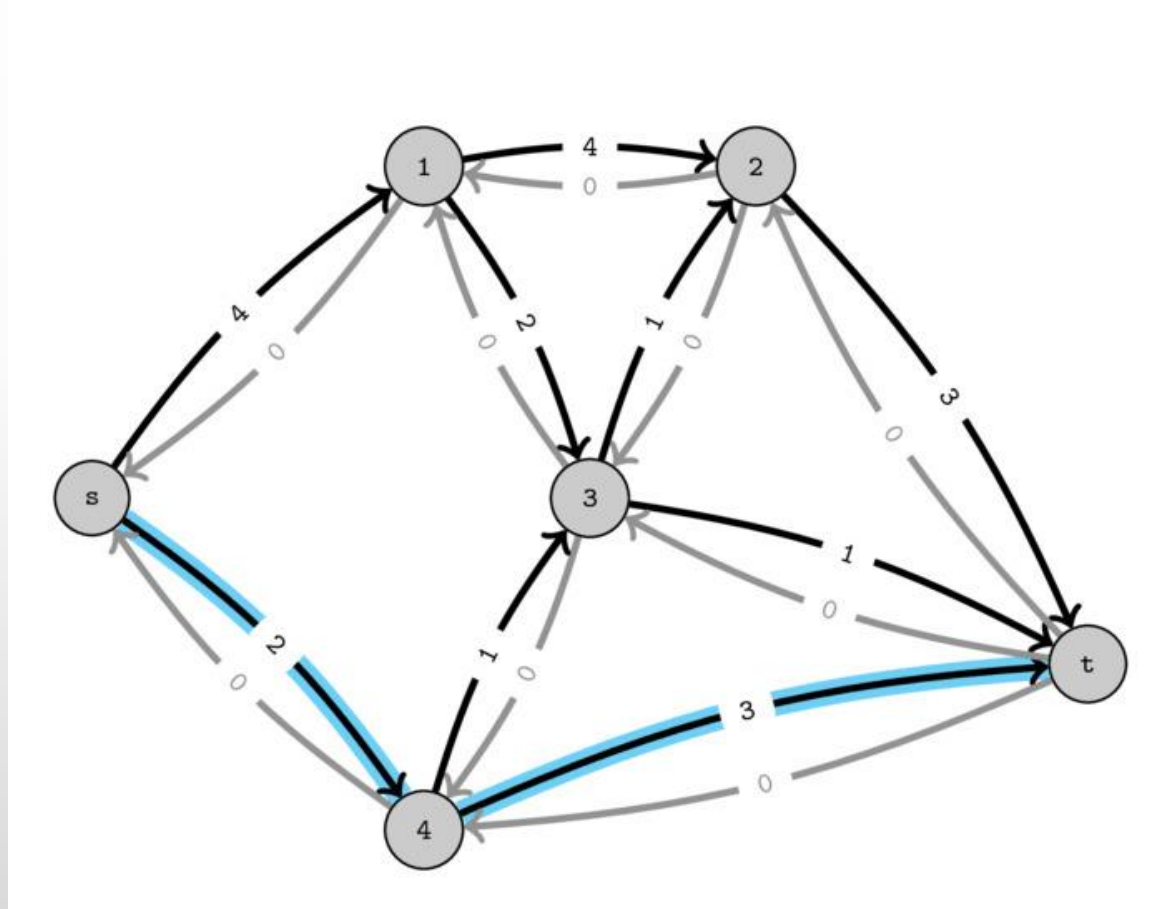


Edmonds Karp



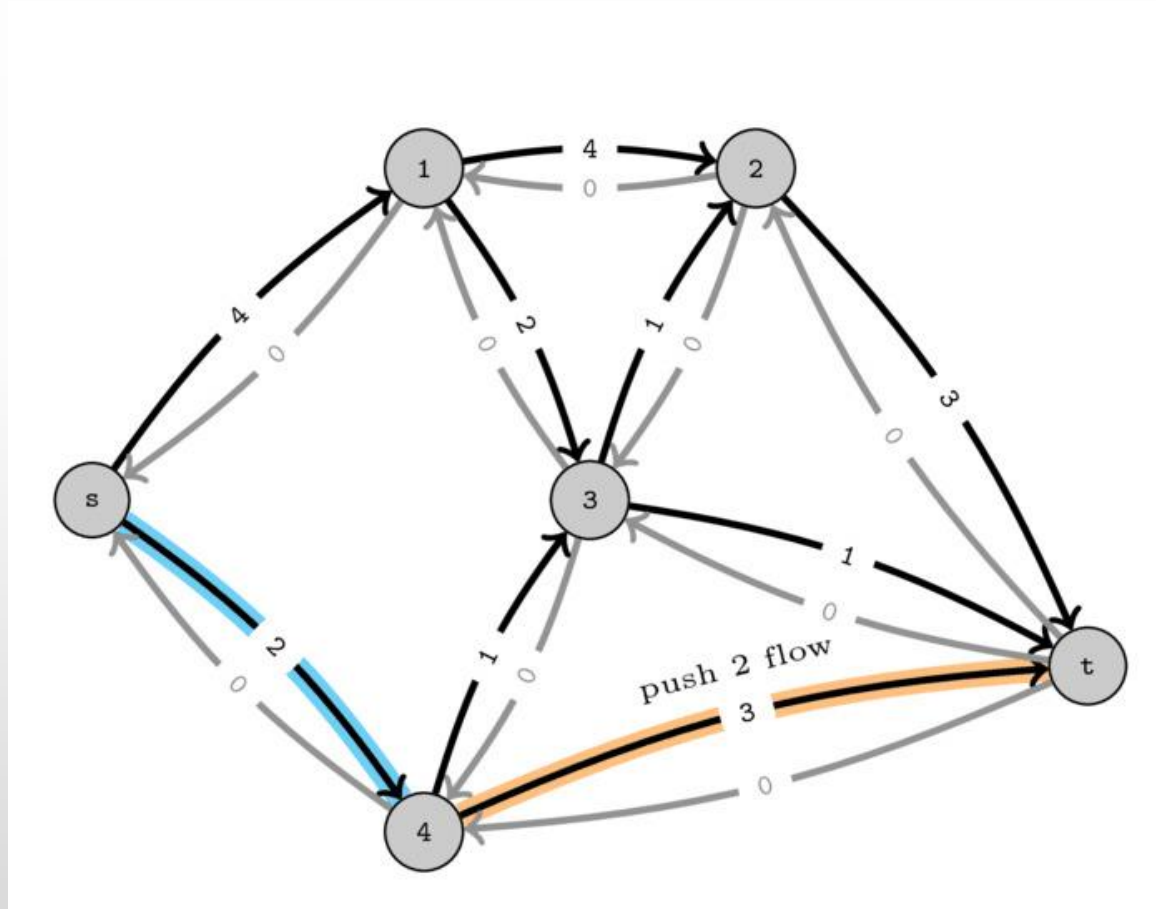


Edmonds Karp



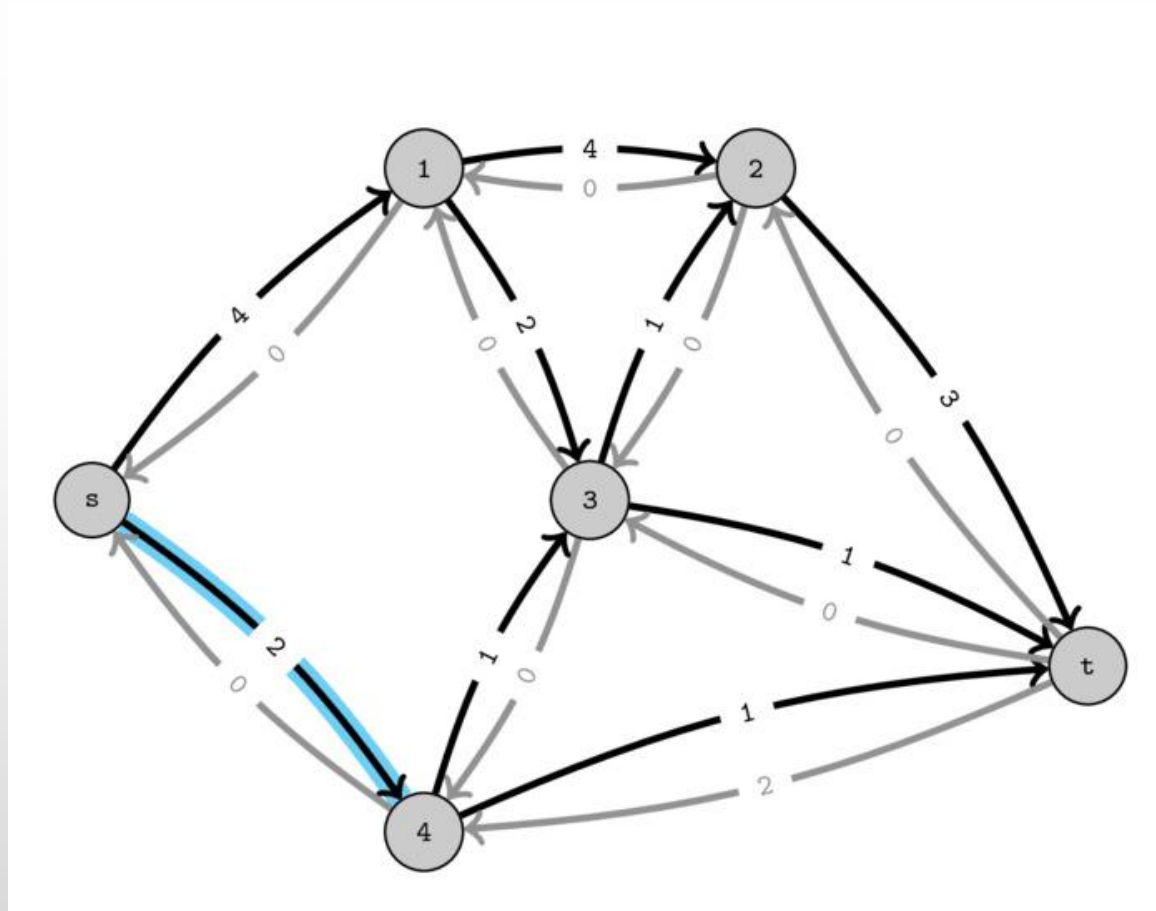


Edmonds Karp



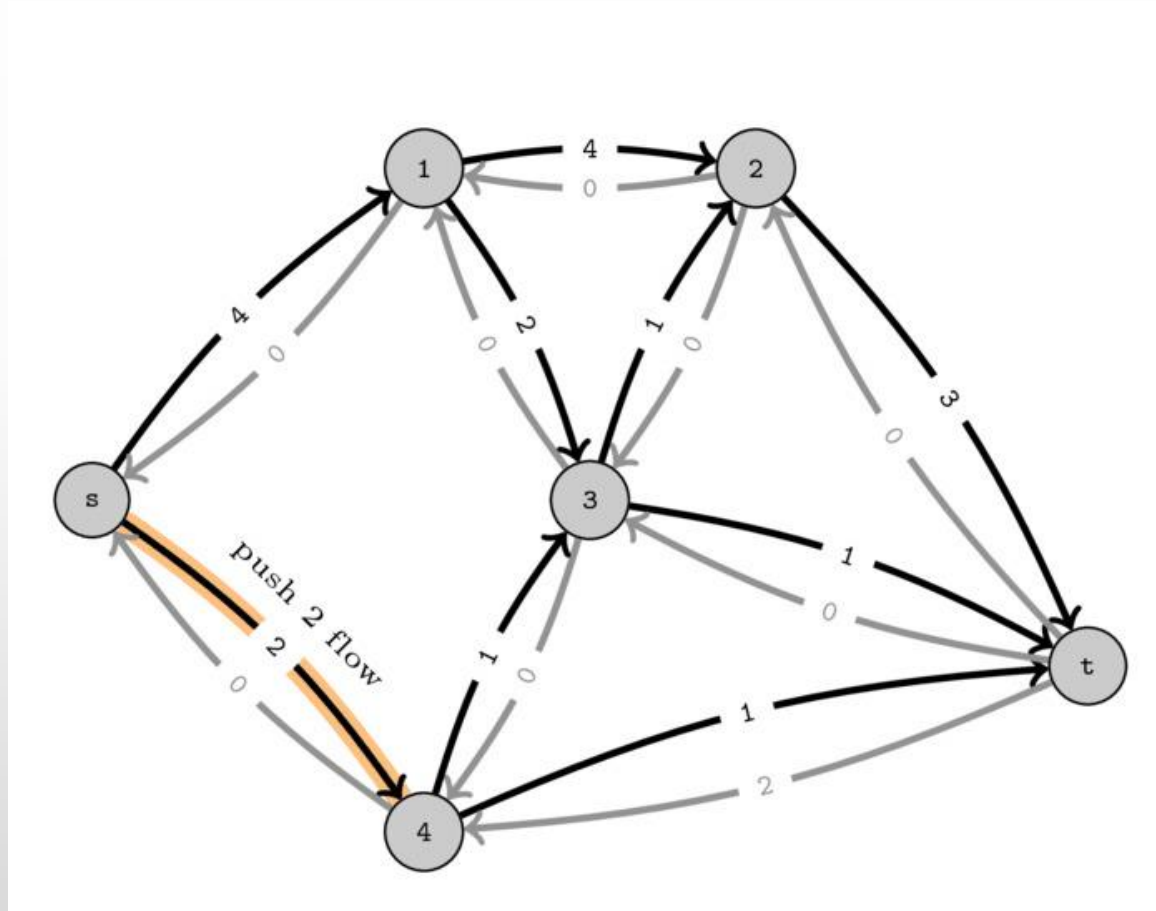


Edmonds Karp



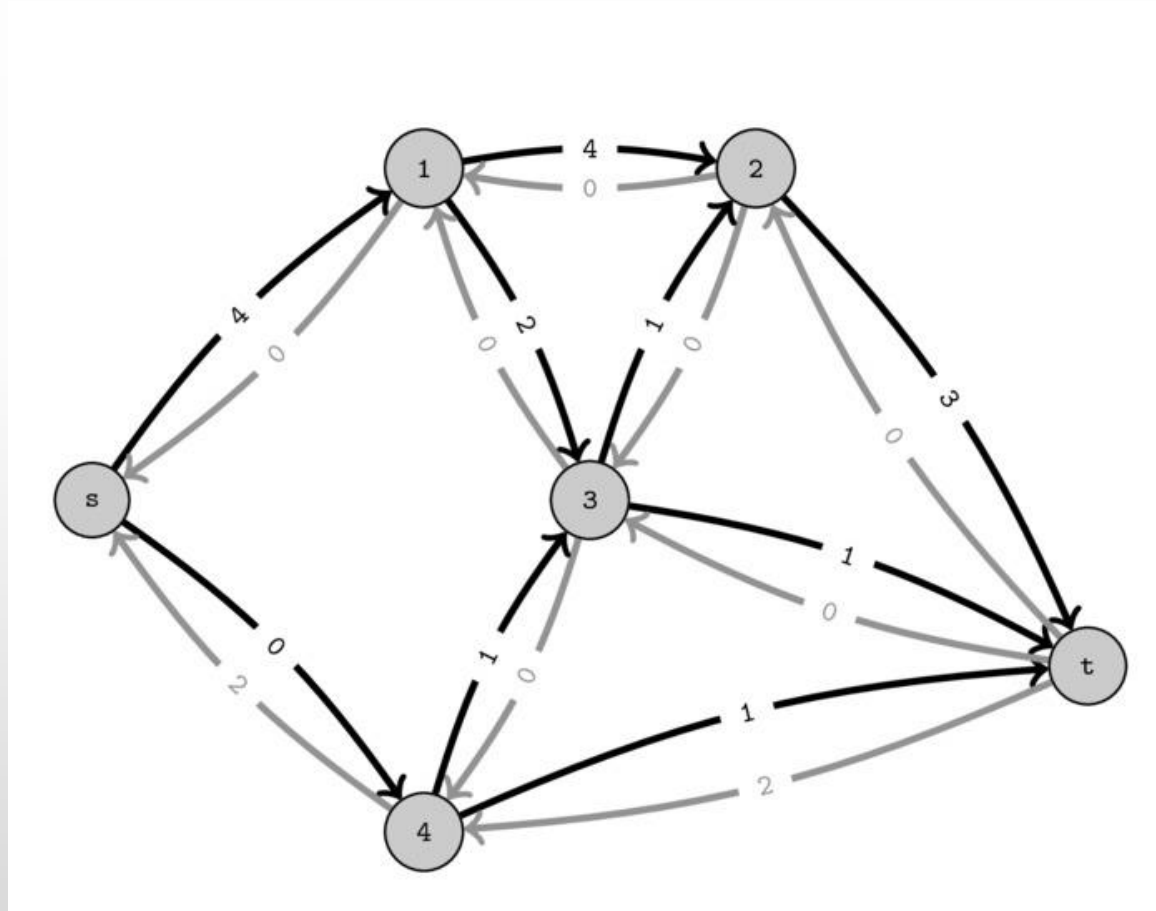


Edmonds Karp



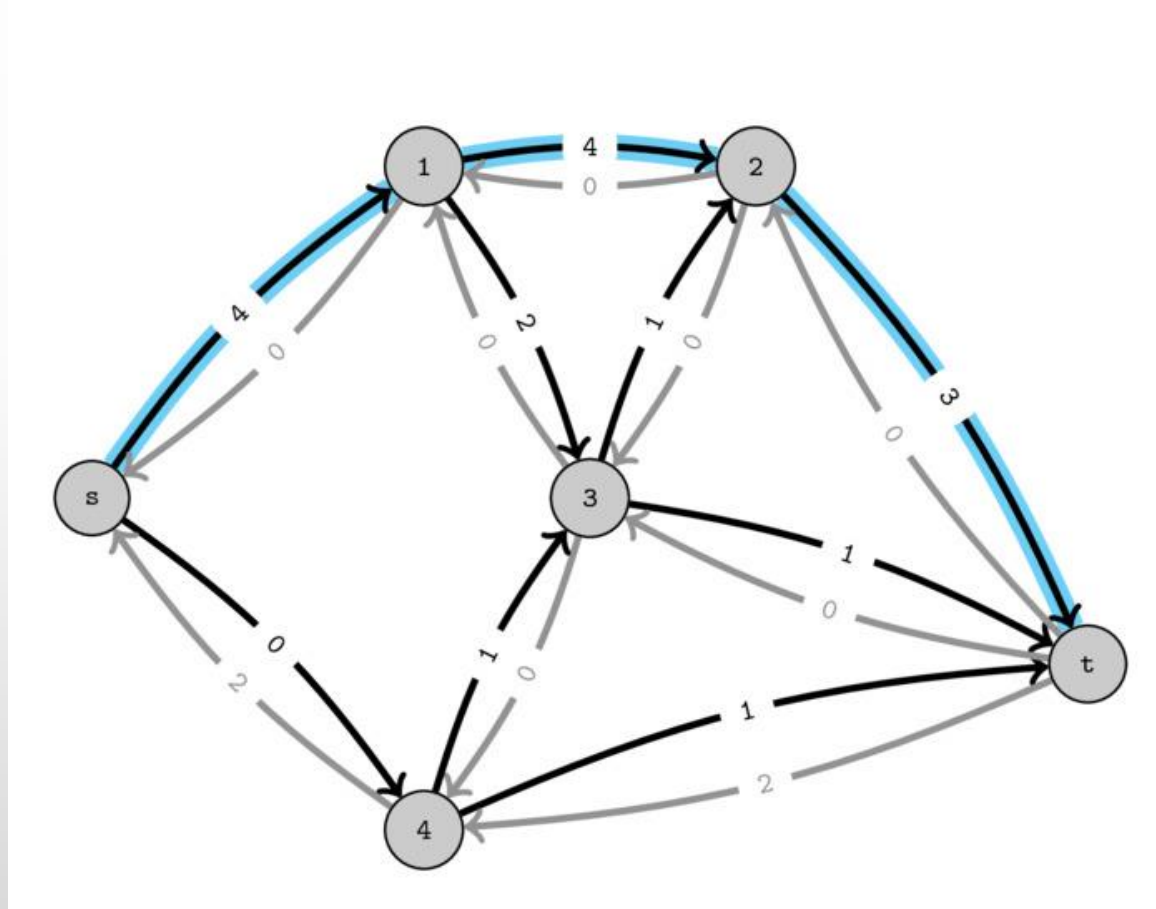


Edmonds Karp



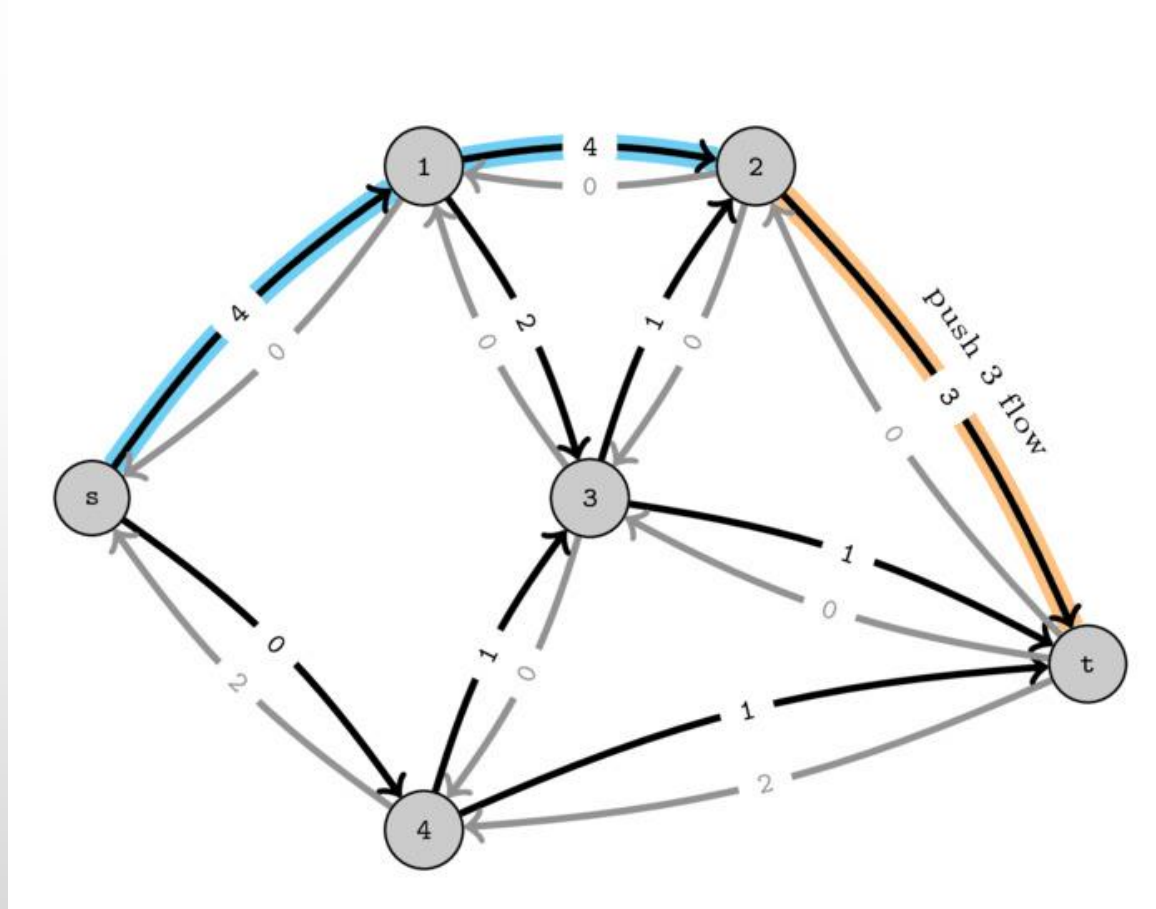


Edmonds Karp



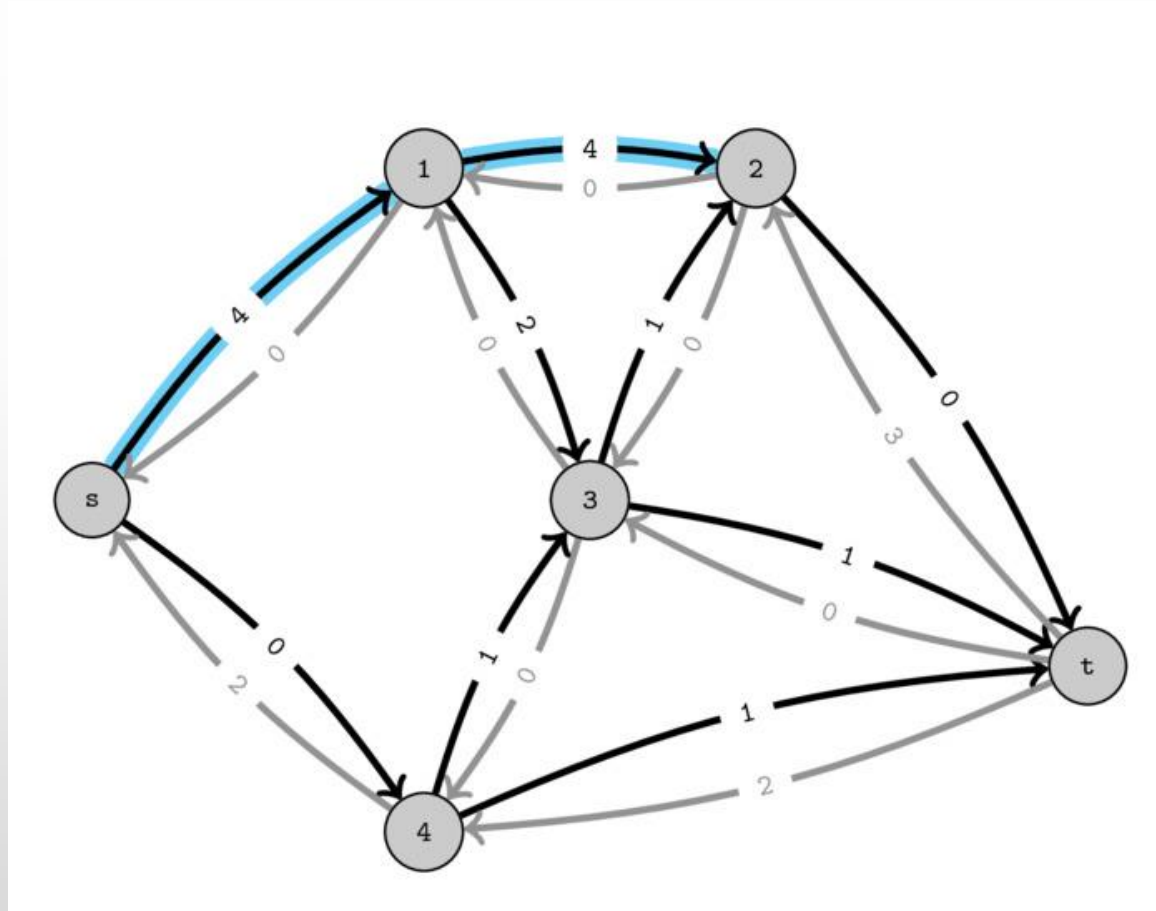


Edmonds Karp



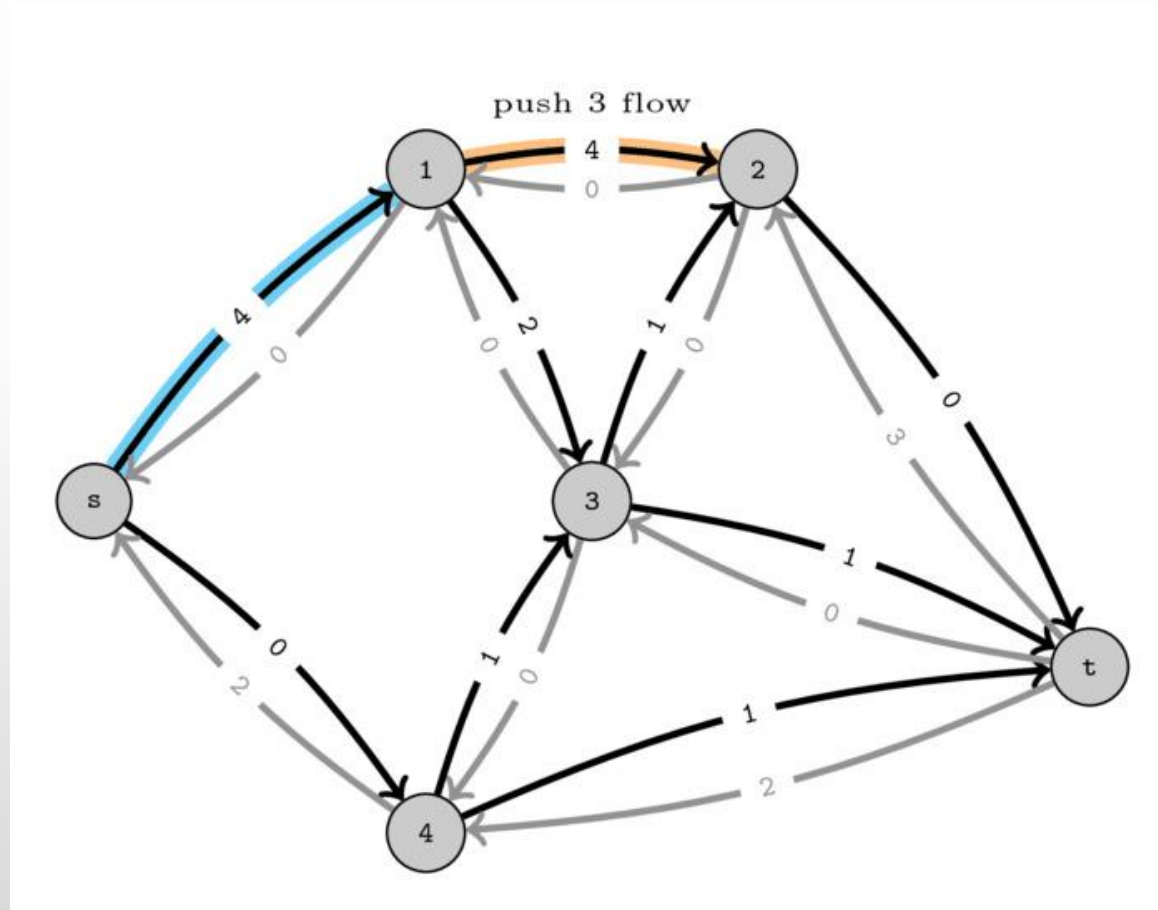


Edmonds Karp



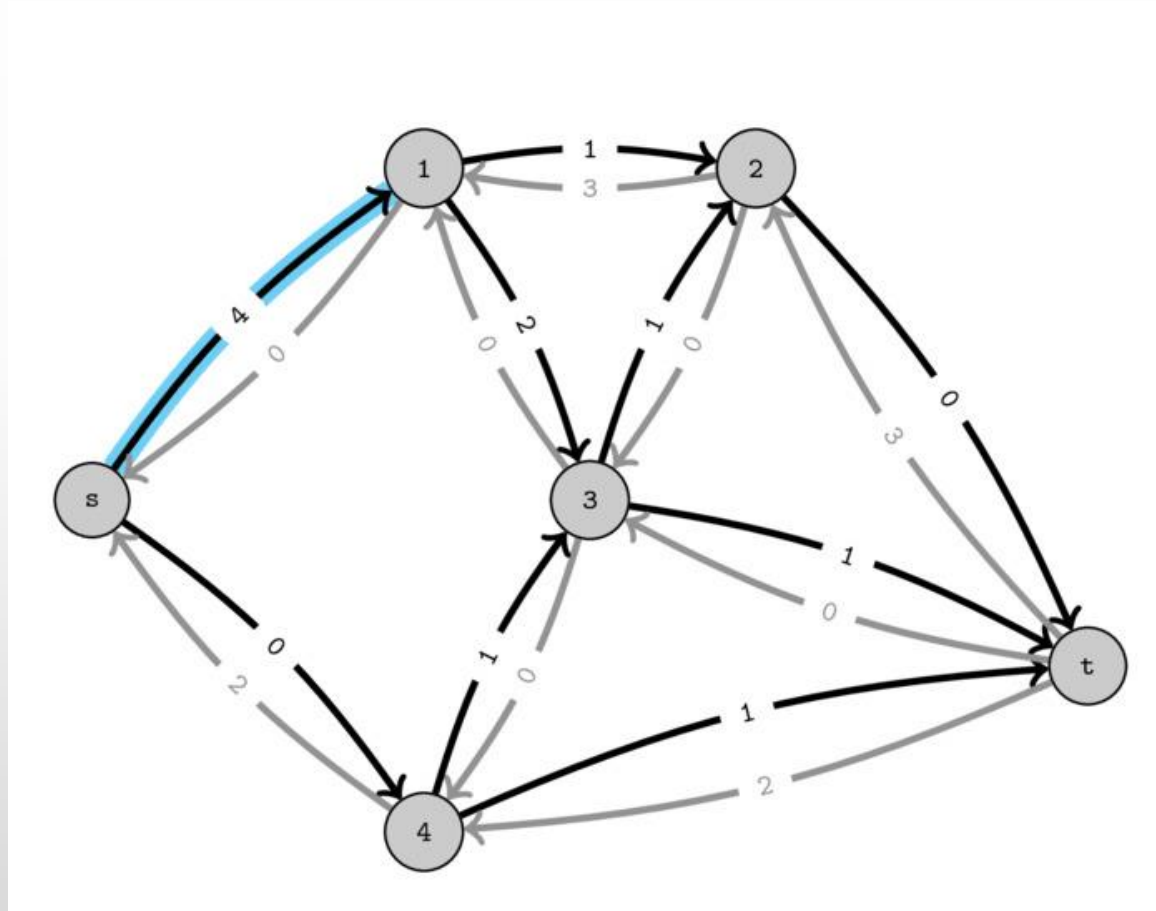


Edmonds Karp



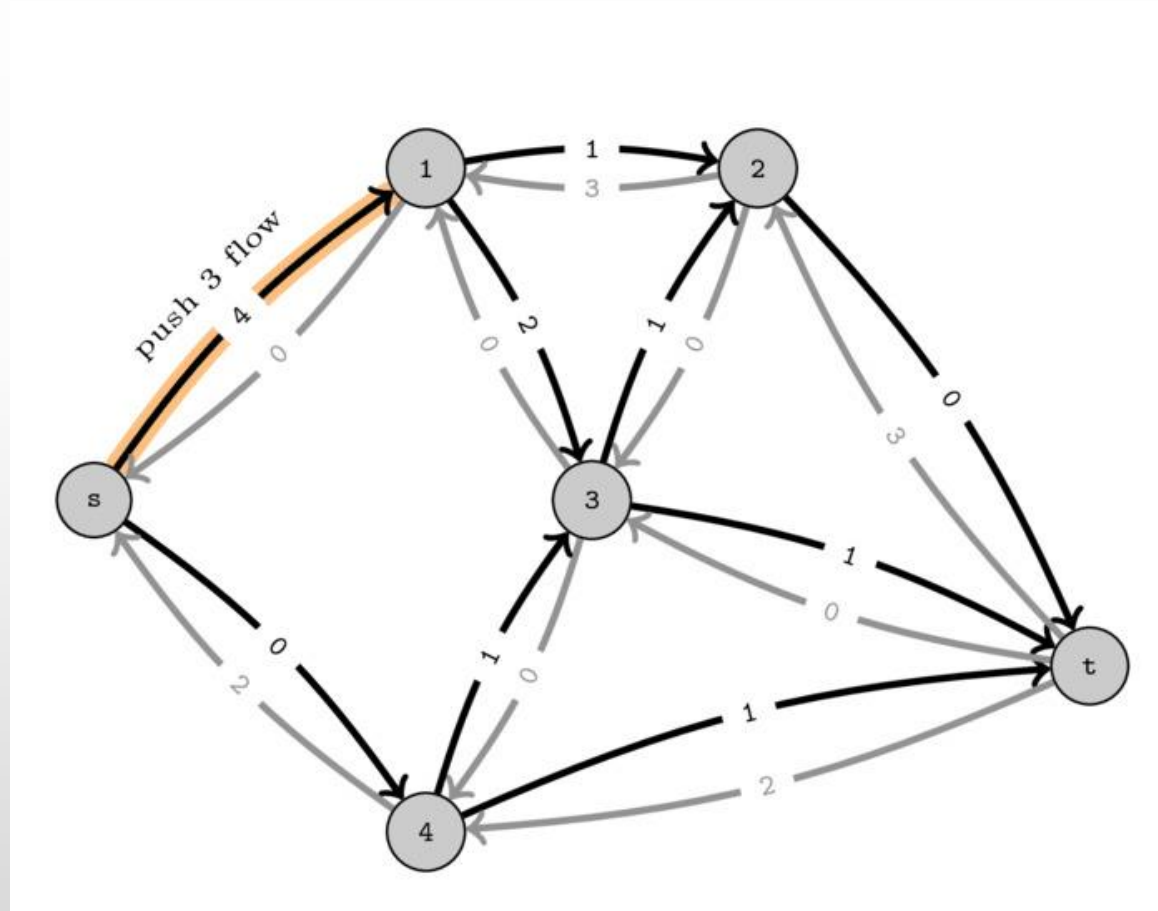


Edmonds Karp



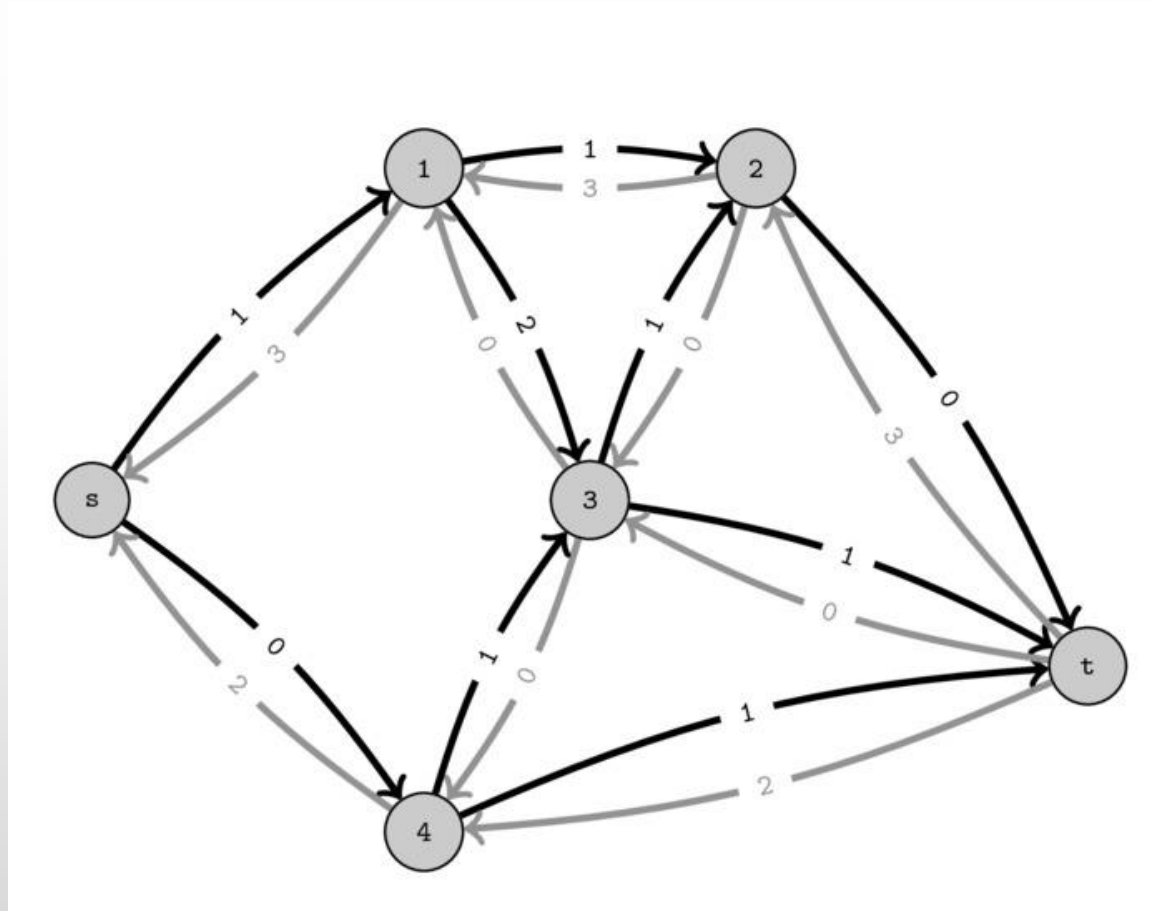


Edmonds Karp



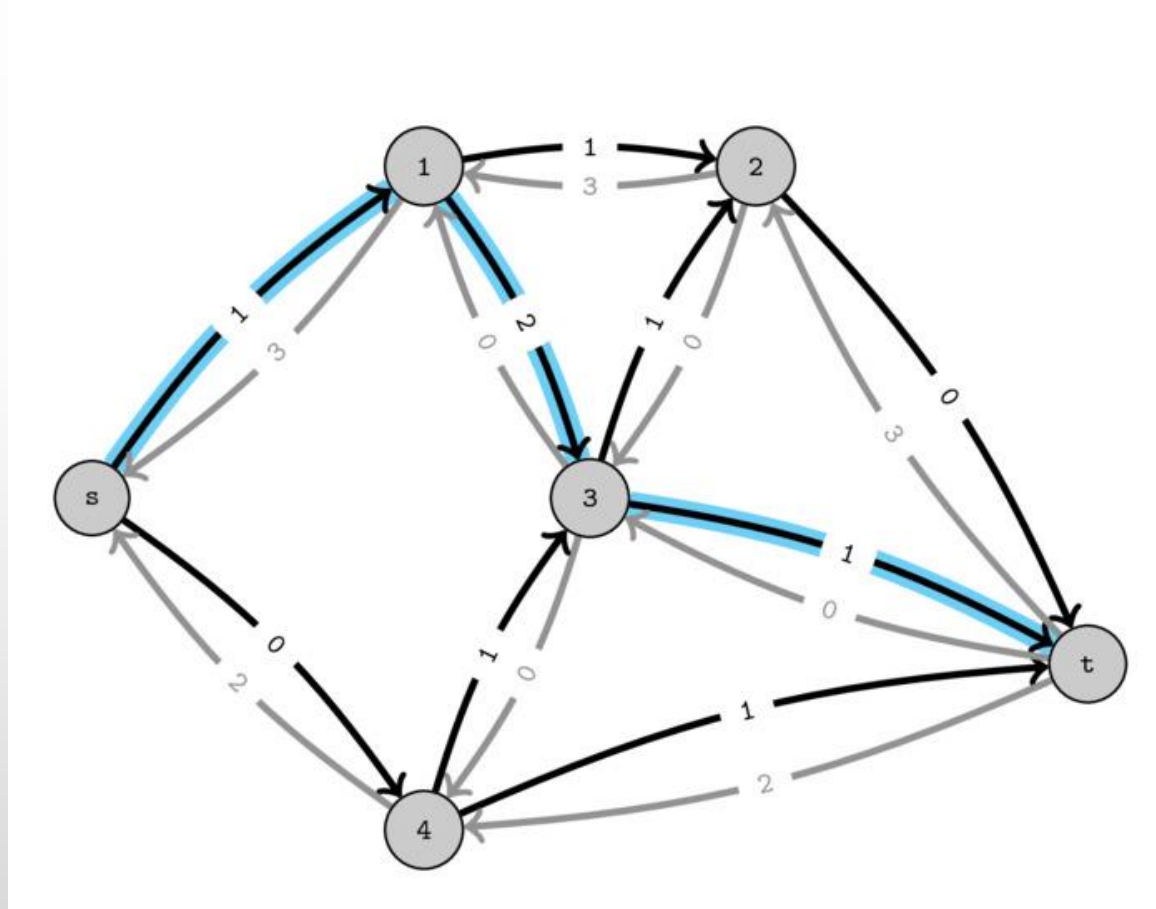


Edmonds Karp



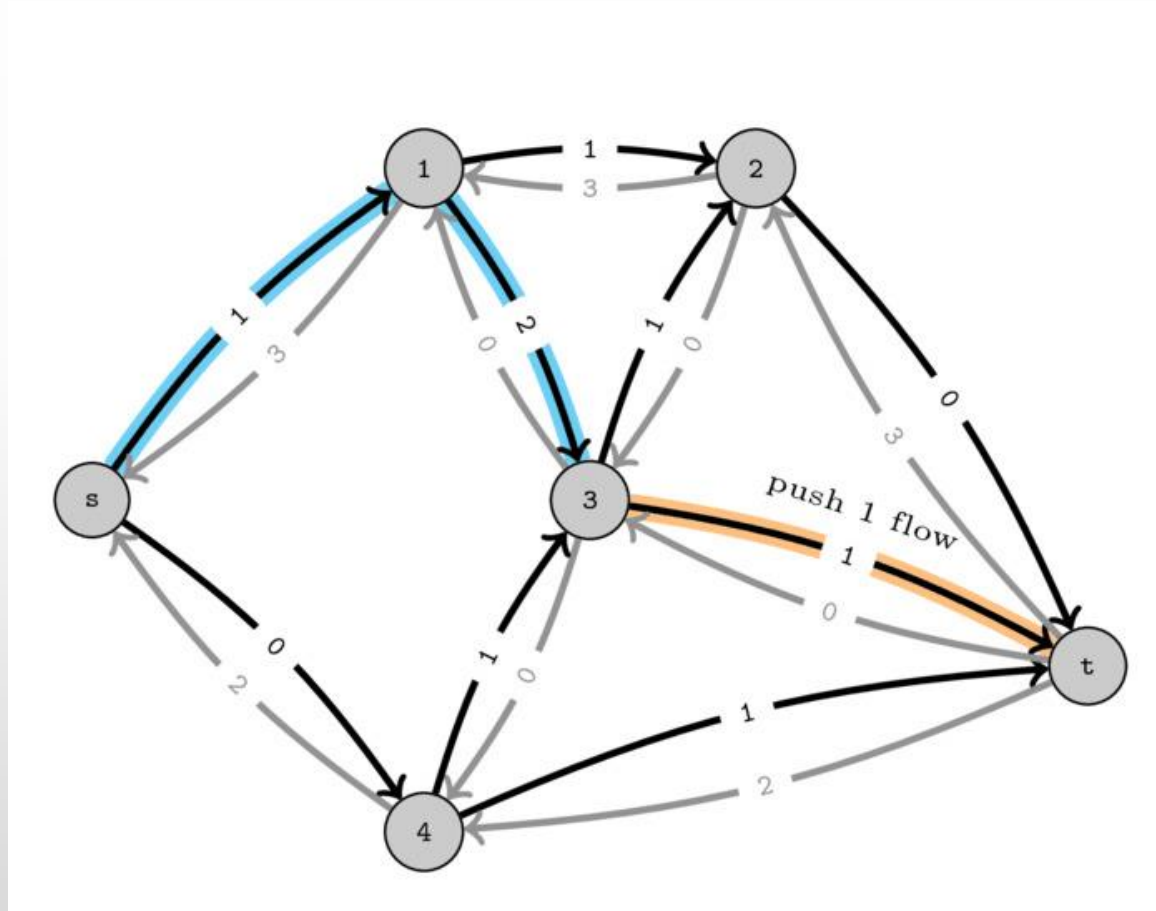


Edmonds Karp



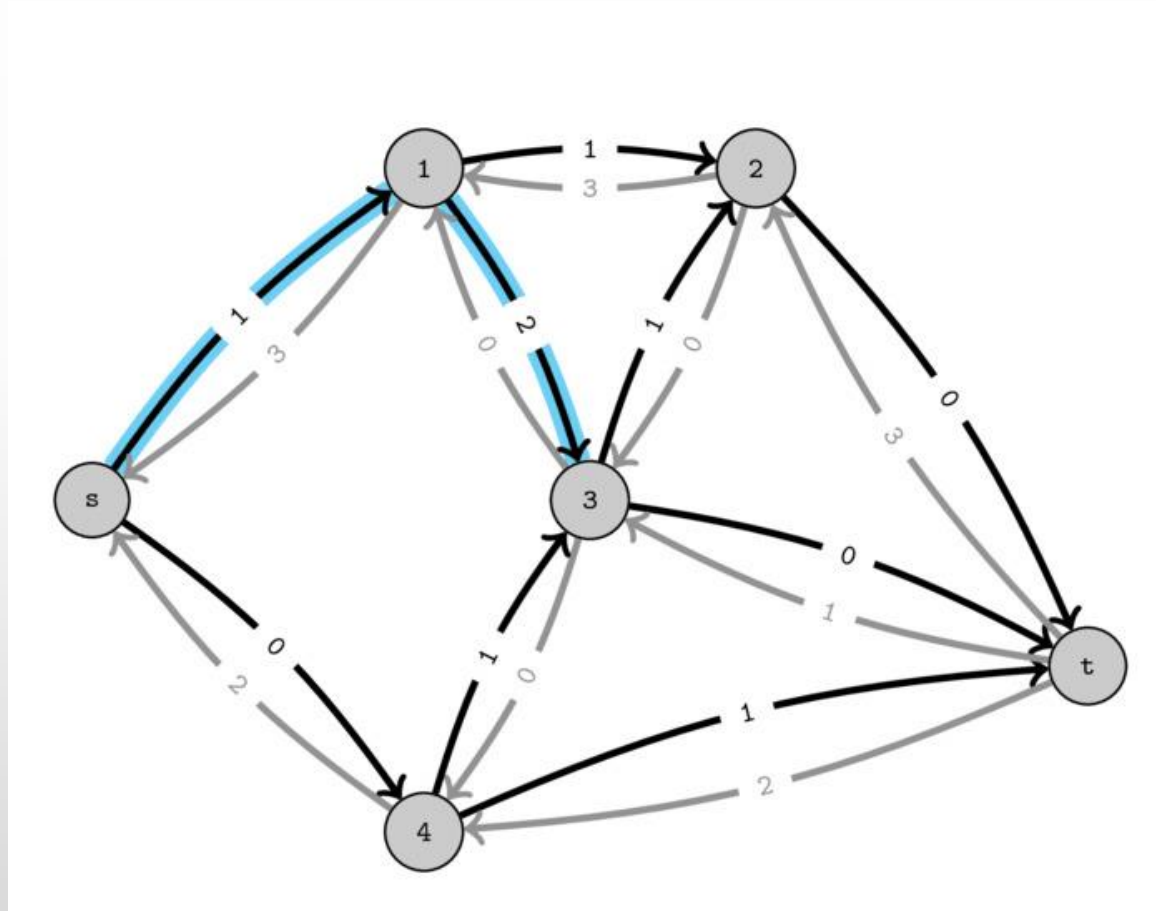


Edmonds Karp



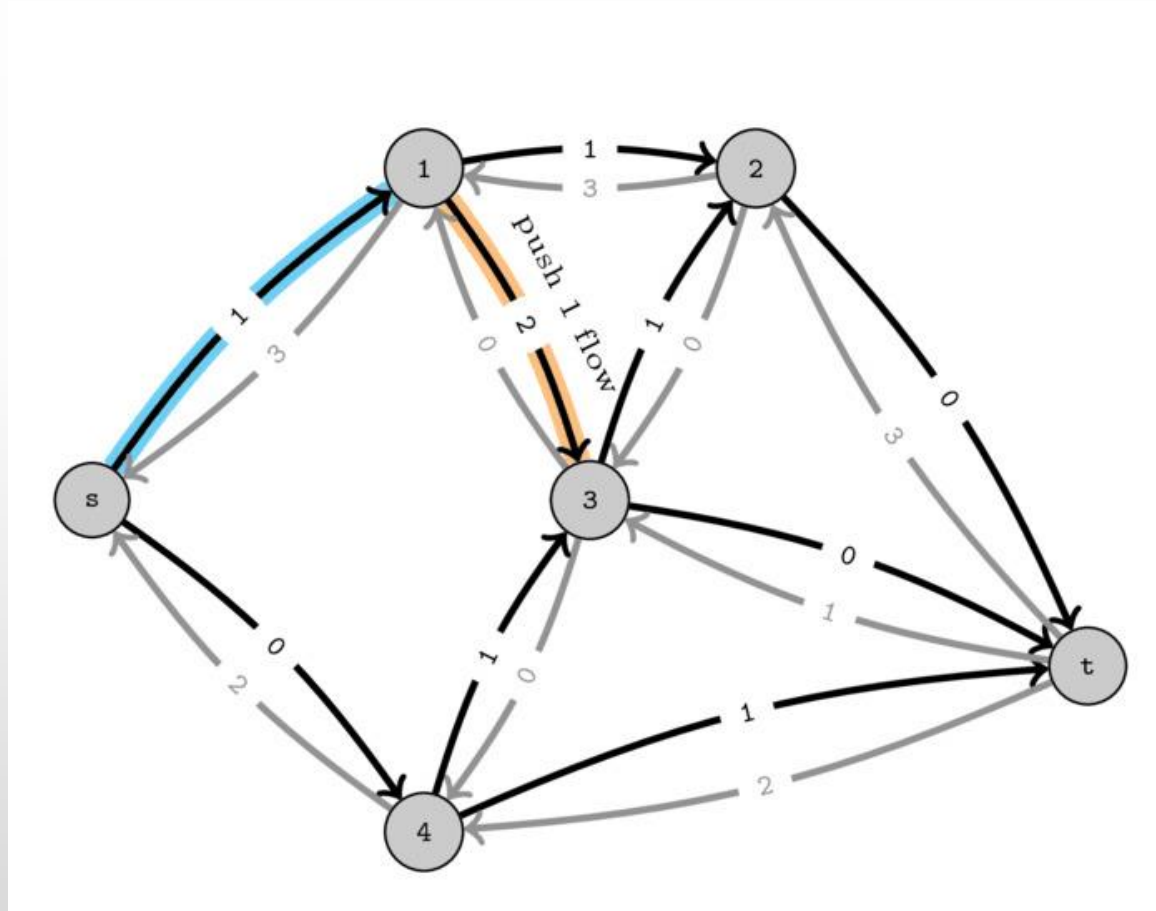


Edmonds Karp



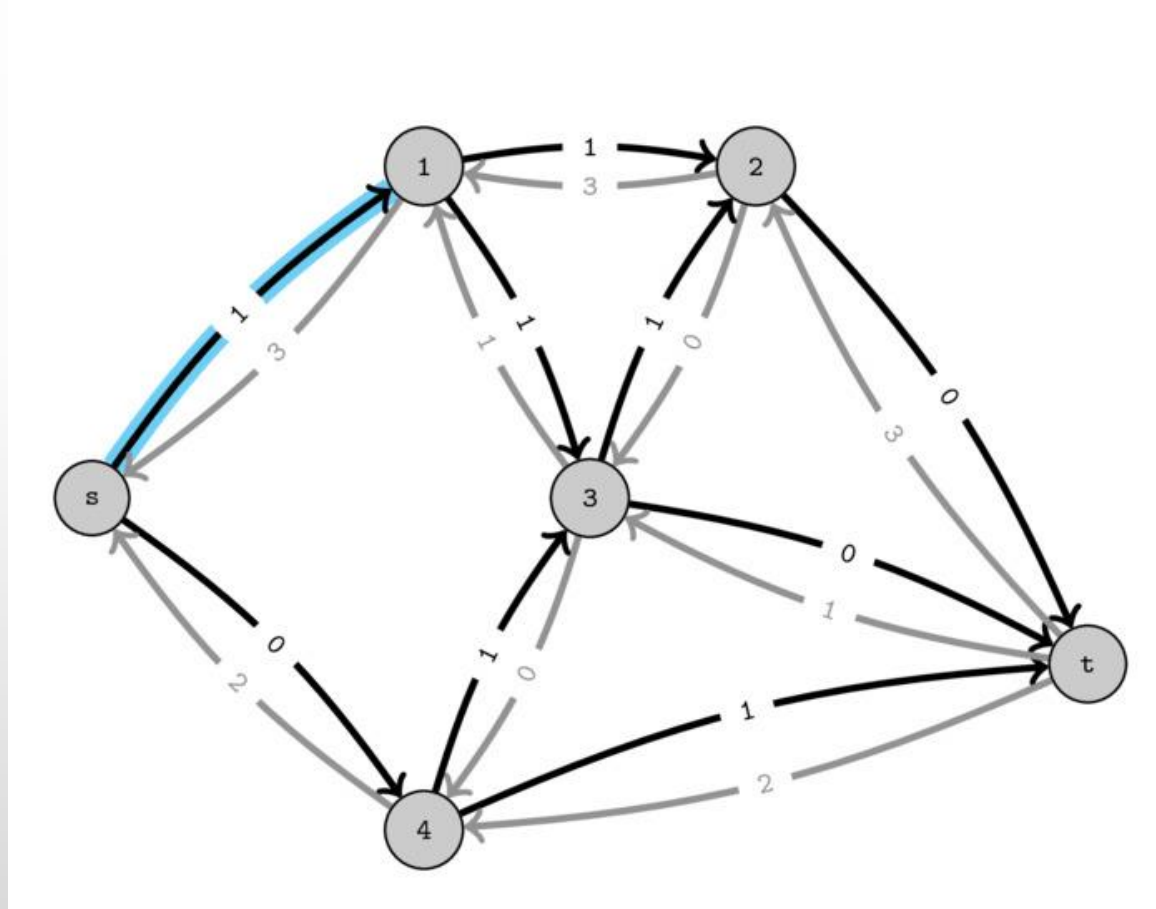


Edmonds Karp



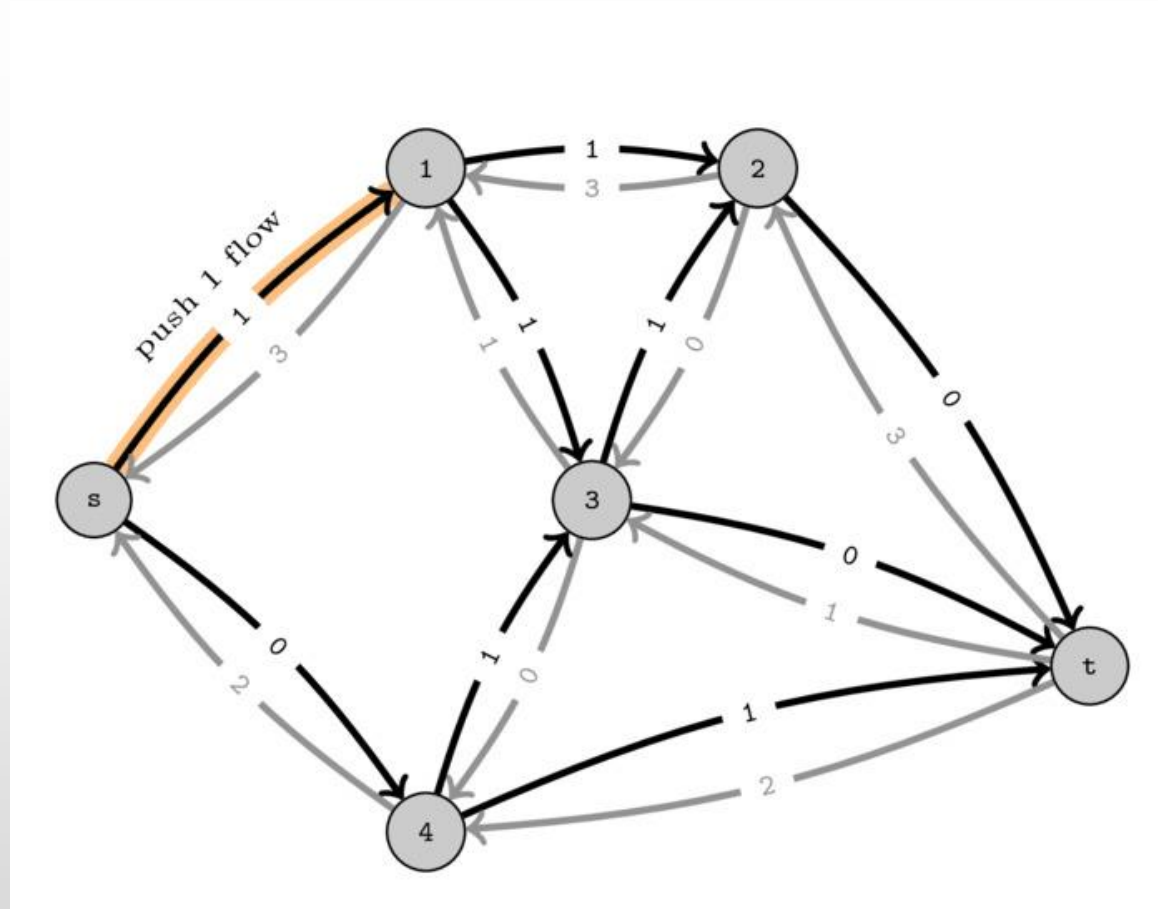


Edmonds Karp



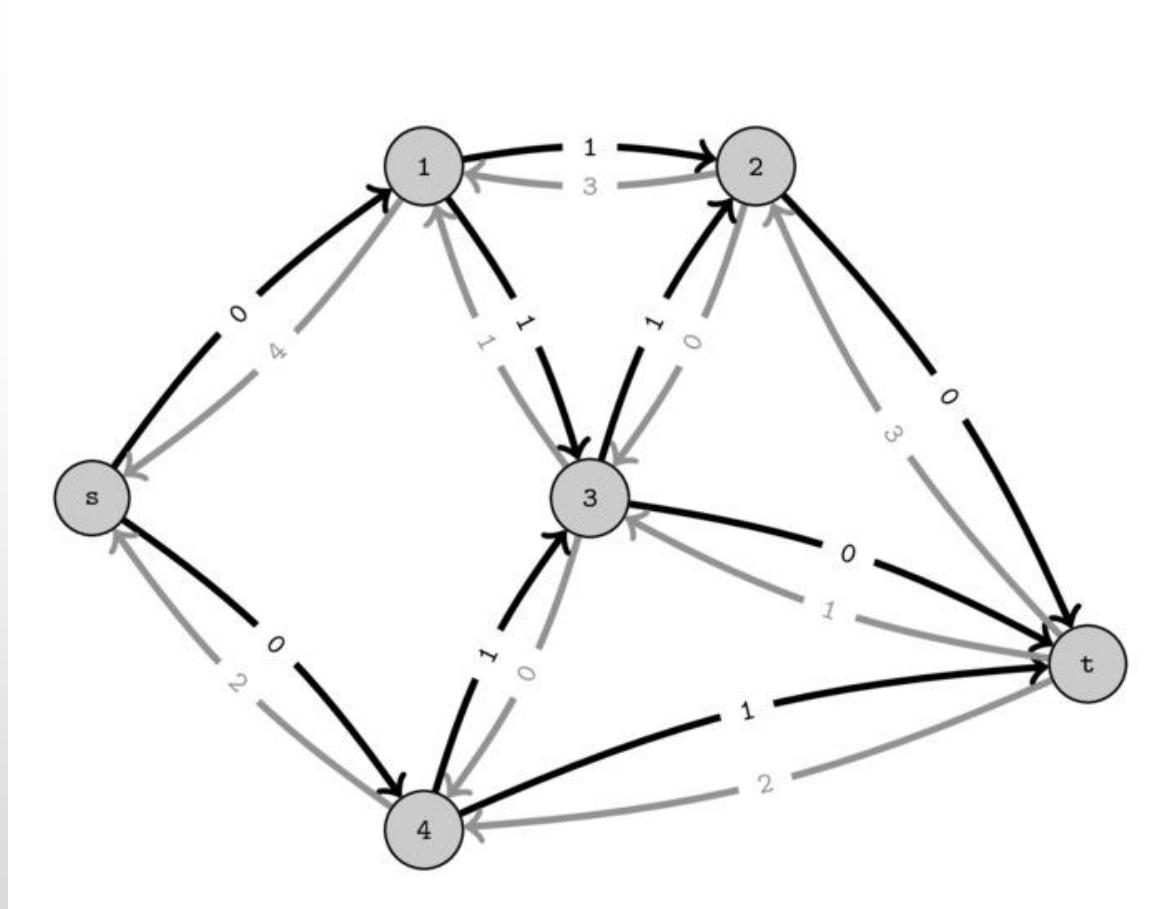


Edmonds Karp





Edmonds Karp







Dinic Algoritması

- Bir çizge içinde maksimum akışı bulur.
- Ağırlıklı çizge üzerinde çalışır.
- Karmaşıklığı; $O(V^2 E)$
 - Her aşama BFS gerektirir $O(V + E)$.
 - En kötü durumda V aşama olabilir.
 - BFS tarafından bulunan artış yollarının uzunluğu $O(V)$ olabilir.



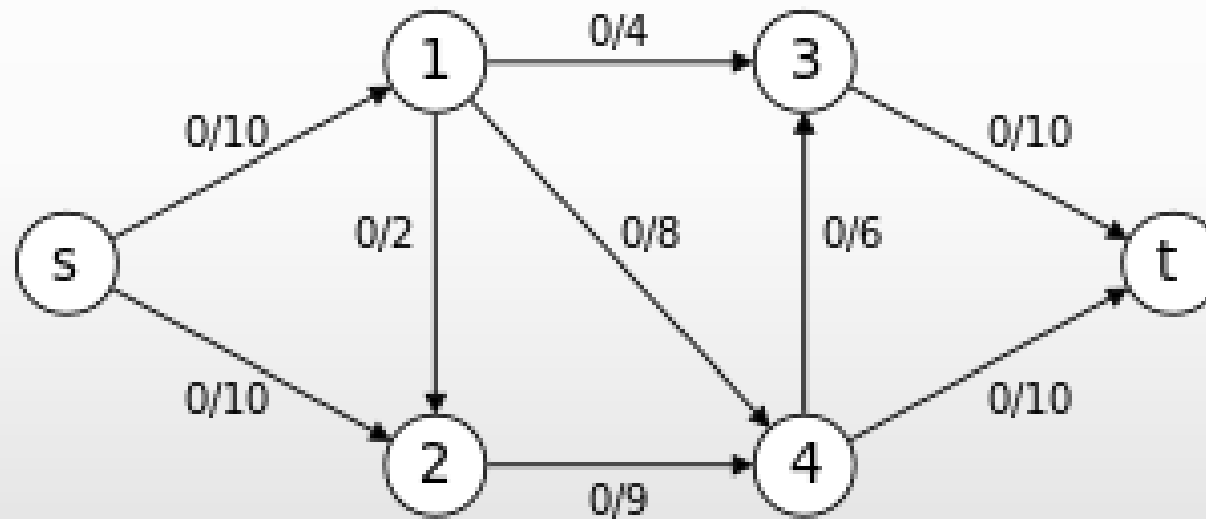
Dinic Algoritması

- Başlangıçta çizgenin tüm kenarlarının akışlarını sıfıra ayarla.
- BFS (Genişlik Öncelikli Arama) kullanarak artan yolları bul.
- Artan yollar üzerindeki minimum kapasiteyi bul ve toplam ağ akışına ekle.
- 2. adıma geri dön ve işlemleri tekrarla.



Dinic - Graph

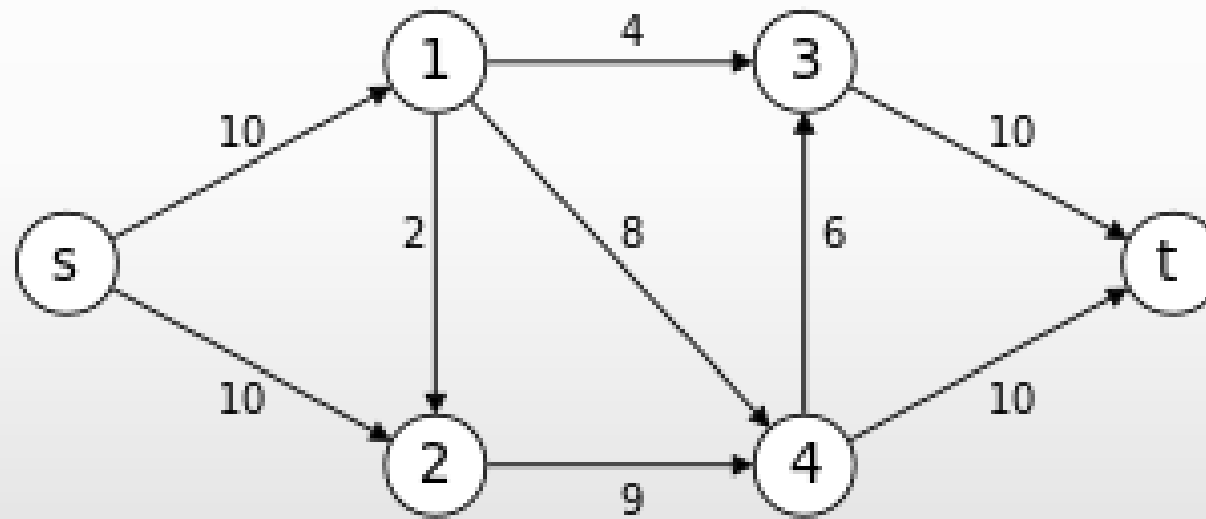
- $\{s,1,3,t\}$ 4 birim akış, $\{s,1,4,t\}$ 6 birim akış, $\{s,2,4,t\}$ 4 birim akış, $|f| = 14$.





Dinic - Graph_f

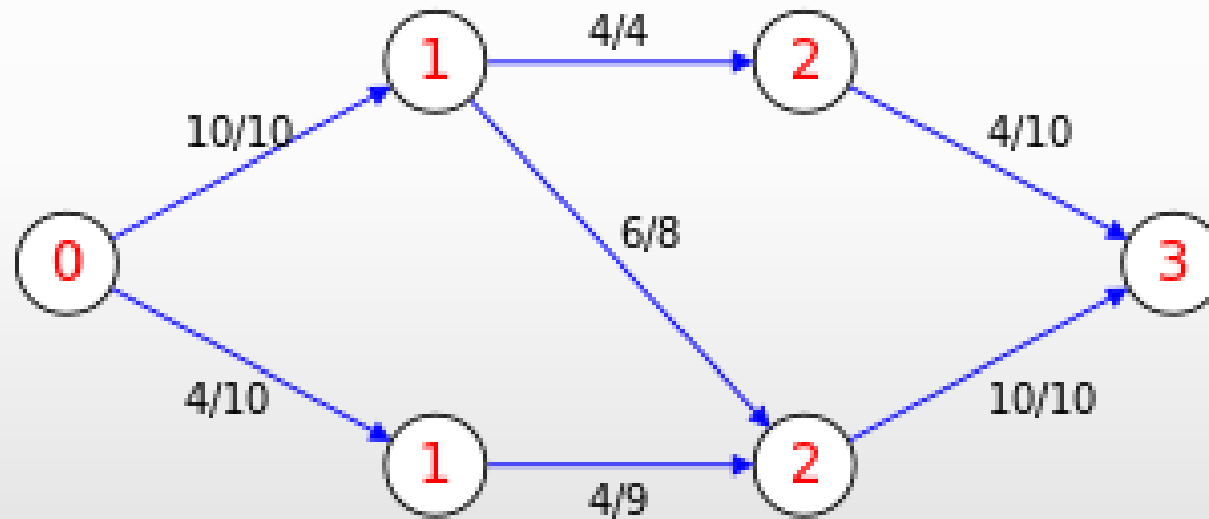
- akış çizgesi





Dinic - Graph_L

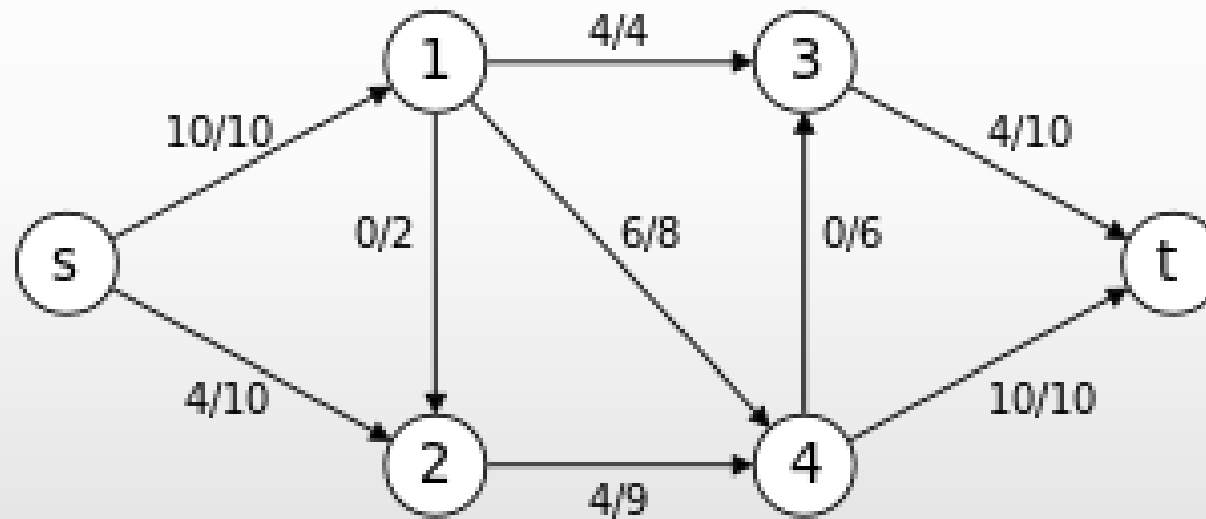
- level graph





Dinic - Graph

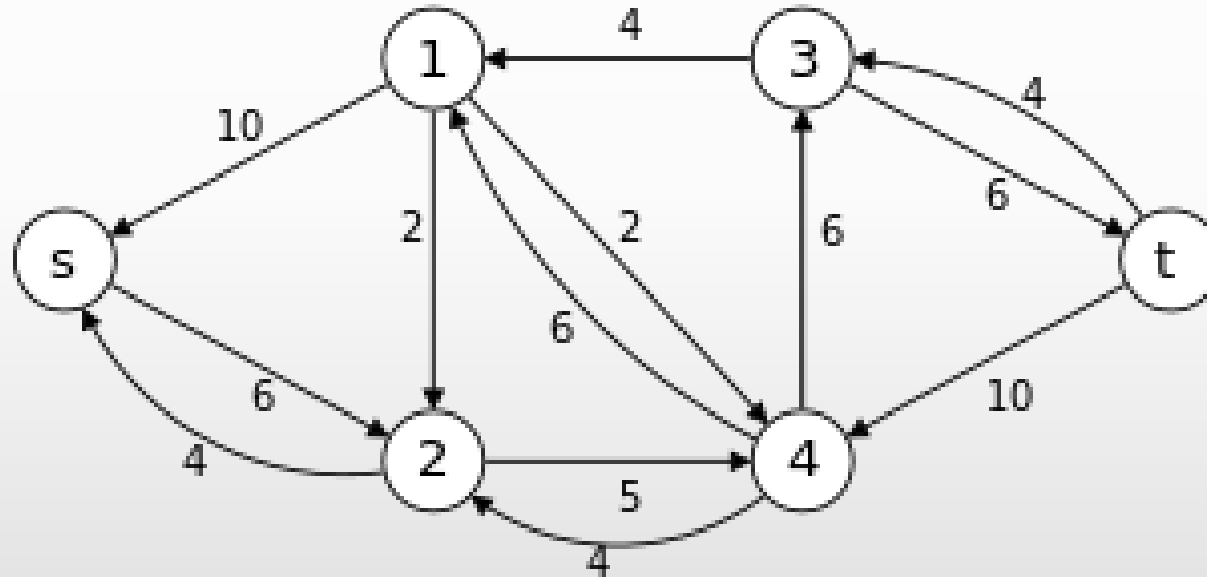
- $\{s, 2, 4, 3, t\}$ 5 birim akış, $|f| = 14 + 5 = 19$.





Dinic - Graph_f

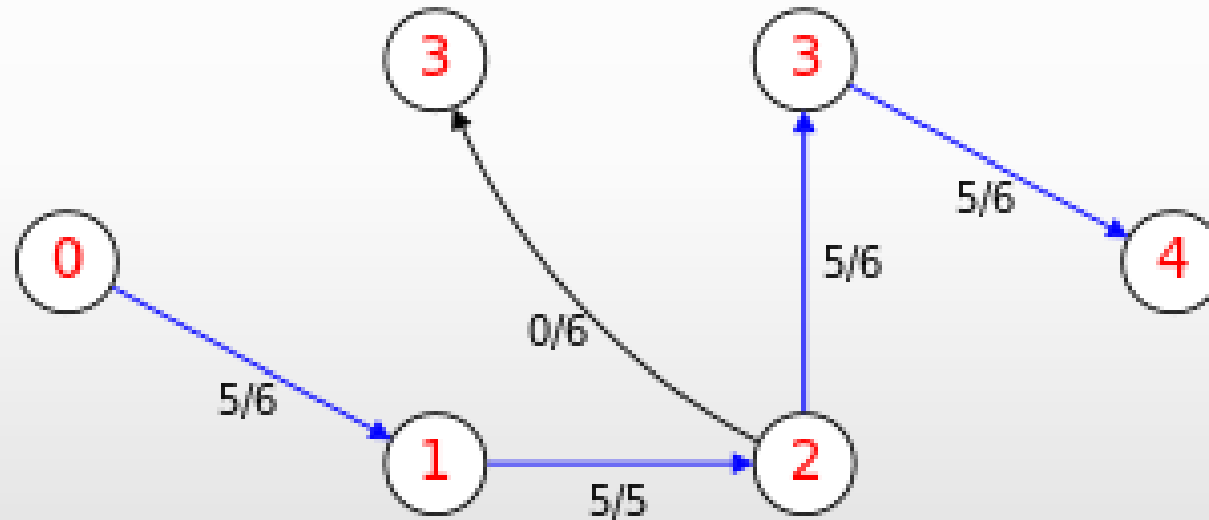
- akış çizgesi





Dinic - Graph_L

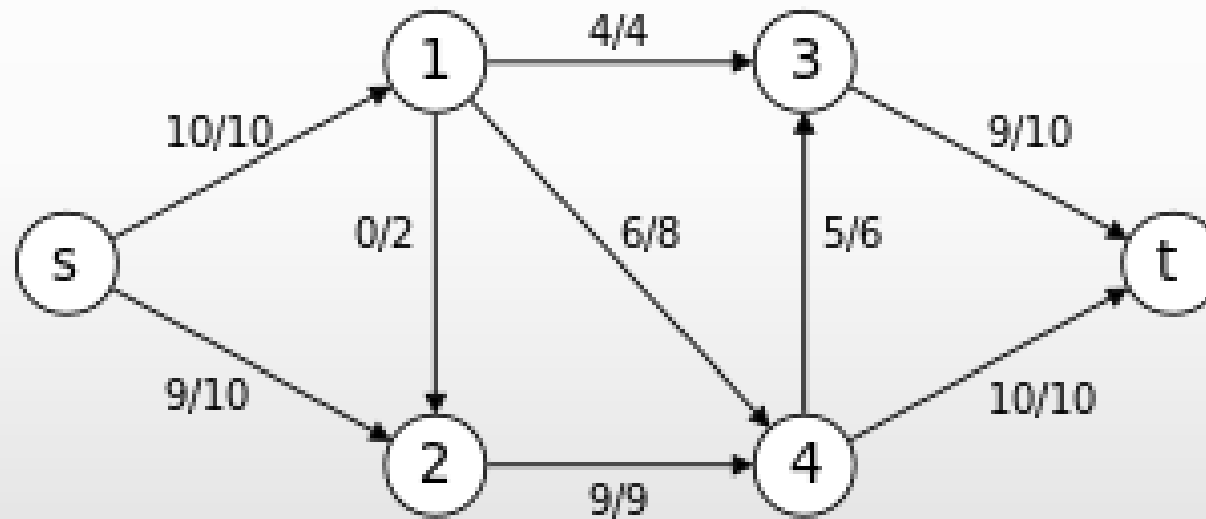
- level graph





Dinic - Graph

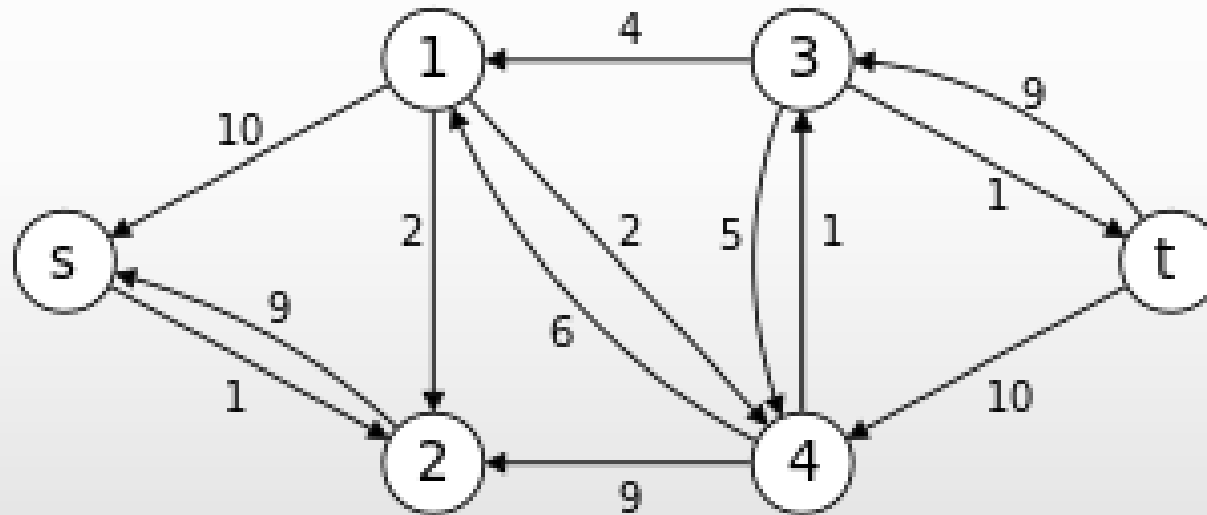
- akış çizgesine bakılarak t'ye erişen başka yol yok. algorithm terminates





Dinic - Graph_f

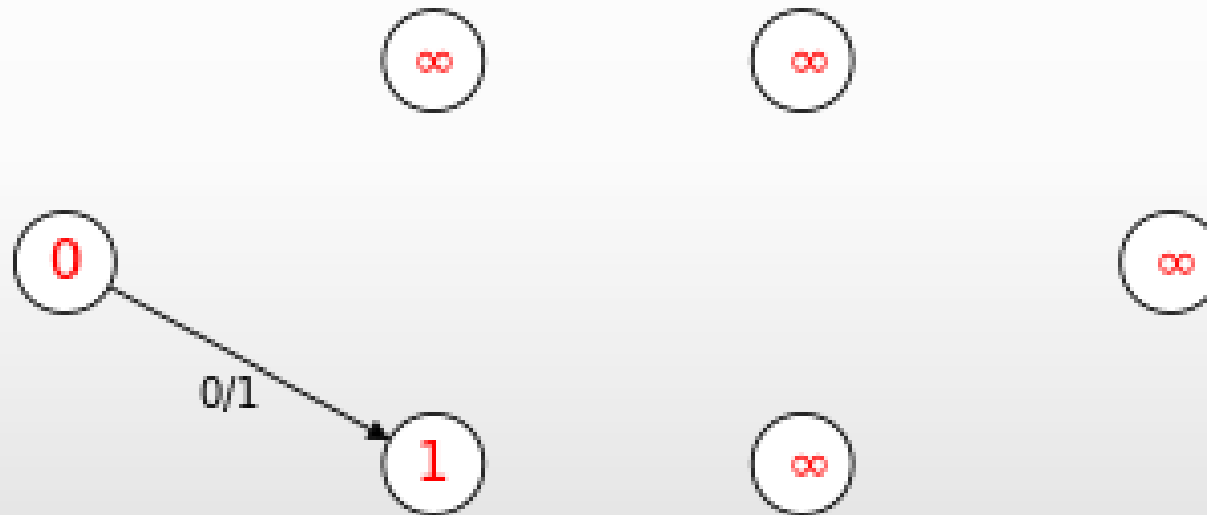
- akış çizgesi





Dinic - Graph_L

- level graph







Push Relabel Algorithm

- Her düğümün bir yükseklik değeri vardır.
- Akışı artırmak için yükseklik değerleri ve kenar kapasiteleri dikkate alınır.
- Düşükten yükseğe itme (push) ve yüksekte düşüğe etiketleme (relabel) adımları yapılır.



Push Relabel Algorithm

- Başlangıçta, tüm kenarların akışını sıfırlar.
- Kaynak düğümünün yüksekliğini diğer düğümlerden bir fazla yapar.
- İki adımdan oluşur.
 - İtme işlemi:
 - Yükseklik önceliğine göre düğümler arasında akışı iter.
 - Etiketleme işlemi:
 - Çizgenin yüksekliğini günceller ve
 - İtme işlemine devam eder.



İtme İşlemi (Push)

- Bir düğümün yüksekliği, bitiş düğümüne en kısa yolun uzunluğuna eşittir.
- Eğer bir kenar üzerinde potansiyel bir akış varsa (kenar kapasitesi ile akışın toplamı arasında fark varsa), akış artırılır.



Etiketleme İşlemi (Relabel)

- Bir düğümün yüksekliği, bitiş düğümüne olan en kısa yolun uzunluğuna eşit değilse, yükseklik değeri güncellenir.
- Potansiyel bir itme işlemi için yeni bir yol bulmak için yapılır.

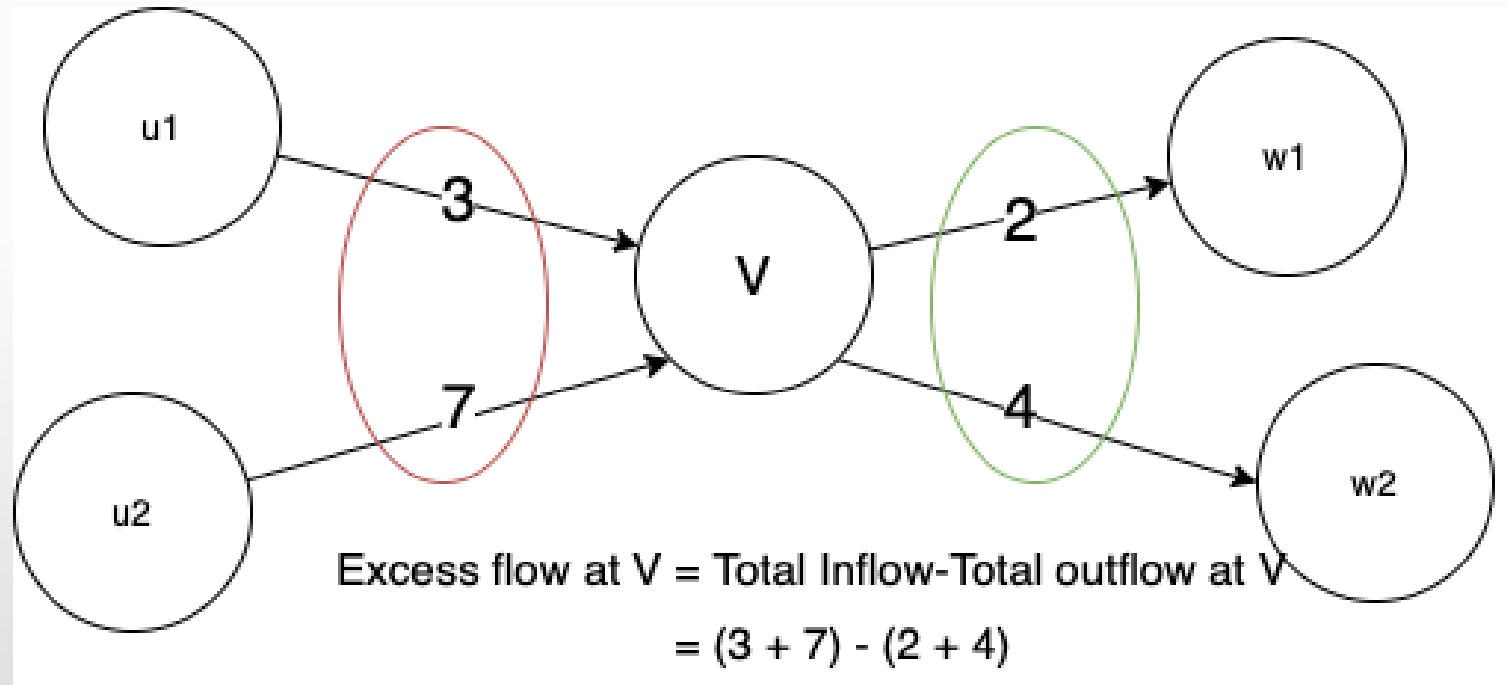


Algoritma Karmaşıklığı

- En kötü durumda $O(V^2 E)$ karmaşıklığa sahiptir.
- Her itme işlemi bir kenarı taramak ve potansiyel olarak akışı artırmak için birçok adım gerektirebilir. $O(V)$
- Etiketleme işlemi, bir düğümün yüksekliğini güncellemek için kullanılır. Her düğüm için tüm kenarları kontrol etmek zorunda kalabilir. $O(V E)$
- Her itme ve etiketleme işlemi, tüm düğümleri ve kenarları tarayabilir. Toplam işlem sayısı, tüm düğümler ve kenarlar üzerinde yapılacak işlemlerin toplamıdır.



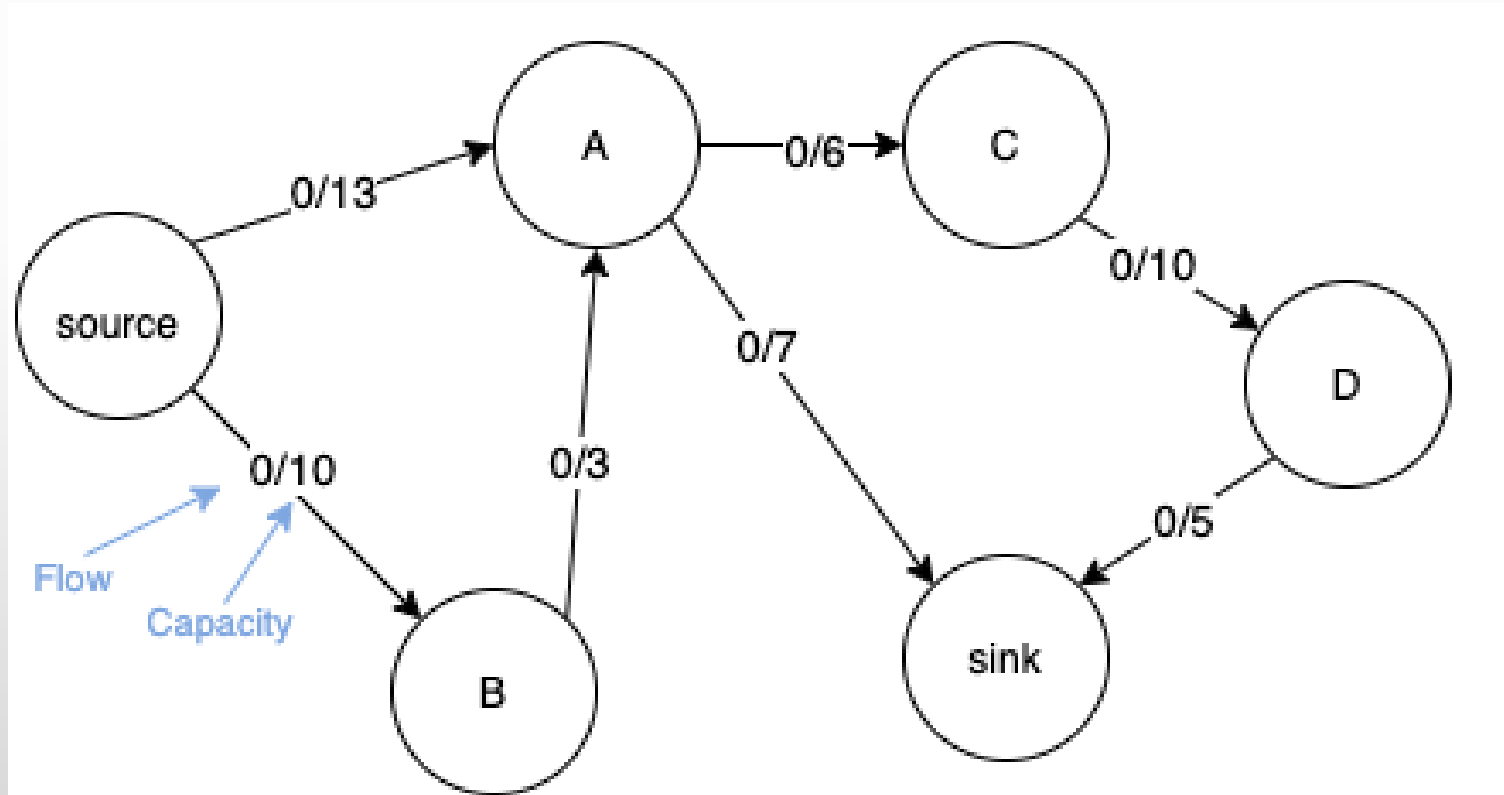
Excess Flow





Push Relabel

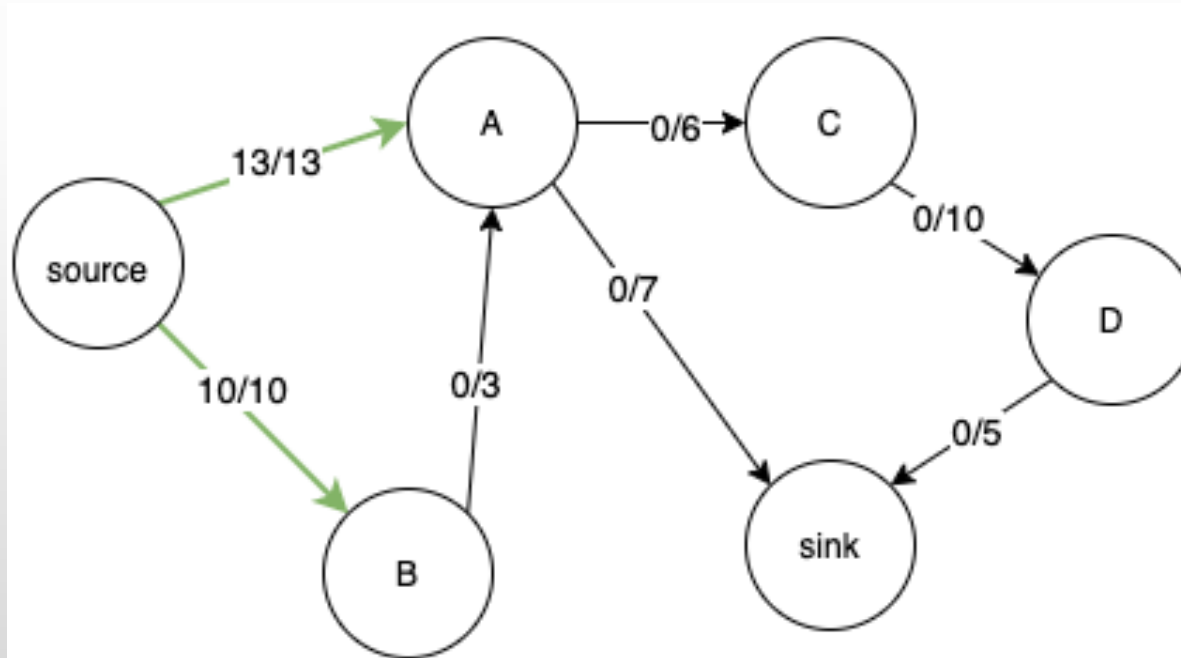
- Aşağıdaki çizge verilmiş olsun.





Push Relabel

- Yükseklik ve artık akış başlangıç değerleri atanır.

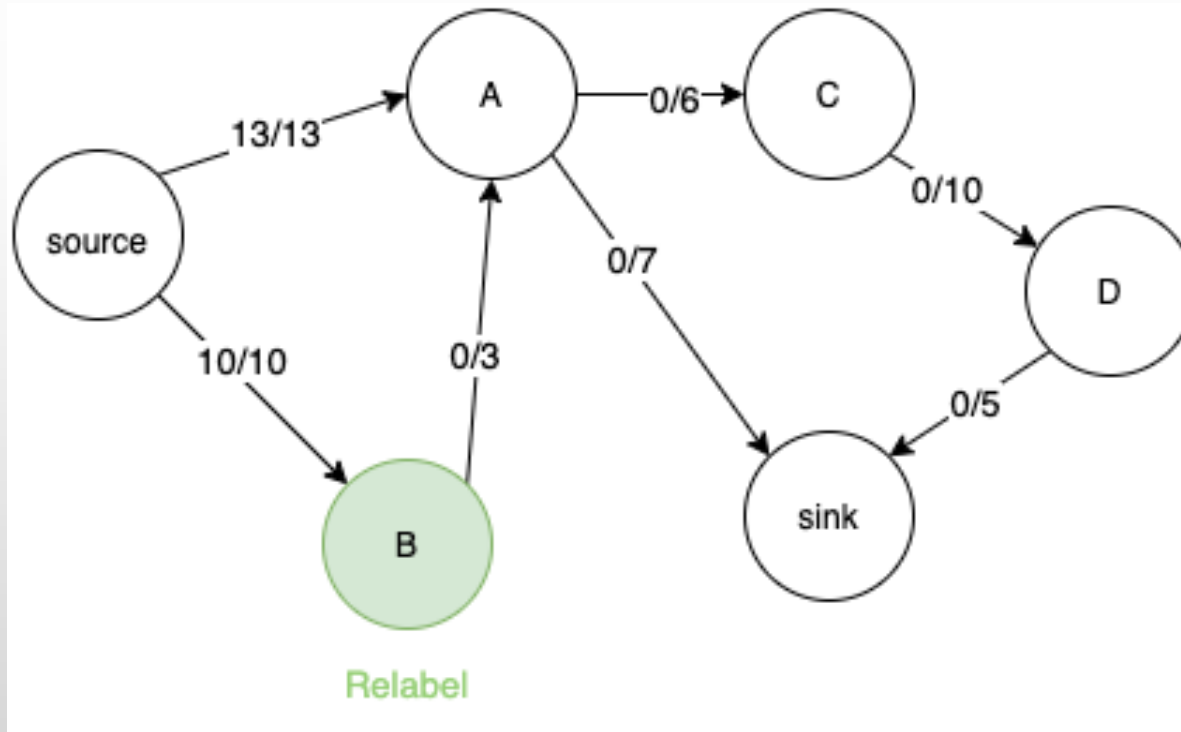


Node	Height	Excess Flow
source	6	
A	0	13
B	0	10
C	0	
D	0	
sink	0	



Push Relabel

- B düğümü ele alınır. A ile aynı yüksekliğe sahip olduğundan artık akışı A ya gönderemez. Bu nedenle relabel yapılır.

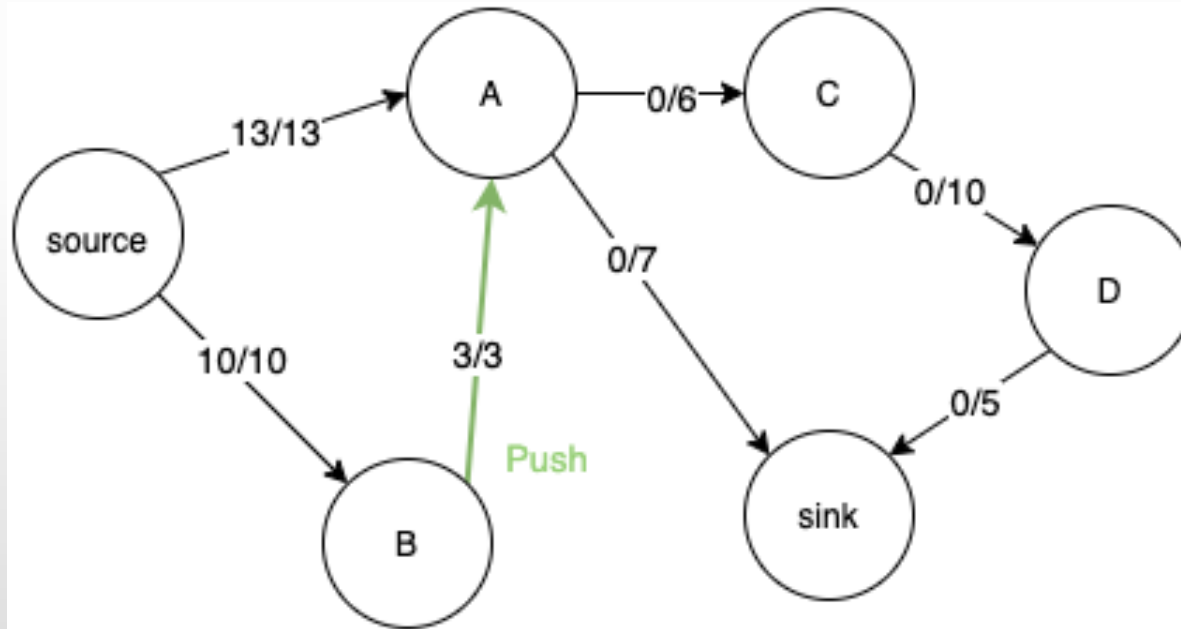


Node	Height	Excess Flow
source	6	-
A	0	13
B	1	10
C	0	
D	0	
sink	0	-



Push Relabel

- B düğümü şimdi artık akışı A'ya gönderebilir.

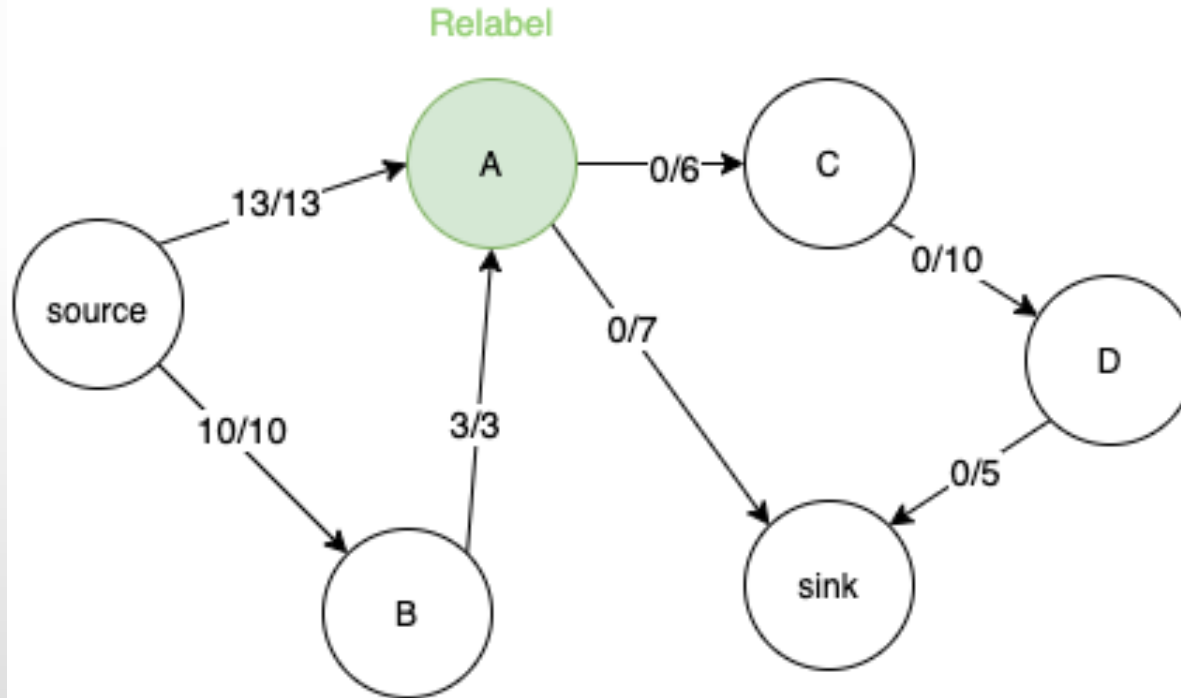


Node	Height	Excess Flow
source	6	-
A	0	16
B	1	7
C	0	
D	0	
sink	0	-



Push Relabel

- A düğümü ele alınır. relabel yapılarak yüksekliği 1 atanır.

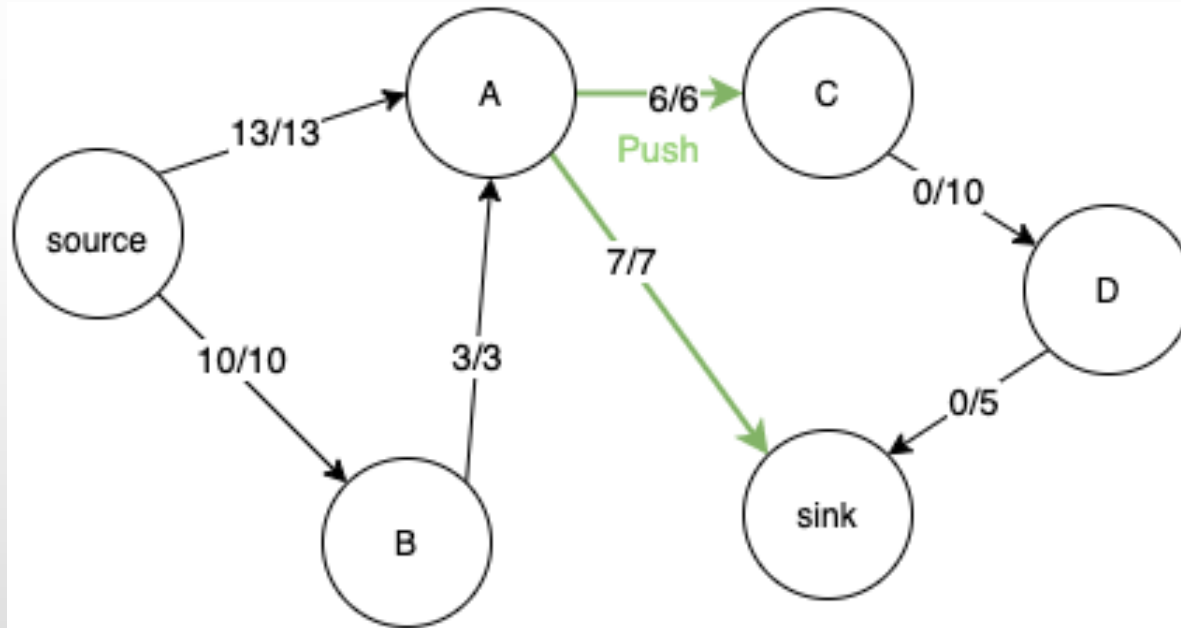


Node	Height	Excess Flow
source	6	-
A	1	16
B	1	7
C	0	
D	0	
sink	0	-



Push Relabel

- A düğümü şimdi C ve sink'e akış yapabilir.

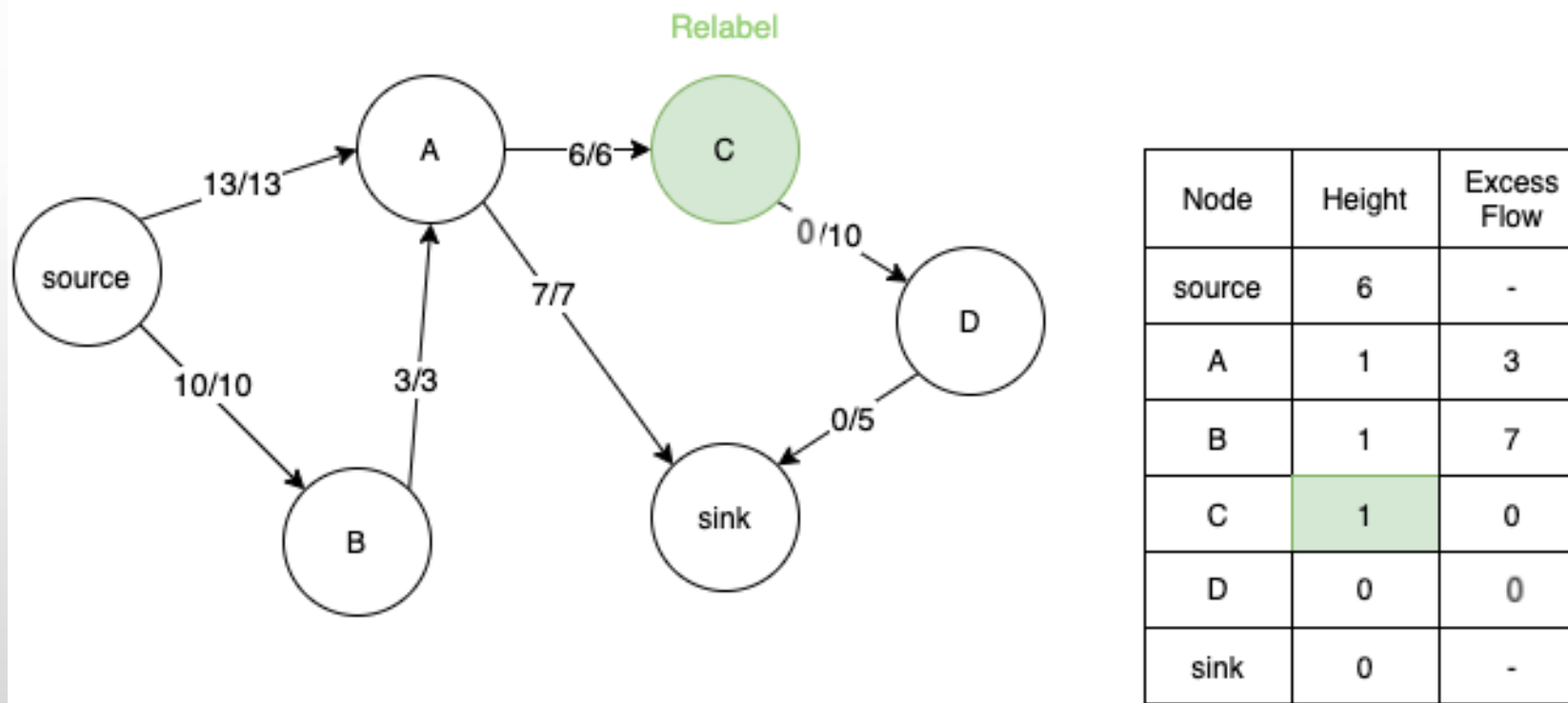


Node	Height	Excess Flow
source	6	-
A	1	3
B	1	7
C	0	6
D	0	
sink	0	-



Push Relabel

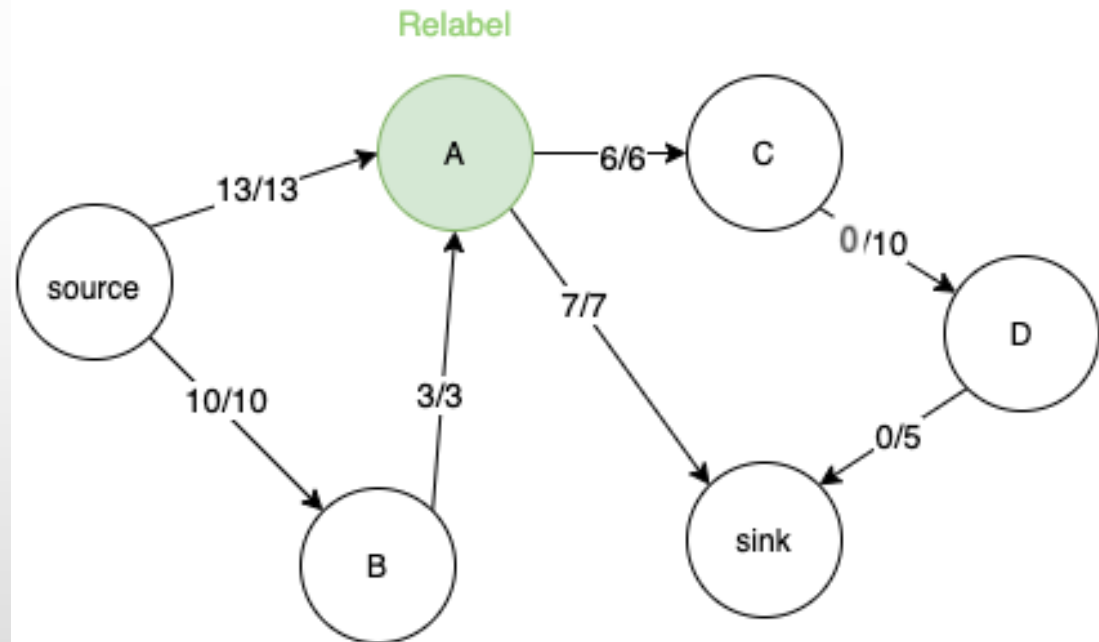
- C düğümü relabel yapılır. A ile aynı yüksekliğe sahip olduğundan A'dan C'ye akış olmaz.





Push Relabel

- A düğümü relabel yapılır.

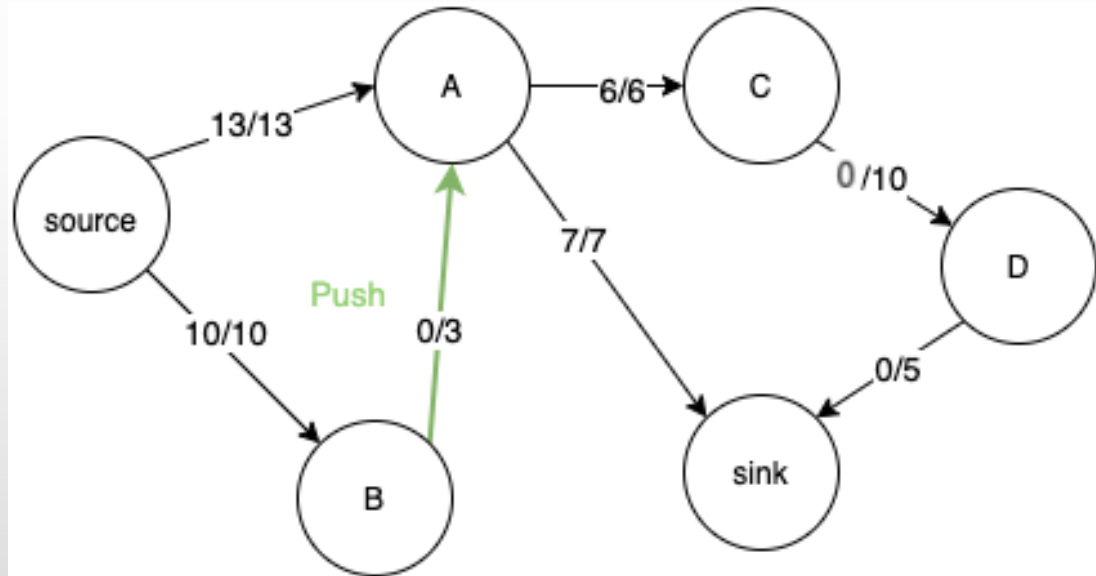


Node	Height	Excess Flow
source	6	-
A	2	3
B	1	7
C	1	0
D	0	0
sink	0	-



Push Relabel

- A'nın yüksekliği B'den fazla olduğu için artık akışı B'ye gönderebilir.

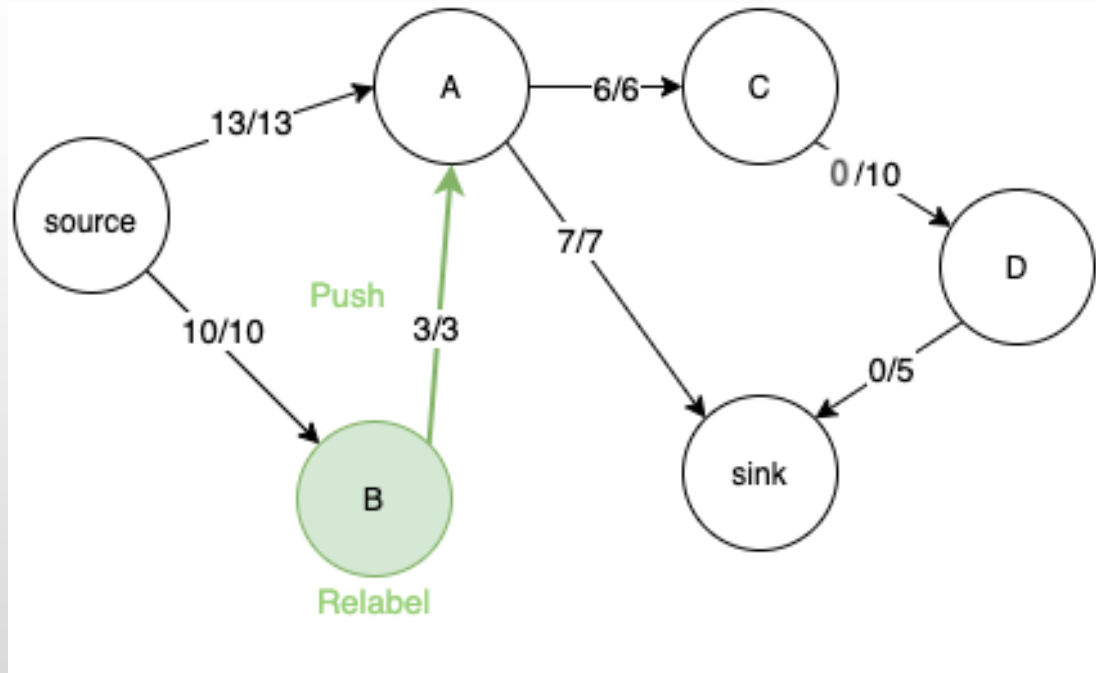


Node	Height	Excess Flow
source	6	-
A	2	0
B	1	10
C	1	0
D	0	0
sink	0	-



Push Relabel

- B'nin akışı göndereceği başka kenar olmadığı için relabel yapılır.

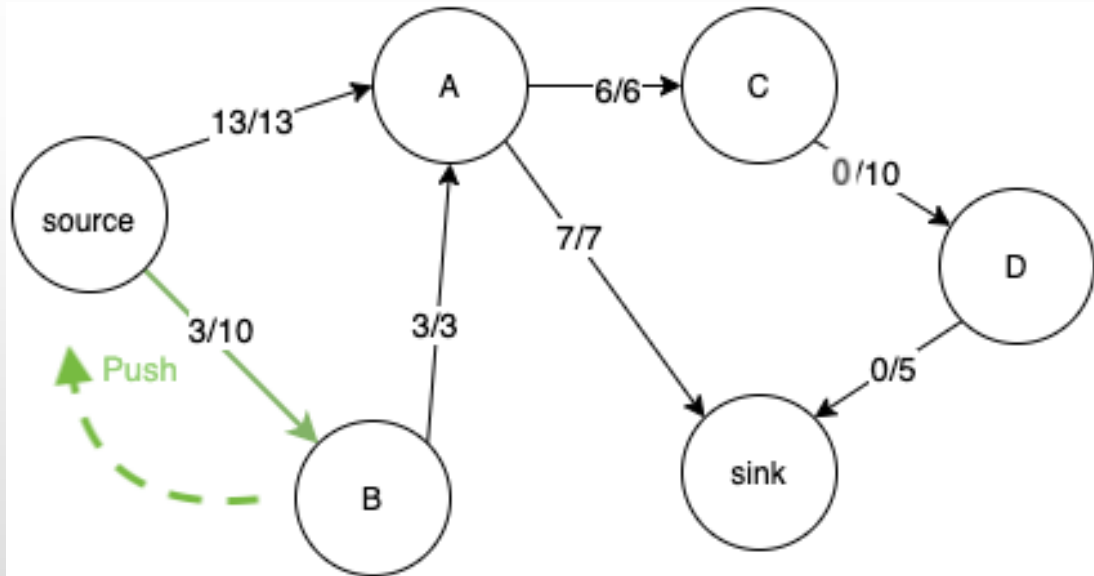


Node	Height	Excess Flow
source	6	-
A	2	3
B	3	7
C	1	0
D	0	0
sink	0	-



Push Relabel

- Bu işlem B'nin yüksekliği source'tan büyük olana kadar devam eder. B şimdi artık akışı kaynak düğüme gönderir.

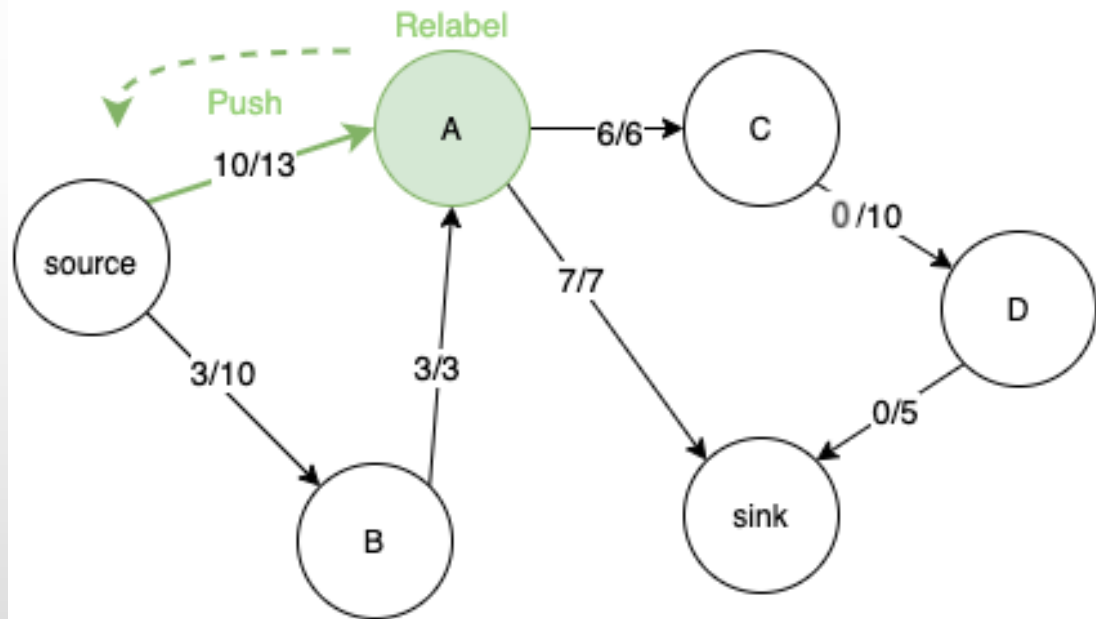


Node	Height	Excess Flow
source	6	-
A	6	3
B	7	0
C	1	0
D	0	0
sink	0	-



Push Relabel

- Aynı şekilde A'nın yüksekliği kaynak düğümden fazla olduğu için artık akışı kaynak düğüme gönderir. Now both A and B nodes have 0 extra flow

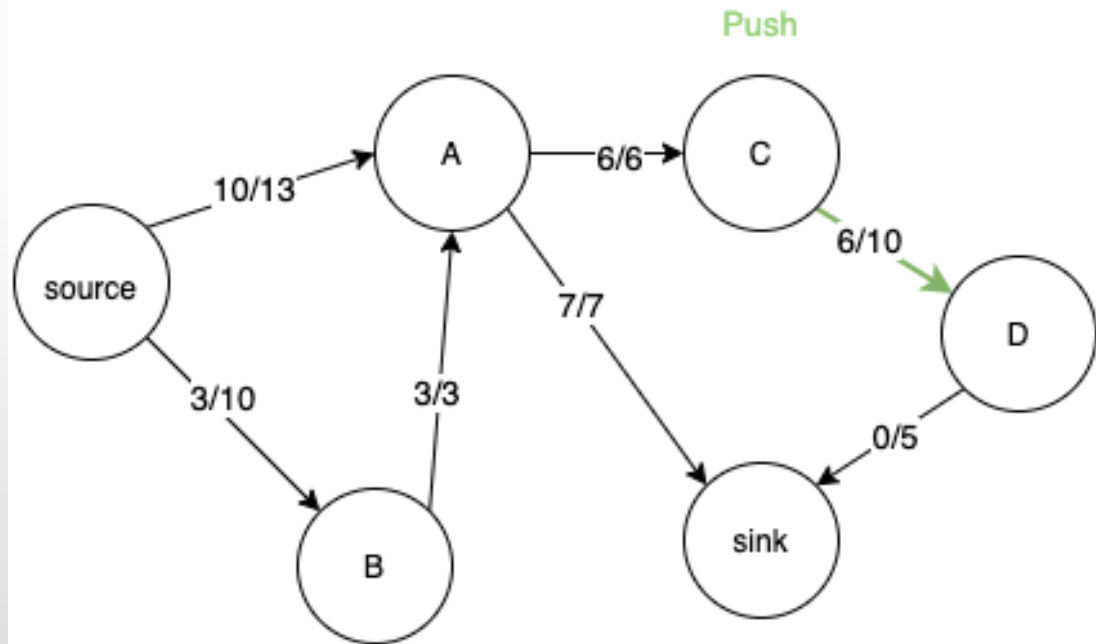


Node	Height	Excess Flow
source	6	-
A	8	0
B	7	0
C	1	0
D	0	0
sink	0	-



Push Relabel

- C ele alınır. Artık akış D'ye gönderilir.

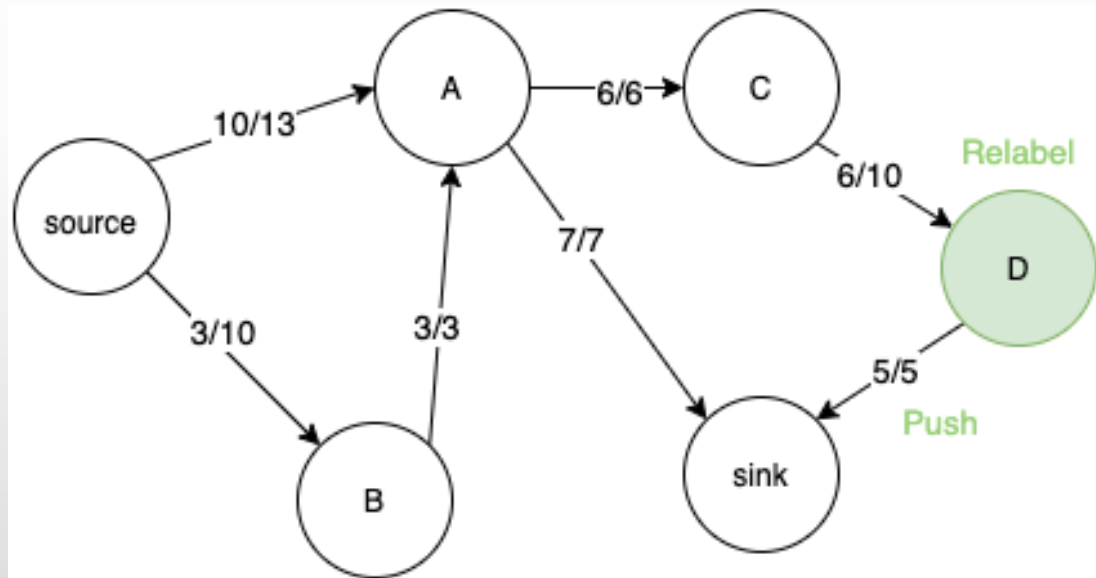


Node	Height	Excess Flow
source	6	-
A	8	0
B	7	0
C	1	0
D	0	6
sink	0	-



Push Relabel

- D relabel yapılır. Artık akış sink'e gönderilir. Hala 1 birim artık akış vardır.

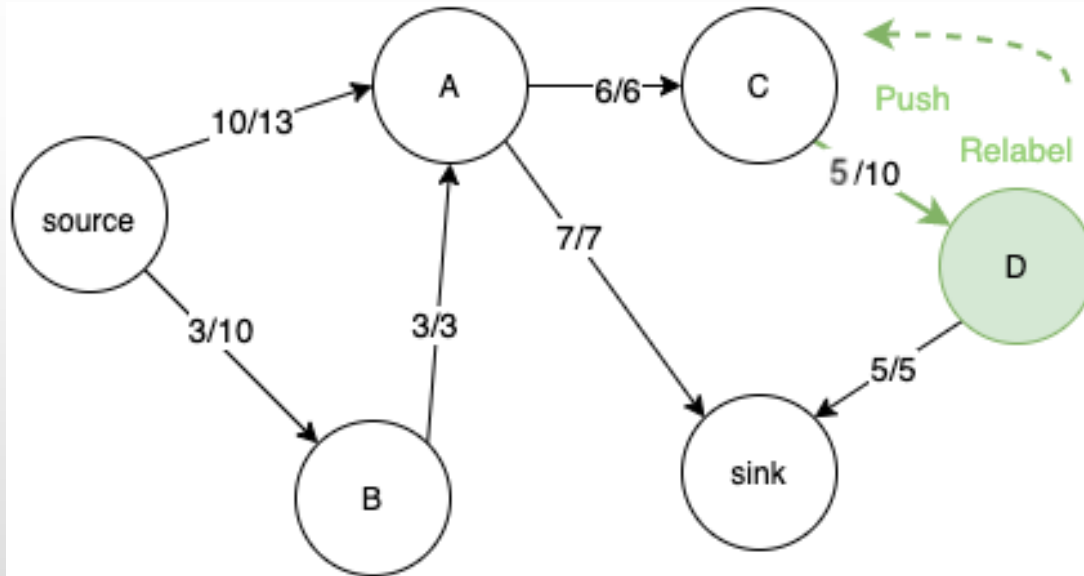


Node	Height	Excess Flow
source	6	-
A	8	0
B	7	0
C	1	0
D	1	1
sink	0	-



Push Relabel

- D relabel yapılır. Artık akış C'ye gönderilir.

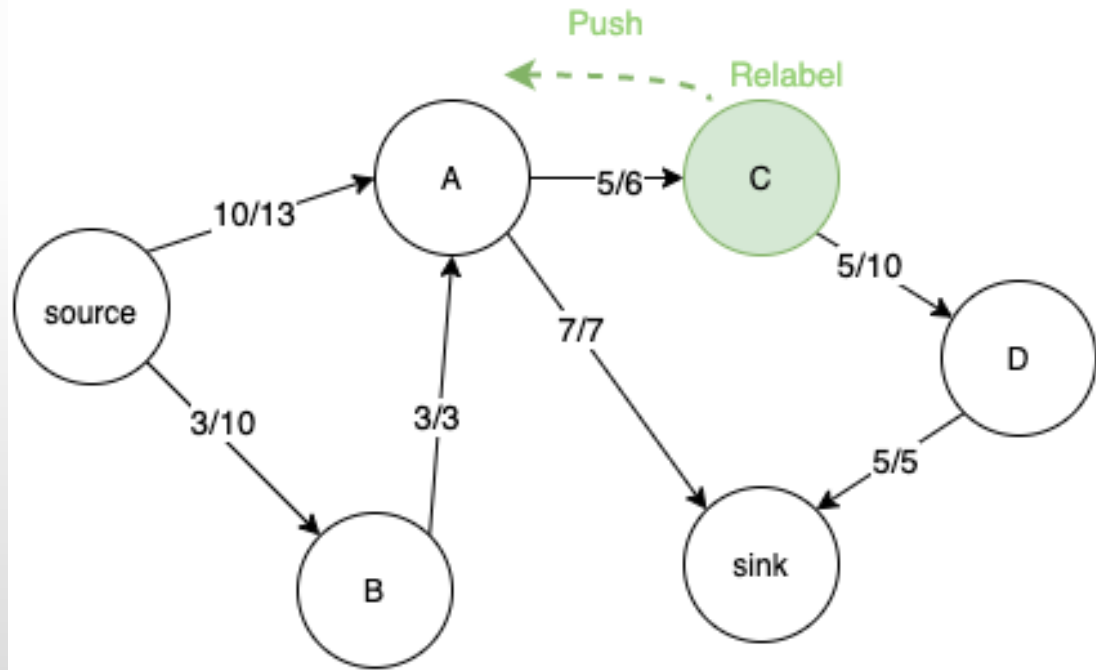


Node	Height	Excess Flow
source	6	-
A	8	0
B	7	0
C	1	1
D	2	0
sink	0	-



Push Relabel

- C'nin yüksekliği A'yı geçene kadar C-D arasında relabel push işlemleri devam eder. C 1 birim ekstra akışı A'ya gönderir.

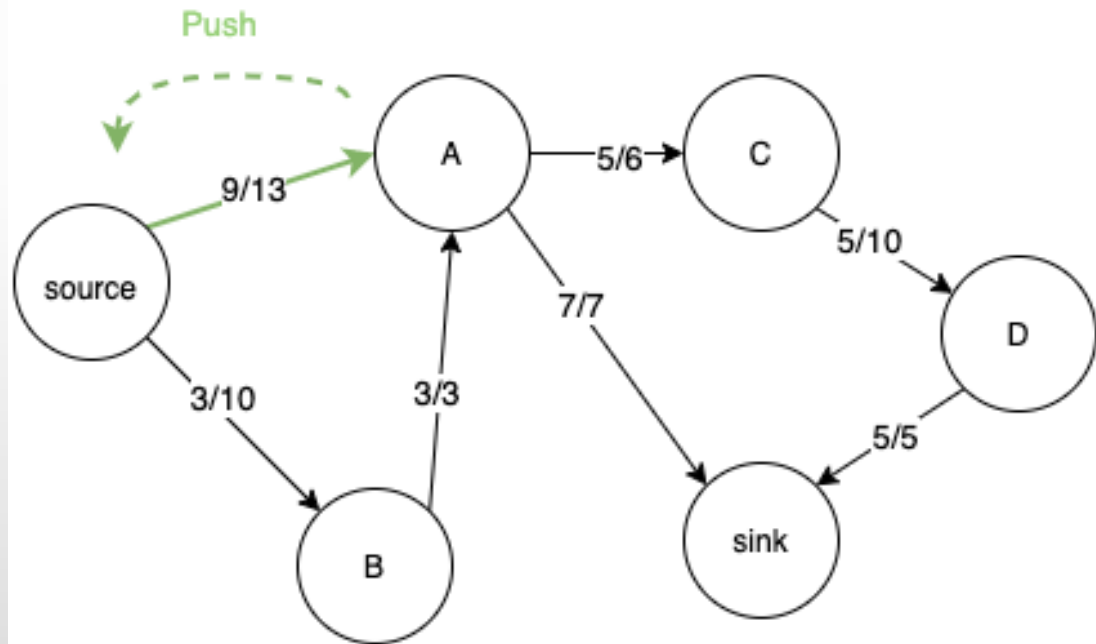


Node	Height	Excess Flow
source	6	-
A	8	1
B	7	0
C	9	0
D	8	0
sink	0	-



Push Relabel

- A 1 birim artık akışı kaynak düğüme gönderir.

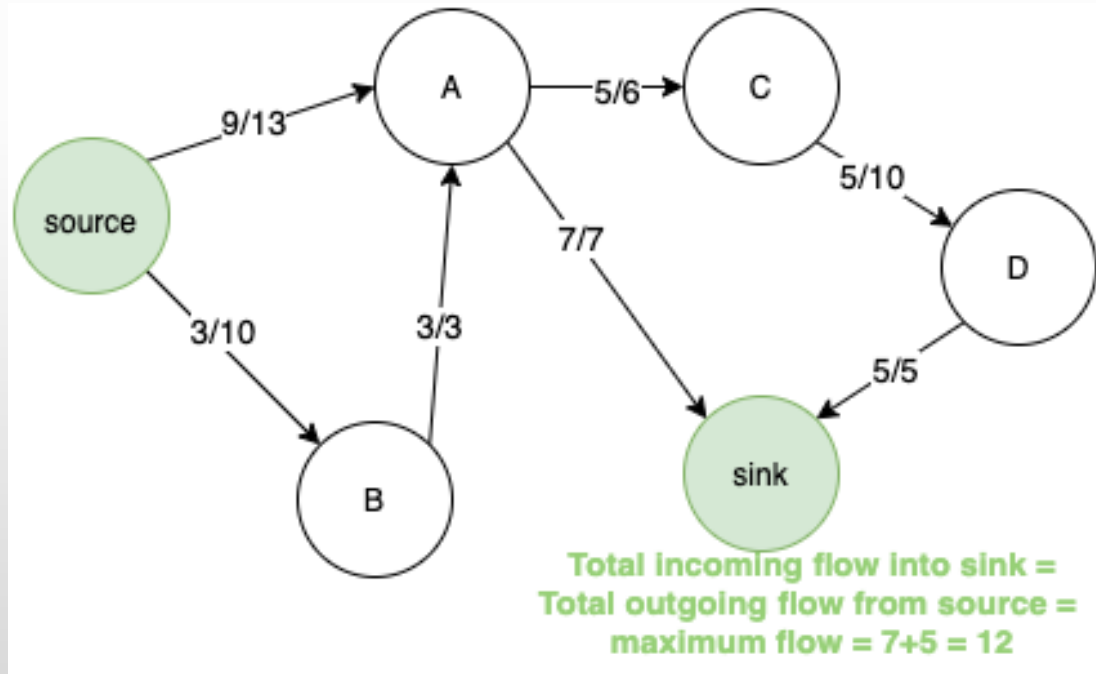


Node	Height	Excess Flow
source	6	-
A	8	0
B	7	0
C	9	0
D	8	0
sink	0	-



Push Relabel

- Maksimum akış kaynak düğümünden çıkan ya da hedef düğüme giren trafik miktarına eşit olur.



Node	Height	Excess Flow
source	6	-
A	8	0
B	7	0
C	9	0
D	8	0
sink	0	-



SON