# 

# Teoriuppgifter till labb 3

## Uppgift 1

**Jämför tidskomplexiteten för Edmonds-Karps algoritm då grafen implementeras som en grannmatris och då den implementeras med grannlistor. (För att satsen f[v,u]:= -f[u,v] ska kunna implementeras effektivt måste grannlisteimplementationen utökas så att varje kant har en pekare till den omvända kanten.)**

**Uttryck tidskomplexiteten i n och m där n är totala antalet hörn och m antalet kanter i den bipartita grafen. Välj sedan den implementation som är snabbast då m=O(n), alltså då grafen är gles.**

**1.1 Hur Edmonds-Karps algoritm fungerar (kort):**

Edmonds-Karps algoritm används för att lösa maximal flödesproblem i en graf. Med algoritmen beräknas den kortaste stigen från startnod till slutnod med BFS (Bredd-först-sökning). Edmonds-Karps algoritm anger tydligt sökordningen för att hitta ökande stigar i en graf. Med ökande stigar menas stigar vars flöde maximerar det maximala flödet i en graf (Ökande stigar optimerar en given lösning till maximal flödesproblem).

* Att hitta kortaste stigen tar: O(|E|)
* Att uppdatera flödet längs stigen tar: O(|V|)

Om den kortaste stigen väljs i Ford-Fulkersons metod hittas det maximala flödet efter högst |V||E| varv i slingan

Total tidskomplexitet: O(|V||E| · |E|) = O(|V||E|²)

Space Complexity: O(V+E)

**1.2 Implementation som en grannlista:**

Denna implementation har en lista med storlek **n** (antalet noder) och varje plats i arrayen är **m + 1** lång (antalet kanter) lista och pekaren. Bredd-först-sökning har en tidskomplexitet på O(|V| + |E| med grannlistor). För att kolla alla kanter i grafen tar då: **O(n+m)**. Vi får slutligen: O(m2 \* n)

**1.3 Implementation som en grannmatris:**

Denna implementation har en grannmatris **n** \* **n** (2D array). För att kolla alla kanter i grafen behöver vi iterera över matrisen med två slingor som tar då: O(n\*n) = **O(n²)**. Vi får slutligen: O(m \* n3)

**1.4 Vilken implementation är snabbast för en gles graf?**

Om m = O(n) (grafen är gles) **föredras implementationen som en grannlista**. Då m är O(n) får vi O(n+n) som ger oss **O(n)**.

## 

## Uppgift 2

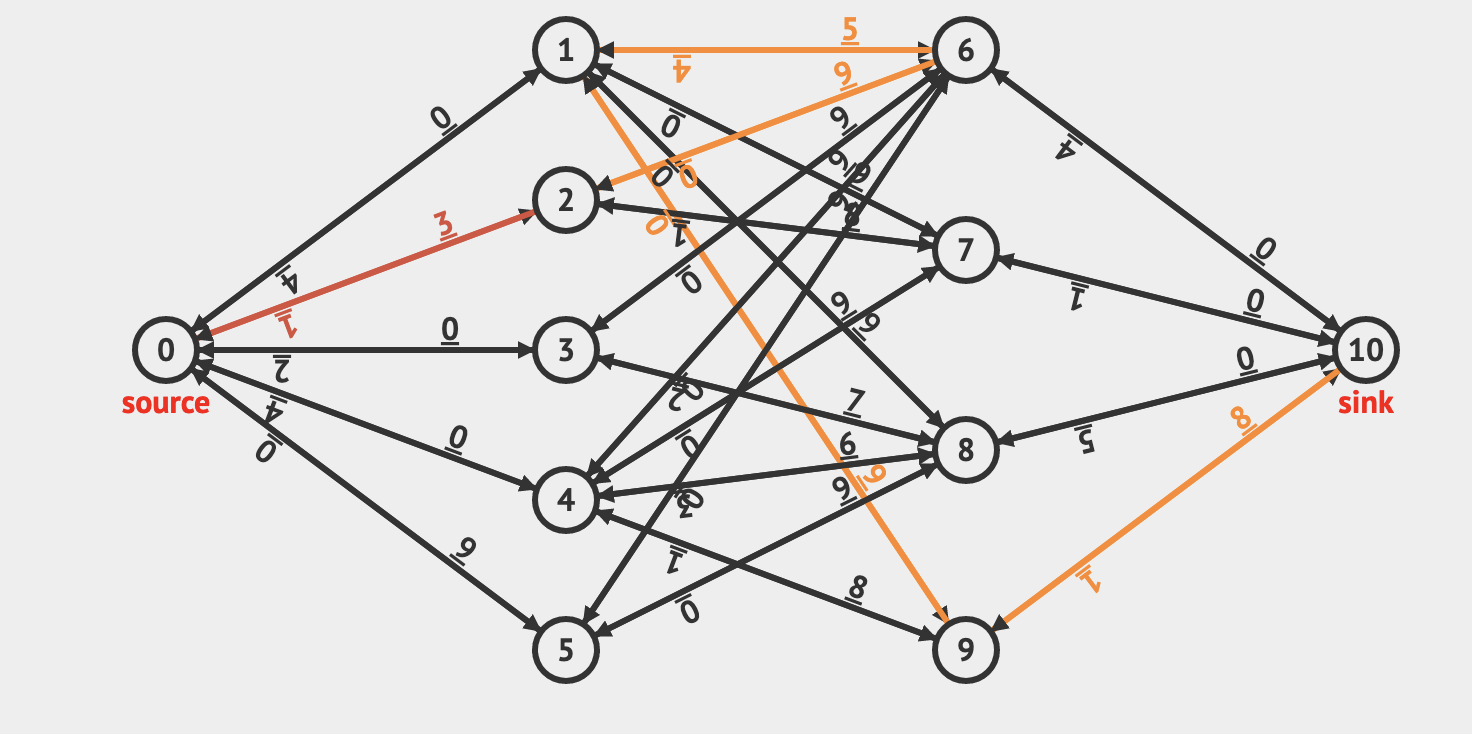
***Kalle menar att om vi börjar med en bipartit graf G och gör om den till en flödesgraf H med ny källa s och nytt utlopp t så kommer avståndet från s till t att vara 3.***

***Kalle tycker därför att BFS-steget alltid kommer att hitta en stig av längd 3 i restflödesgrafen (om det finns någon stig).***

***Det första påståendet är sant, men inte det andra. Varför har stigarna som BFS hittar i restflödesgrafen inte nödvändigtvis längd 3? Hur långa kan de bli?***

När restflödesgrafen skapas från en bipartit graf kan stigarnas riktning ändras till följd av flödet. BFS returnerar inte alltid en stig av längd 3 då en riktning från ett hörn kan leda åt andra hållet och bli en omväg.Den längsta stigen som BFS hittar är när den tar stigen som passerar alla hörn på den sidan (s eller t) som har minst antal kanter.

I värsta fallet så kan BFS stötta på en graph där den ska besöka alla noder i mängden med minst antal noder (de noderna efter start i sketchen) tills den når slutnoden.

Ett exempel med en längd större än 3 visas i figuren:

## 

## Uppgift 3

**Anledningen till att bipartit matchning kan reduceras till flöde är att en lösning till flödesproblemet kan tolkas som en lösning till matchningsproblemet. Detta gäller bara om det flöde som algoritmen ger är ett heltalsflöde (flödet i varje kant är ett heltal), vilket i detta fall innebär att flödet längs en kant antingen är 0 eller 1. Som tur är så är det på det sättet.**

* **Bevisa att Ford-Fulkerson alltid genererar heltalsflöden om kantkapaciteterna är heltal!**

**I föreläsning 13 fick vi lära oss om följande sats:**

*Om c(u,v) ∈ ℕ (heltalskapaciteter) så producerar algoritmen ett maximalt flöde med heltalsflöden i varje kant*

**Vi fick även följande formel i samma föreläsning:**

Kantents kapacitet, restkapaciteten är:  
*(u,v) = c(u,v) − f(u,v), där f(u,v) = −f(v,u)*

I formeln framgår det att om flödet i en kant är ett heltal så måste även *c(u,v)* och *f(u,v)* vara det. Det följer att kantkapaciteten *c(u,v)* är ett heltal.

* **Vad händer med lösningarna som flödesalgoritmen ger om man ändrar i reduktionen så att kantkapaciteterna sätts till 2 istället för 1?**

Det händer inget med lösningarna till flödesalgoritmen om man ändrar i reduktionen så att kantkapaciteterna sätts till 2 istället för 1 (1 representerar true och 0 false). Vi bör kunna använda vilket heltal som helst för att representera true i flödesalgoritmen. Vi måste bara komma ihåg att reduktionen är 2 nu, om vi valde det, istället för 1 (2 ska representera true istället för 1).