ÇİZGE ALGORİTMALARI

Ezgi Karabulut Türkseven

Sabancı Üniversitesi

KONU BAŞLIKLARI

01

ARAMA ALGORİTMALARI

02

EN KISA YOL PROBLEMİ

03

MAKSİMUM AKIŞ PROBLEMİ

04

MİNİMUM MALİYETLİ AKIŞ PROBLEMİ

ARAMA ALGORİTMALARI

MOTİVASYON

Bize verilenler:

- ***G = (N , A)*** bir çizge

Amaç:

G'de istenilen yapıyı aramak

Örnek sorular / amaçlar:

- s düğümünden hangi düğümlere erişebiliriz?
- t düğümüne hangi düğümlerden erişebiliriz?
- Çizgedeki tüm bağlı bileşenlerin bulunması

ÖRNEK PROBLEM

s düğümünden hangi düğümlere hangi yoldan ulaşabiliriz?

Tüm düğümler *işaretsiz*, yalnızca *s işaretli*

pred(s) = 0 , *LİSTE* = { s }

LİSTE ≠ ø oldukça

LİSTE 'den i düğümünü seç

i'ye komşu *geçerli* (i,j) kenarı varsa

j'yi işaretle , pred(j) = i

LİSTE := *LİSTE* ∪ {j}

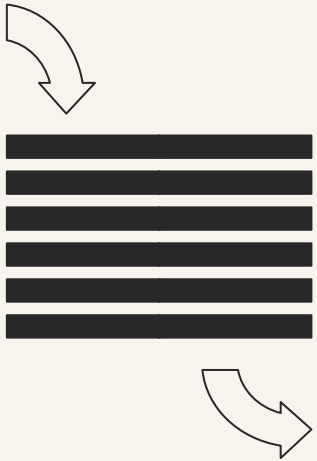
yoksa

LİSTE := *LİSTE* \ {i}

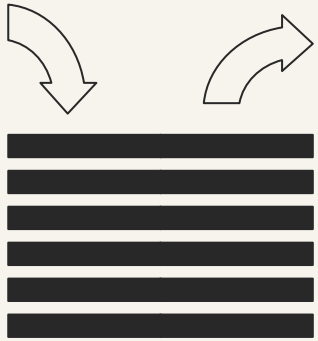
- Algoritma boyunca her düğüm ya *işaretli* ya da *işaretsiz* olacak
- (i,j) kenarı eğer i düğümü işaretli, j düğümü işaretsiz ise *geçerli kenar* olarak adlandırılır

LİSTE YAPISININ ÖNEMİ

FIFO



Kuyruk Veri Tipi

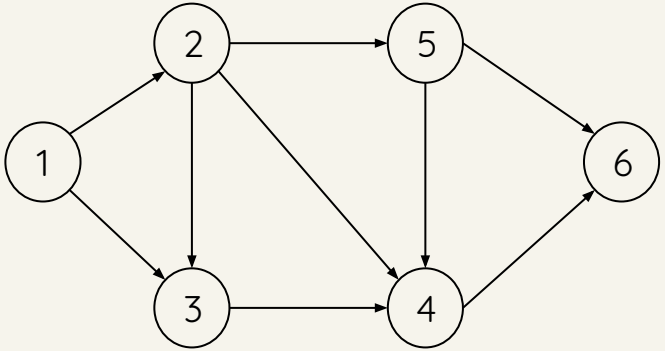


LIFO

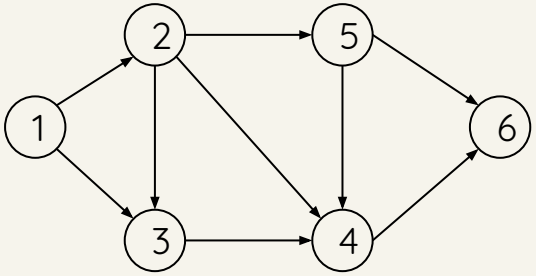
Yığın Veri Tipi

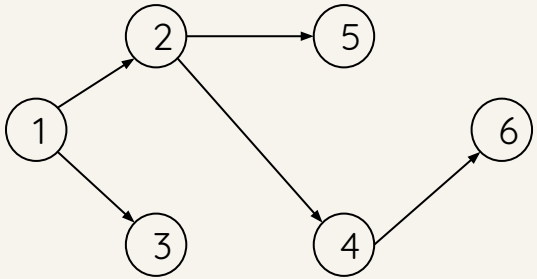
SIĞ ÖNCELİKLİ ARAMA (BFS)

Kuyruk yapısında listeye önce eklenen önce incelenir.



iterasyon	***i***	***Geçerli Kenar***	***İşaretli Düğümler***	***LİSTE***
0				
1				
2				
3				
4				

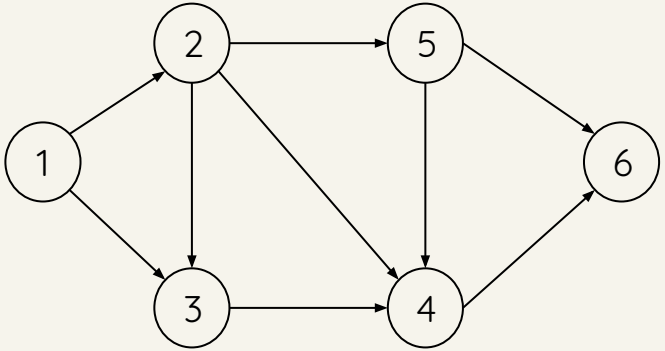




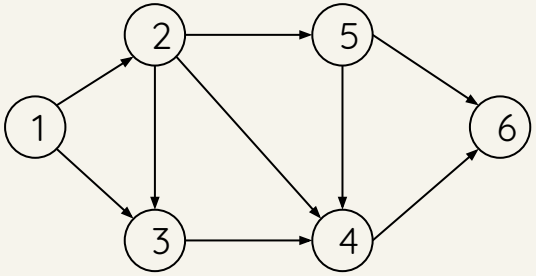
iterasyon	***i***	***Geçerli Kenar***	***İşaretli Düğümler***	***LİSTE***
0	-	-	1	{1}
1	1	(1,2)	1, 2	{1, 2}
2	1	(1,3)	1, 2, 3	{1, 2, 3}
3	1	-	1, 2, 3	{2, 3}
4	2	(2, 4)	1, 2, 3, 4	{2, 3, 4}
5	2	(2, 5)	1, 2, 3, 4, 5	{2, 3, 4, 5}
6	2	-	1, 2, 3, 4, 5	{3, 4, 5}
7	3	-	1, 2, 3, 4, 5	{4, 5}
8	4	(4, 6)	1, 2, 3, 4, 5, 6	{4, 5, 6}
9	4	-	1, 2, 3, 4, 5, 6	{5, 6}
10	5	-	1, 2, 3, 4, 5, 6	{6}
11	6	-	1, 2, 3, 4, 5, 6	{ }

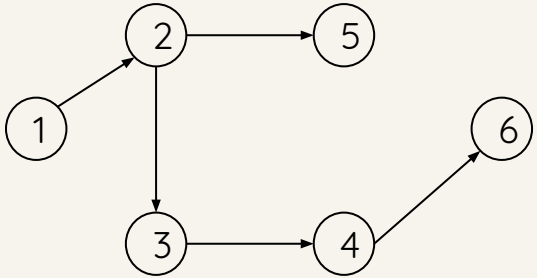
DERİN ÖNCELİKLİ ARAMA (DFS)

Kuyruk yapısında listeye sonra eklenen önce incelenir.



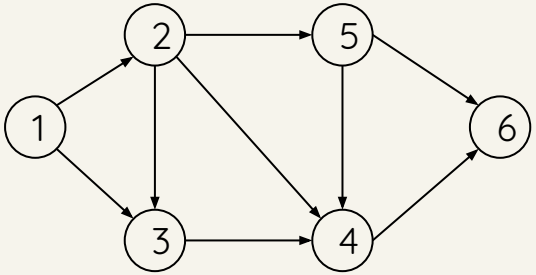
iterasyon	***i***	***Geçerli Kenar***	***İşaretli Düğümler***	***LİSTE***
0				
1				
2				
3				
4				



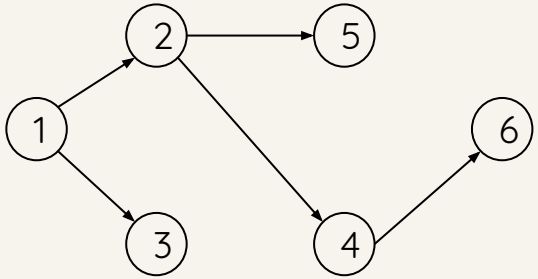


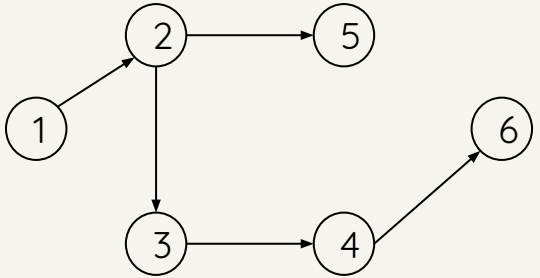
iterasyon	*i*	*Geçerli Kenar*	*İşaretli Düğümler*	*LİSTE*
0	-	-	1	{1}
1	1	(1,2)	1, 2	{1, 2}
2	2	(2,3)	1, 2, 3	{1, 2, 3}
3	3	(3,4)	1, 2, 3, 4	{1, 2, 3, 4}
4	4	(4,6)	1, 2, 3, 4, 6	{1, 2, 3, 4, 6}
5	6	-	1, 2, 3, 4, 6	{1, 2, 3, 4}
6	4	-	1, 2, 3, 4, 6	{1, 2, 3}
7	3	-	1, 2, 3, 4, 6	{1, 2}
8	2	(2,5)	1, 2, 3, 4, 6, 5	{1, 2, 5}
9	5	-	1, 2, 3, 4, 6, 5	{1, 2}
10	2	-	1, 2, 3, 4, 6, 5	{1}
11	1	-	1, 2, 3, 4, 6, 5	{ }

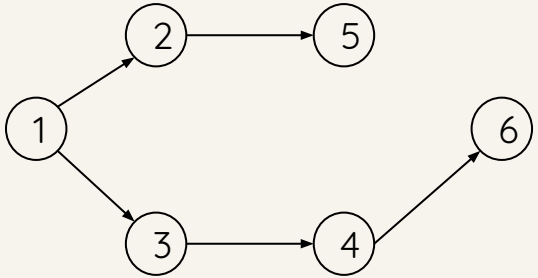
BİRİCİKLİK

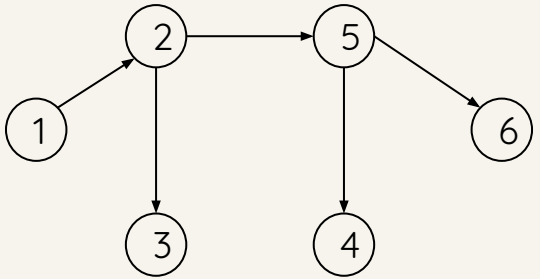


BFS ve DFS arama ağaçları biricik değildir!









EN KISA YOL PROBLEMİ

MOTİVASYON

Bize verilenler:

- ***G = (N , A)*** bir çizge ($|N| = n, |A| = m$)
- $\forall (i,j) \in A$, c_{ij} : kenar uzunlukları
- ***s*** : başlangıç düğümü
- ***t*** : bitiş düğümü

Amaç:

s'den ***t***'ye kenar uzunlukları toplamı en küçük olan yolu bulmak

UYGULAMA ALANLARI

Rotalama Problemleri

Farklı Uygulamalar

(bkz: bir sonraki örnek)

Başka algoritmaların alt programları

ÖR: ÜRETİM HATTINDA DENETLEME İSTASYONLARI

- Seri halinde n aşamadan oluşan bir üretim hattı düşünün.
- Üretim hattına ürünler B sayıda grup halinde geliyor *(batch size)*.
- Bir ürünün i aşamasında defolu hale gelme ihtimali α_i
- Her bir aşamanın sonrasına bir denetleme istasyonu kurabiliriz.
  - Tüm ürünler denetlenir. Defolular atılır, defosuzlar sonraki aşamaya devam eder.
- Maliyetler:
  - p_i : i istasyonunda işlem gören parça başına maliyet
  - f_{ij} : en son denetleme i 'deyken j 'ye denetleme istasyonu kurma maliyeti
  - g_{ij} : en son denetleme i 'deyken j 'de denetleme için parça başına maliyet

ÖR: ÜRETİM HATTINDA DENETLEME İSTASYONLARI

- *(i aşaması sonunda defolu olmayan ürün beklenen sayısı)*

$$B(i) = B \cdot \prod_{k=1}^{i} (1 - \alpha_k)$$

- G=(N,A) : N = {1, … , n} *(aşamalar)* , A = {(i,j) : i < j}

$$c_{ij} = f_{ij} + B(i)\, g_{ij} + B(i) \sum_{k=i+1}^{j} p_k$$

VARSAYIMLAR

Varsayım 1 Kenar uzunlukları tamsayı

Varsayım 2 *s* düğümünden diğer tüm düğümlere yönlü bir yol var

Varsayım 3 Negatif döngü yok

Varsayım 4 Tüm kenarlar yönlü

Etiket Sabitleme

DIJKSTA

Kenar uzunlukları negatif olmamalı

Etiket Düzeltme

BELLMAN - FORD

DIJKSTRA

$S \coloneqq \varnothing$ *(kalıcı etiketli düğümler)* , $\check{S} \coloneqq N$

$d(i) \coloneqq \infty \quad \forall\ i \in N - \{s\}$ ve $d(s) \coloneqq 0$, $pred(s) = 0$

$|S| < n$ oldukça

$i : d(i) = \min \{d(j) : j \in \check{S}\}$ *(etiketi en düşük düğüm)*

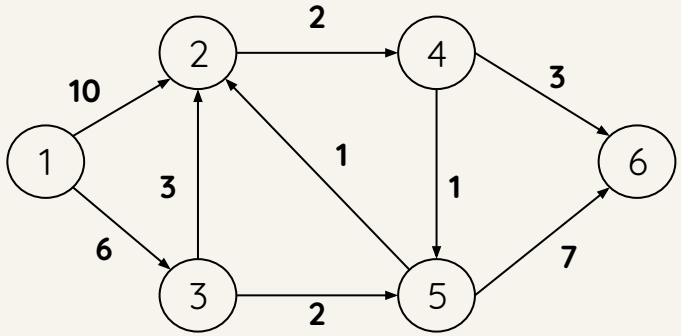
$S \coloneqq S \cup \{i\}$, $\check{S} \coloneqq \check{S} \setminus \{i\}$ *(i'nin etiketini kalıcı yap)*

i'ye komşu tüm $(i,j) \in A$ için

eğer $d(j) > d(i) + c_{ij}$ ise *(etiketleri güncelle)*

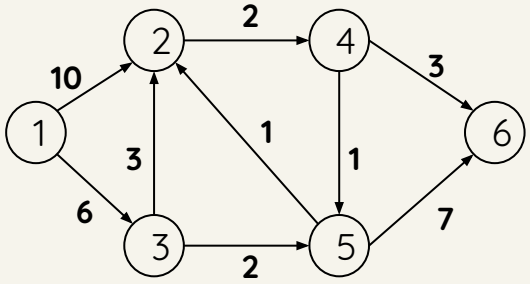
$d(j) \coloneqq d(i) + c_{ij}$, $pred(j) = i$

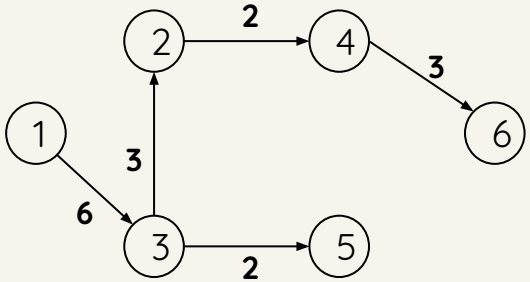
DIJKSTRA - UYGULAMA



	S	*Š*	*d(1)*	*d(2)*	*d(3)*	*d(4)*	*d(5)*	*d(6)*
0								
1								
2								
3								
4								

DIJKSTRA - UYGULAMA





	S	Š	d(1)	d(2)	d(3)	d(4)	d(5)	d(6)
0	{}	{1, 2, 3, 4, 5, 6}	**0**	∞	∞	∞	∞	∞
1	{1}	{2, 3, 4, 5, 6}		10	**6**	∞	∞	∞
2	{1, 3}	{2, 4, 5, 6}		9		∞	**8**	∞
3	{1, 3, 5}	{2, 4, 6}		**9**		∞		15
4	{1, 2, 3, 5}	{4, 6}				**11**		15
5	{1, 2, 3, 4, 5}	{6}						**14**
6	{1, 2, 3, 4, 5, 6}	{}						

DIJKSTRA - KARMAŞIKLIK

$S \coloneqq \varnothing$ *(kalıcı etiketli düğümler)* , $\check{S} \coloneqq N$

$d(i) \coloneqq \infty \quad \forall\ i \in N - \{s\}$ ve $d(s) \coloneqq 0$, $pred(s) = 0$

$|S| < n$ oldukça

- $i : d(i) = \min \{d(j) : j \in \check{S}\}$ *(etiketi en düşük düğüm)*
- $S \coloneqq S \cup \{i\}$, $\check{S} \coloneqq \check{S} \setminus \{i\}$ *(i'nin etiketini kalıcı yap)*
- i'ye komşu tüm $(i,j) \in A$ için
  - eğer $d(j) > d(i) + c_{ij}$ ise *(etiketleri güncelle)*
    - $d(j) \coloneqq d(i) + c_{ij}$, $pred(j) = i$

Düğüm seçimi:

Her turda (n) minimum işlemi (O(n))
⇒ **$O(n^2)$**

Etiket güncelleme:

Her (i,j) kenarı için yalnızca 1 kere (i'nin etiketi sabitlendiğinde) güncelleme işlemi ⇒ **O(m)**

Toplam karmaşıklık ⇒ $O(n^2)$

DIJKSTRA'NIN FARKLI UYGULAMALARI

ALGORİTMA	KARMAŞIKLIK
Orijinal	$O(n^2)$
d-Yığın	$O(m \log_{m/n} n)$
Fibonacci-Yığın	$O(m + n \log n)$
Radix-Yığın	$O(m + n \log(nC))$

BELLMAN - FORD

$d(i) \coloneqq \infty \quad \forall\ i \in N - \{s\}$

$d(s) \coloneqq 0$, $pred(s) = 0$

n - 1 kere:

Tüm $(i,j) \in A$ için

eğer $d(j) > d(i) + c_{ij}$ ise *(etiketleri güncelle)*

$d(j) \coloneqq d(i) + c_{ij}$, $pred(j) = i$

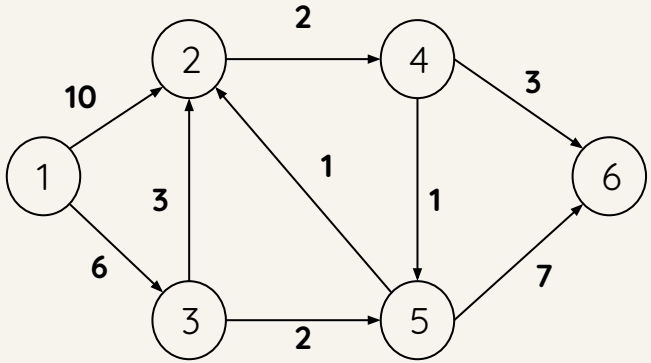
Negatif döngü tespiti: Eğer n. kere tekrar ettiğimizde d(i)ler arasında azalan varsa, negatif döngü vardır.

(*)

d'leri güncelleme kuralına bağlı olarak k. iterasyondaki d(j) değeri :

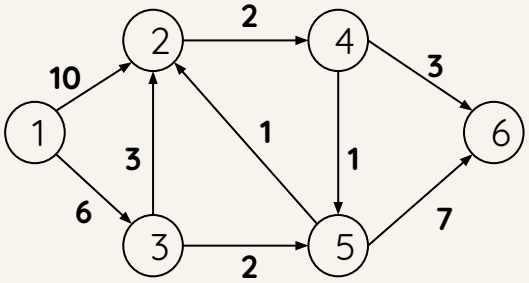
1) = en fazla k kenar kullanarak s'den j'ye giden en kısa yol
2) ≤ en fazla k kenar kullanarak s'den j'ye giden en kısa yol

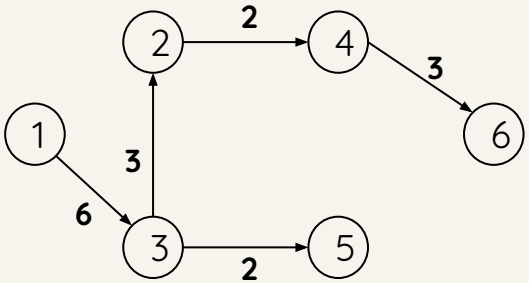
BELLMAN - FORD - UYGULAMA



	d(1)	d(2)	d(3)	d(4)	d(5)	d(6)
0						
1						
2						
3						
4						

BELLMAN - FORD - UYGULAMA





	d(1)	d(2)	d(3)	d(4)	d(5)	d(6)
0	0	∞	∞	∞	∞	∞
1	0	10	6	∞	∞	∞
2	0	9	6	12	8	∞
3	0	9	6	11	8	15
4	0	9	6	11	8	14
5	0	9	6	11	8	14

BELLMAN - FORD - KARMAŞIKLIK

$d(i) \coloneqq \infty \quad \forall\ i \in N - \{s\}$

$d(s) \coloneqq 0$, $pred(s) = 0$

n - 1 kere:

Tüm $(i,j) \in A$ için

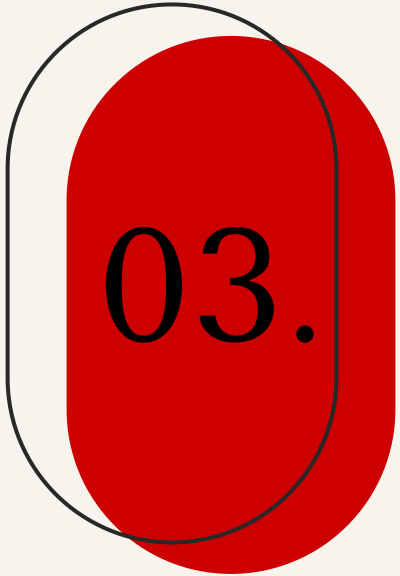
eğer $d(j) > d(i) + c_{ij}$ ise *(etiketleri güncelle)*

$d(j) \coloneqq d(i) + c_{ij}$, $pred(j) = i$

n - 1 tur ⇒ **O(n)**

Her turda m kontrol / O(1) işlem ⇒ **O(m)**

Toplam karmaşıklık ⇒ O(mn)



MAKSİMUM AKIŞ PROBLEMİ

MOTİVASYON

Bize verilenler:

- $G = (N, A)$ bir çizge
- $\forall (i,j) \in A$, u_{ij} : kenar kapasiteleri
- s : başlangıç düğümü
- t : bitiş düğümü

Amaç:

Kenar kapasitelerini aşmayacak şekilde s'den t'ye gönderilebilecek maksimum akışı bulmak

UYGULAMA ALANLARI

“Akış” Problemleri

(ör: boru hattı, yol kapasitesi)

Başka çizge problemlerinin indirgenmesi

(ör: eşleme)

ÖR: PARALEL İŞ ÇİZELGELEME

- Elimizde çizelgelenmesi gereken işler *(J)*, ve bu işleri gerçekleştirebileceğimiz *M* paralel özdeş makine olduğunu düşünün.
- Her iş için bize aşağıdaki üç bilgi verilmiş :
  - p_j : *j* işinin tamamlanması için gereken iş günü
  - r_j : *j* işinin çalışılmaya hazır hale geldiği gün
  - d_j : *j* işinin teslim edilmesi gereken gün
- Bir makinede aynı anda en fazla bir iş işlenebilir.
- Bir iş aynı anda en fazla bir makinede işlenebilir.
- İşler bölünebilir *(ör: 1. gün işlenmeye başlayan iş 2. gün durdurulup 3. gün devam edebilir)*.

ÖR: PARALEL İŞ ÇİZELGELEME

- G = (N, A):
  - N = **J** *(işler)* U **D** *(günler)* U **{s,t}**
  - A = **{(s, j): j ∈ J}** U **{(j , i): j ∈ J , i ∈ D, $r_j \leq i < d_j$}** U **{(i, t): i ∈ D}**

    A_1 A_2 A_3
  - $u_{ij} = p_{ij}$ $(i,j) \in A_1$ / $u_{ij} = 1$ $(i,j) \in A_2$ / $u_{ij} = M$ $(i,j) \in A_3$

VARSAYIMLAR

Varsayım 1 Kenar kapasiteleri negatif olmayan tamsayı

Varsayım 2 s düğümünden t düğümüne kapasitesi ∞ olan yol yok

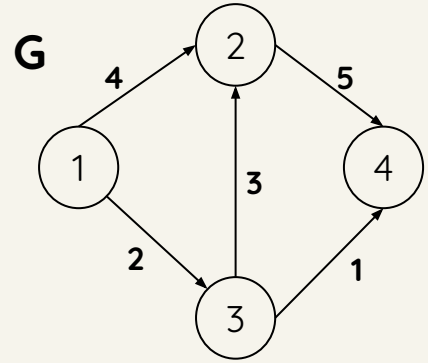
Varsayım 3 $(i,j) \in A \Rightarrow (j,i) \in A$

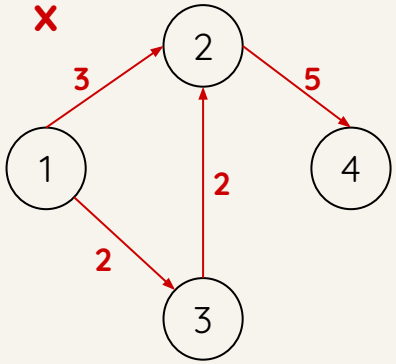
Varsayım 4 Tüm kenarlar yönlü

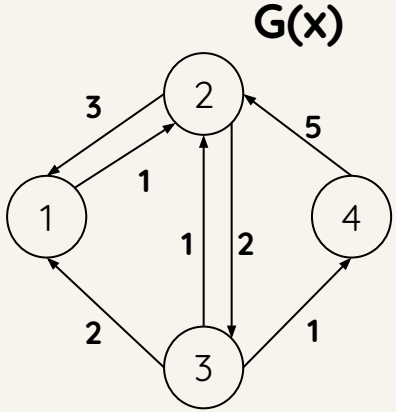
Varsayım 5 Paralel kenar yok

ARTIK AĞ (Residual Network)

- Artık ağ x akışının bir fonksiyonu olarak tanımlanır : G(x)
- x_{ij} = (i,j) kenarından gönderilen akış miktarı
- Artık kapasite (residual capacity)
  - $r_{ij} = u_{ij} - x_{ij} + x_{ji}$







FORD - FULKERSON

$x = 0$, $G(x) = G$, $r = u$

G(x)'te s'den t'ye pozitif kapasiteli bir yol olduğu sürece:

P ≔ s'den t'ye pozitif kapasiteli bir yol

$\delta \coloneqq \min \{ r(i,j) : (i,j) \in P \}$ *(P üzerinde kapasiteleri aşmadan kaç birim akış gönderebiliriz)*

$x(i,j) \coloneqq x(i,j) + \delta \quad \forall (i,j) \in P$ *(akışı arttır)*

$r(i,j) \coloneqq r(i,j) - \delta \quad \forall (i,j) \in P$ *(artık kapasiteleri güncelle)*

$r(j,i) \coloneqq r(j,i) + \delta \quad \forall (i,j) \in P$ *(artık kapasiteleri güncelle)*

FORD - FULKERSON - UYGULAMA

FORD - FULKERSON - UYGULAMA

FORD - FULKERSON - KARMAŞIKLIK

$x = 0$, $G(x) = G$, $r = u$

G(x)'te s'den t'ye pozitif kapasiteli bir yol olduğu sürece:

- P ≔ s'den t'ye pozitif kapasiteli bir yol
- $\delta \coloneqq \min \{ r(i,j) : (i,j) \in P \}$
- $x(i,j) \coloneqq x(i,j) + \delta \quad \forall (i,j) \in P$
- $r(i,j) \coloneqq r(i,j) - \delta \quad \forall (i,j) \in P$
- $r(j,i) \coloneqq r(j,i) + \delta \quad \forall (i,j) \in P$

En kötü senaryoda her adımda δ = 1 birimlik akış arttırırız, maksimum akış kadar adım gerekir ⇒ **O(nU)**

Arama algoritması ⇒ **O(m)**

P üzerinde en fazla n-1 kenar olabilir ⇒ **O(n)**

Toplam karmaşıklık ⇒ O(mnU)

Her adımda 1 birim akış itmekten daha iyisini yapabilir miyiz?

Evet

KAPASİTE ÖLÇEKLEME (Capacity Scaling)

G(x, Δ) : yalnızca r(i,j) ≥ Δ olan kenarlardan oluşan artık ağ.

$x = 0$, $\Delta = 2^{\lfloor \log U \rfloor}$

Δ ≥ 1 olduğu sürece:

G(x, Δ)'da s'den t'ye pozitif kapasiteli bir yol olduğu sürece:

P ≔ G(x, Δ)'da s'den t'ye pozitif kapasiteli bir yol

δ ≔ min { r(i,j) : (i,j) ∈ P }

P üzerinden δ miktarda akış arttır, G(x, Δ)'yı güncelle

Δ ≔ Δ / 2

KAPASİTE ÖLÇEKLEME - KARMAŞIKLIK

$x = 0$, $\Delta = 2^{\lfloor \log U \rfloor}$

$\Delta \geq 1$ olduğu sürece: ⇒ **O(log U)**

G(x, Δ)'da s'den t'ye pozitif kapasiteli bir yol olduğu sürece: ⇒ **O(m)**

P ≔ G(x, Δ)'da s'den t'ye pozitif kapasiteli bir yol

δ ≔ min { r(i,j) : (i,j) ∈ P }

P üzerinden δ miktarda akış arttır, G(x, Δ)'yı güncelle ⇒ **O(m)**

$\Delta \coloneqq \Delta / 2$

G(x, Δ)'da arttırılan her akış en az Δ kadar.
G(x, Δ)'da en fazla 2m kere akış arttırabiliriz.

Toplam karmaşıklık ⇒ $O(m^2 \log(U))$

Her adımda yeniden yol bulup tüm yol boyunca akış göndermekten daha iyisini yapabilir miyiz?

Evet

MOTİVASYON

TANIMLAR

Akış : (i'ye gelen) - (i'den giden) = 0 $\quad i \in N - \{s, t\}$

Önakış : (i'ye gelen) - (i'den giden) ≥ 0 $\quad i \in N - \{s, t\}$

Fazlalık : $e(i) := $ (i'ye gelen) - (i'den giden)

Aktif düğüm : fazlalığı olan düğümler ⇔ $i : e(i) > 0$

TANIMLAR

d(i) : i düğümünün t düğümünden "uzaklığı" *(Algoritma boyunca akış ittirdikçe güncellenecek)*

Geçerli kenar : üzerinden akış ittirebildiğimiz kenarlar ⇔ (i,j) : d(j)=d(i) + 1

(i,j) kenarı üzerinden ittiğimiz akış miktarı = **δ**

- δ = r(i,j) ise **doyurucu itiş**
- δ < r(i,j) ise **doyurucu olmayan itiş**

ÖNAKIŞ - İTİŞ (Preflow-Push)

x ≔ 0

d(i)leri hesapla *(i düğümünün t düğümünden uzaklığı)*

Bütün (s,j) ∈ A için x(s,j) = u(s,j)

d(s) ≔ n

Aktif düğüm olduğu sürece:

Aktif düğümler arasından i düğümünü seç

Eğer geçerli bir (i,j) kenarı varsa

i'den j'ye **δ** ≔ min {e(i), r(i,j)} miktarda akış ittir

Yoksa

d(i) ≔ min{ d(j) + 1 : (i,j) ∈ A ve r(i,j) > 0}

ÖNAKIŞ - İTİŞ - UYGULAMA

ÖNAKIŞ - İTİŞ - UYGULAMA

ÖNAKIŞ - İTİŞ - KARMAŞIKLIK

x ≔ 0

d(i)leri hesapla

Bütün (s,j) ∈ A için x(s,j) = u(s,j)

d(s) ≔ n

Aktif düğüm olduğu sürece:

    Aktif düğümler arasından i düğümünü seç

    Eğer geçerli bir (i,j) kenarı varsa

        i'den j'ye **δ** ≔ min {e(i), r(i,j)} miktarda akış ittir

    Yoksa

        d(i) ≔ min{ d(j) + 1 : (i,j) ∈ A ve r(i,j) > 0}

Yeniden etiketleme ⇒ **$O(n^2)$**

Doyurucu itiş ⇒ **$O(mn)$**

Doyurucu olmayan itiş ⇒ **$O(mn^2)$**

Toplam karmaşıklık ⇒ $O(mn^2)$

04.

MİNİMUM MALİYETLİ AKIŞ PROBLEMİ

MOTİVASYON

Bize verilenler:

- $G = (N, A)$ bir çizge
- $\forall (i,j) \in A$, u_{ij} : kenar kapasiteleri
- $\forall (i,j) \in A$, c_{ij} : kenar maliyetleri
- $\forall i \in N$, b_i : düğüm arz / talepleri

Amaç:

Kenar kapasitelerini aşmayacak ve tüm düğümlerin arz / taleplerini karşılayacak maliyeti en ucuz akışı bulmak

ÖR: "DOLMUŞ" UÇAK YOLCU YÜKLEMESİ

- Sırayla $C = \{1, 2, 3, \ldots, n\}$ şehirlerine uçan bir uçak düşünün.
- Boş koltuk olduğu sürece ekstra yolcu almak mümkün.
- *i* şehrinden *j* şehrine uçmak isteyen b_{ij} tane yolcu var.
- *i* şehrinden *j* şehrine uçan yolcunun ödeyeceği ücret f_{ij}
- *i* şehriyle *i+1* şehri arası uçuş için uçakta p_i tane boş koltuk var.

ÖR: "DOLMUŞ" UÇAK YOLCU YÜKLEMESİ

- G = (N, A):
  - N = **C** *(şehirler)* ∪ **F** = {(i,j): i,j ∈ C , i<j } *(uçuşlar)*
  - A = {(i, i+1): i ∈ C} ∪ {((i,j) , i): (i,j) ∈ F} ∪ {((i,j) , j): (i,j) ∈ F}

    A_1 A_2 A_3

  - $c_{ij} = -f_{ij}$ $((i,j) , i) \in A_2$
  - $u_{i,i+1} = p_i$ $(i, i+1) \in A_1$
  - $b((i,j)) = b_{ij}$ $(i,j) \in F$
  - $b(i) = - \Sigma j\ b_{ji}$ $i \in C$

VARSAYIMLAR

Varsayım 1 Tüm parametreler tamsayı

Varsayım 2 Her düğüm çifti arasında yönlü bir yol var

Varsayım 3 Olurlu bir akış var

Varsayım 4 Tüm kenarlar yönlü

DÖNGÜ ELEME (Cycle Cancelling)

Artık Ağ

G: (c_{ij}, u_{ij}) ve verilmiş x_{ij} akışı için artık ağda kapasite ve maliyetler

- $(i,j) \rightarrow (c_{ij}, r_{ij} = u_{ij} - x_{ij})$
- $(j,i) \rightarrow (-c_{ij}, r_{ji} = x_{ij})$

Olurlu bir x akışını, bir döngü (w) ile arttırmak olurluluğu bozmaz

(döngüde her düğüme giren akış ile çıkan akış aynıdır, denge bozulmaz).

Eğer bu döngünün toplam maliyeti negatif ise x + w in maliyeti x'in maliyetinden daha küçüktür ⇒ x optimal olamaz.

Optimallik Koşulu : Artık ağda negatif döngü olamaz.

DÖNGÜ ELEME

x = olurlu bir akış

G(x)'te negatif bir döngü olduğu sürece:

w ≔ G(x)'te negatif bir döngü

δ ≔ min { r(i,j) : (i,j) ∈ w }

w üzerinden δ birim akış gönder

G(x)'i güncelle

DÖNGÜ ELEME - KARMAŞIKLIK

x = olurlu bir akış

G(x)'te negatif bir döngü olduğu sürece:

- w ≔ G(x)'te negatif bir döngü
- δ ≔ min { r(i,j) : (i,j) ∈ w }
- w üzerinden δ birim akış gönder
- G(x)'i güncelle

Her adımda toplam maliyeti en az 1 birim azaltıyoruz.

C = max { |cij| } , U = max { |uij| }

- mCU ≤ Toplam Maliyet ≤ mCU

⇒ **O(mCU)**

Bellman - Ford ⇒ **O(mn)**

⇒ **O(n)**

⇒ **O(m)**

Toplam karmaşıklık ⇒ $O(m^2nCU)$

Adım sayısını azaltabilir miyiz?

Evet

OPTİMİZASYON ÖNBİLGİSİ

$$\min \sum c_{ij}\, x_{ij}$$

$$\text{s.t.} \quad \sum_j x_{ij} - \sum_j x_{ji} = b_i \qquad \forall\, i \in N$$

$$0 \le x_i \le u_{ij} \qquad \forall\, (i,j) \in A$$

$$\max \sum b_i\, \pi_i - \sum u_{ij}\, \alpha_{ij}$$

$$\text{s.t.} \quad \pi_i - \pi_j - \alpha_{ij} \le u_{ij} \qquad \forall\, (i,j) \in A$$

$$0 \le \alpha_{ij} \qquad \forall\, (i,j) \in A$$

Optimallik için gerekenler:

(1) Olurlu bir x

(2) Olurlu bir (π , α)

(3) Tümler gevşeklik

(2) ve (3) birleşince elde ettiğimiz:

π değerleri $\forall\, (i,j) : x_{ij} < u_{ij}$ için

$c^{\pi}_{ij} = c_{ij} - \pi_i + \pi_j \ge 0$ ‘ı sağlamalı

“Optimallik koşulları”

TANIMLAR

Sözde Akış : $e(i) = (i'ye\ gelen) - (i'den\ giden) \neq 0$

- $i : e(i) > 0$: **fazlalık düğümleri** (F)
- $i : e(i) < 0$: **eksiklik düğümleri** (E)
- $i : e(i) = 0$: **denge düğümleri** (D)

ARDIŞIK EN KISA YOL (Successive Shortest Path)

$x = 0$, $\pi = 0$, $e(i) = b(i)$

$F = \{i : e(i) > 0\}$, $E = \{i : e(i) < 0\}$

$F \neq \emptyset$ olduğu sürece:

- $k \in F$, $l \in E$ düğümlerini seç
- $d :=$ G(x) üzerinde c^{π} uzunlukları ile k'den diğer düğümlere en kısa yol uzunlukları
- $\pi = \pi - d$
- P := k'den l'ye en kısa yol
- $\delta := \min \{ e(k) , - e(l) , r(i,j) : (i,j) \in P \}$
- P üzerinden k'dan l'ye δ birim akış gönder
- F, E, c^{π}, G(x)'i güncelle

ARDIŞIK EN KISA YOL - KARMAŞIKLIK

$x = 0$, $\pi = 0$, $e(i) = b(i)$

$F = \{i : e(i) > 0\}$, $E = \{i : e(i) < 0\}$

$F \neq \varnothing$ olduğu sürece:

- $k \in F$, $l \in E$ düğümlerini seç
- $d :=$ c^{π} uzunlukları ile en kısa yol uzunlukları
- $\pi = \pi - d$
- $P :=$ k'den l'ye en kısa yol
- $\delta := \min \{ e(k) , - e(l) , r(i,j) : (i,j) \in P \}$
- P üzerinden k'dan l'ye δ birim akış gönder
- F, E, c^{π}, G(x)'i güncelle

Her adımda toplam toplam akışı en az 1 birim arttırıyoruz.
Toplam Akış ≤ nU
⇒ **O(nU)**

En Kısa Yol Karmaşıklığı
⇒ **O(SP(n,m,C))**
$c^{\pi} \geq$ olduğu için Dijkstra kullanılabilir

Toplam karmaşıklık ⇒ O(nU SP(n,m,C))

Her adımda 1 birim akış itmekten daha iyisini yapabilir miyiz?

Evet

KAPASİTE ÖLÇEKLEME

- Tüm çizge üzerinde değil, tek seferde en az Δ birim akış arttırabileceğimiz bir altçizge üzerinde çalış.
- G(x,Δ) için problemi Ardışık En Kısa Yol algoritması ile çöz.
- Δ = Δ/2

Toplam karmaşıklık ⇒ O(m log(U) SP(n,m,C))

TEŞEKKÜRLER

Sorularınız için:

ezgi.turkseven@sabanciuniv.edu

Daha detaylı bilgi için kaynak:

Ahuja, Ravindra K., Thomas L. Magnanti, and James B. Orlin. "Network flows." (1988).