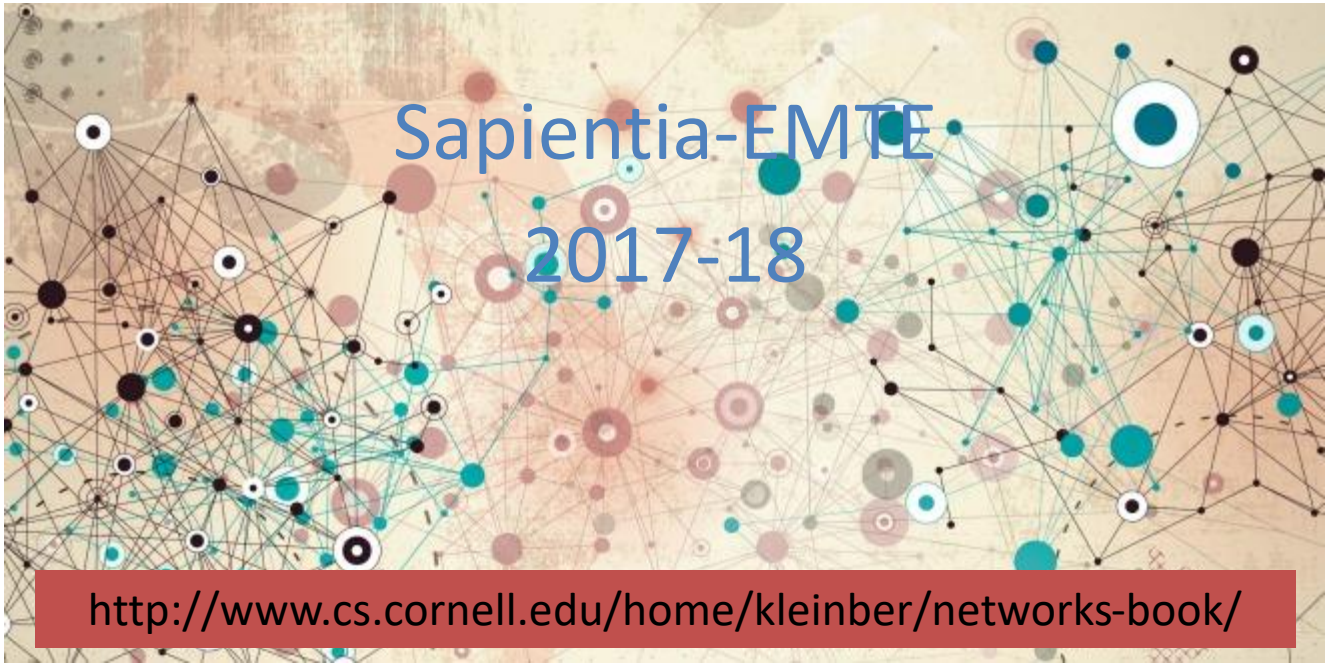
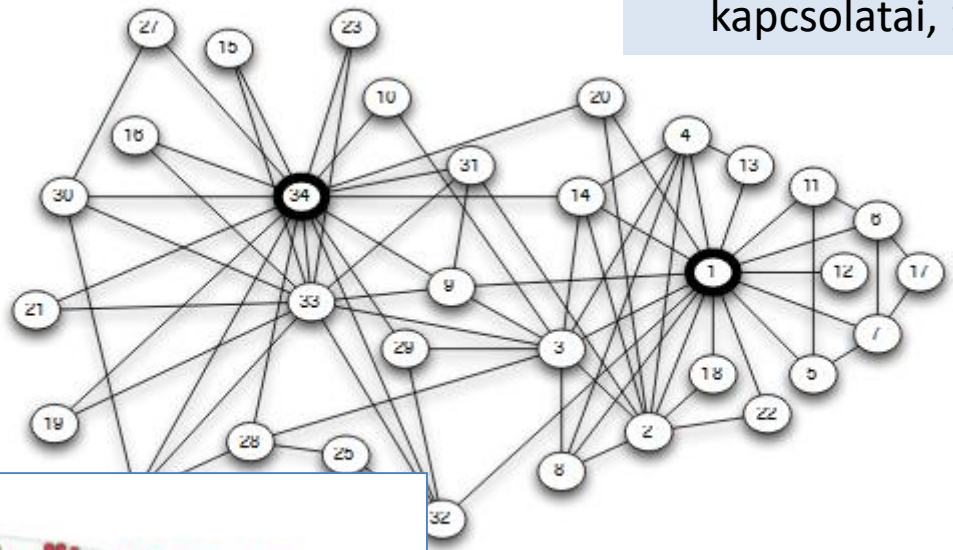
Gráf-algoritmusok
BEVEZETŐ

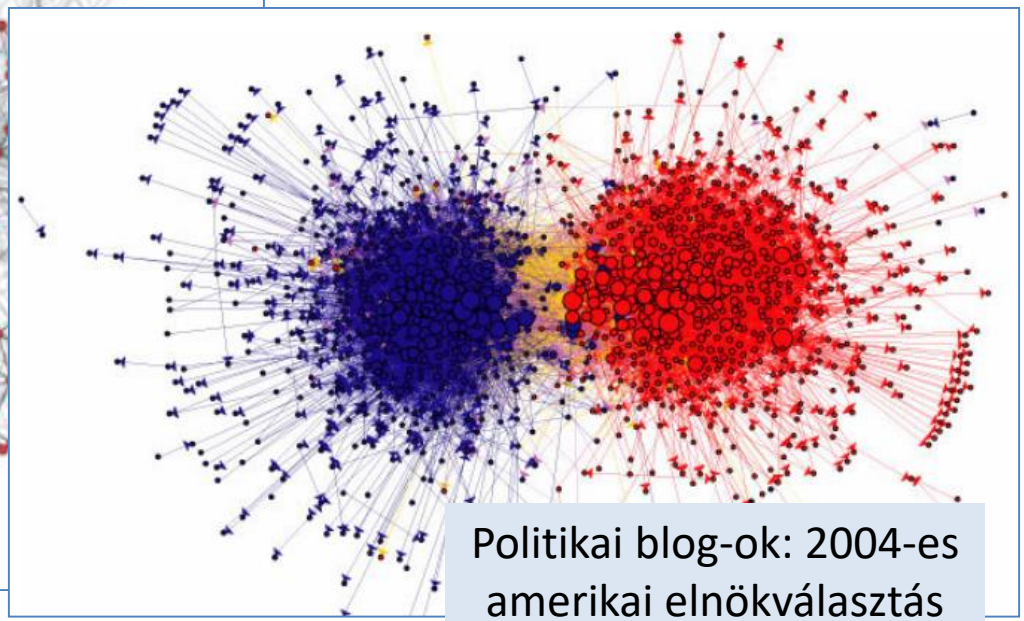


GRÁF - RENESZÁNSZ

34 karate-klubtag baráti kapcsolatai, 1970



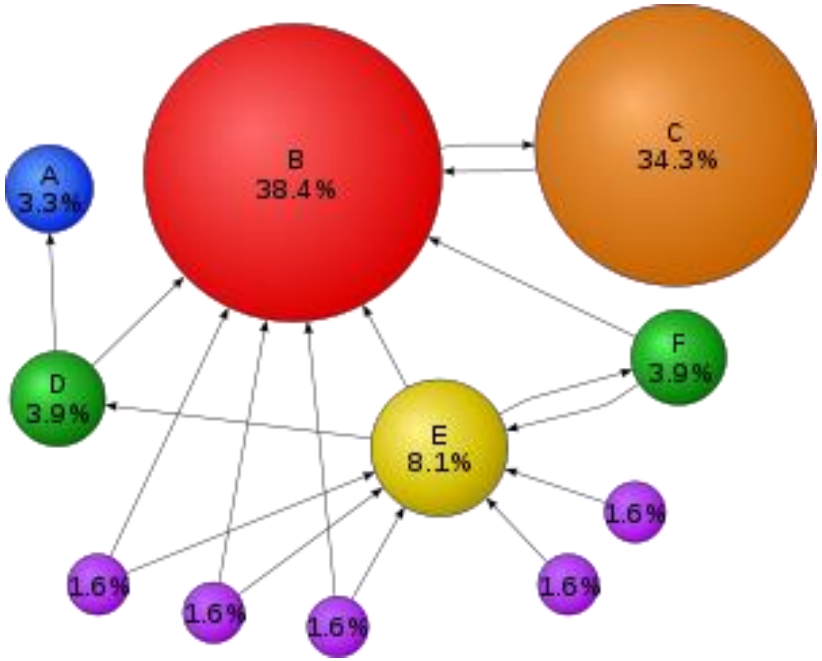
436 HP alkalmazott email kommunikációja



Politikai blog-ok: 2004-es amerikai elnökválasztás

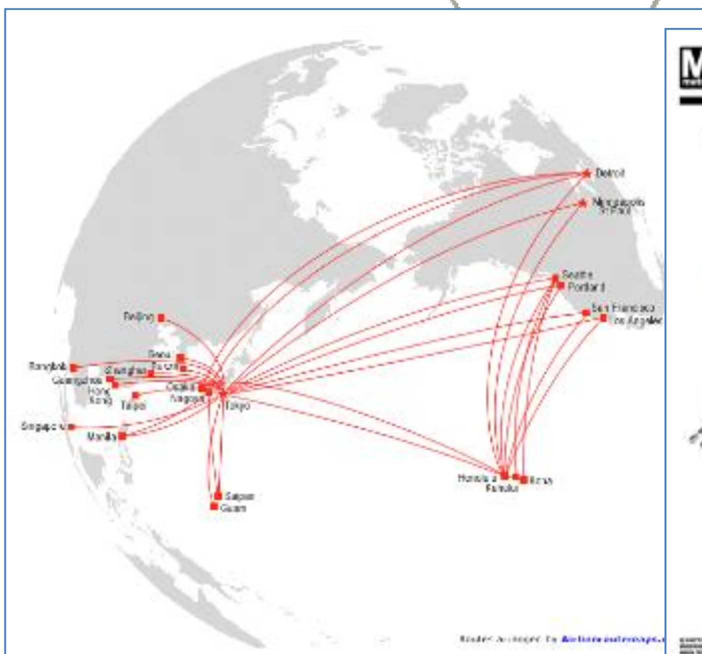
Graph Theory: Key to Understanding Big Data

- „to Google”
  - PageRank algoritmus (gráf-megközelítés)
    - 1998: Larry Page, Sergey Brin
    - az a fontos oldal, amire fontos oldalak mutatnak

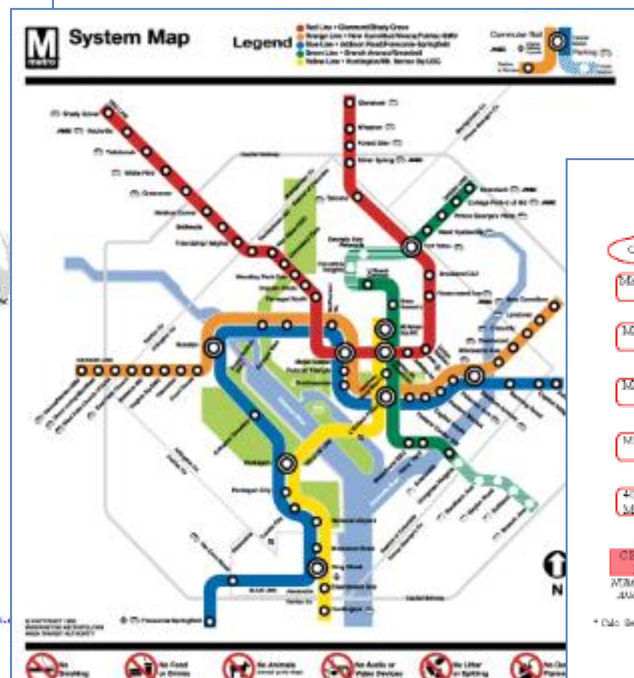


Néhány alapfogalom
pontok, élek

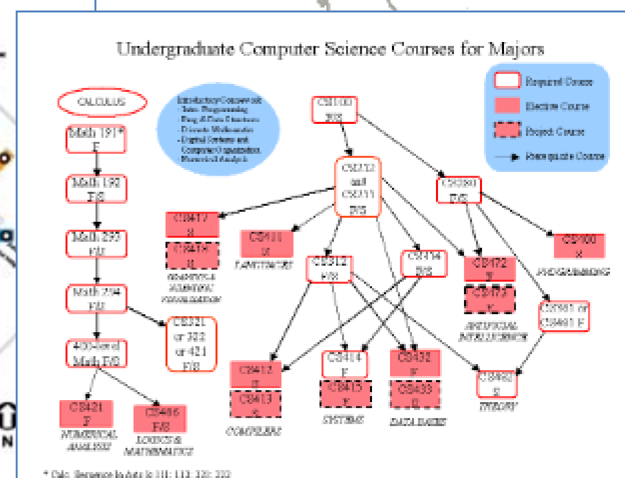
Első internet-háló (Arpanet), 1970



(a) *Airline routes*

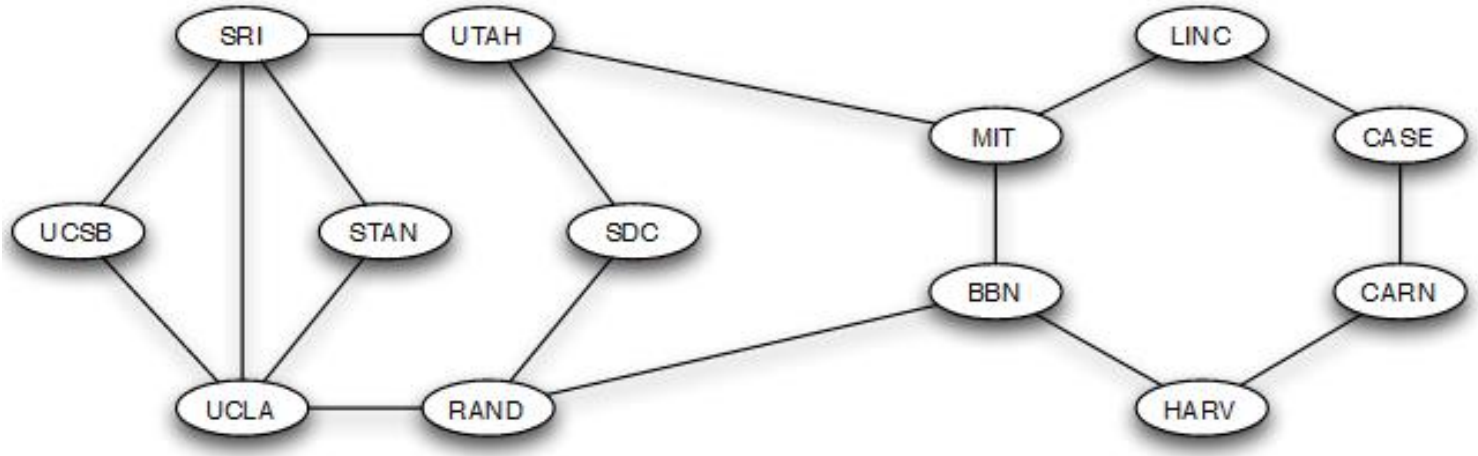


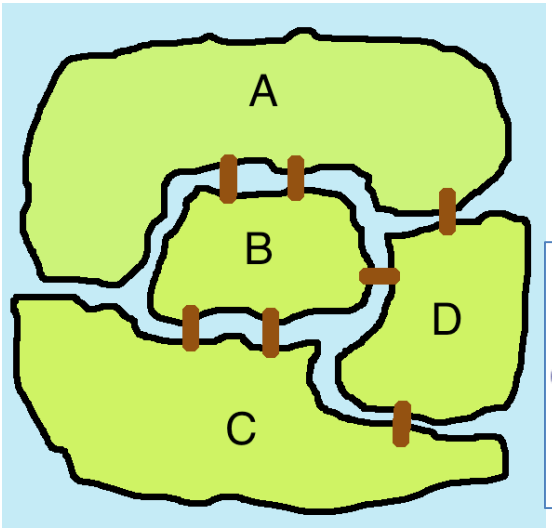
(b) *Subway map*



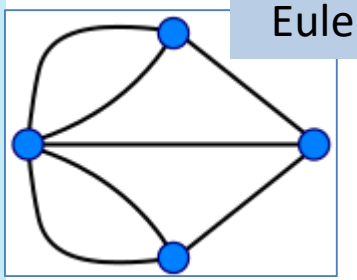
(c) *Flowchart of college courses*

Néhány alapfogalom
út (vonal, séta), kör (zárt vonal)

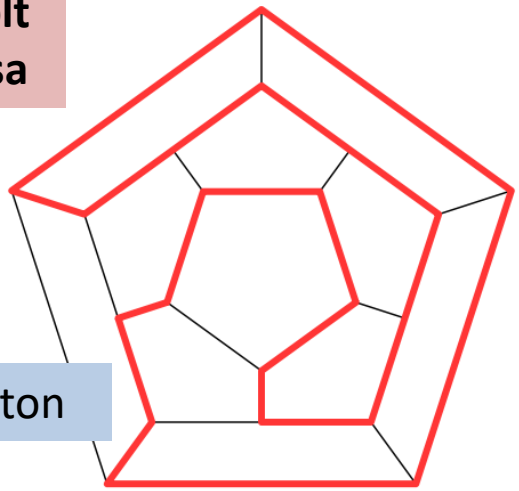




A felségem unokatestvérének a volt iskolatársa a testvérem munkatársa

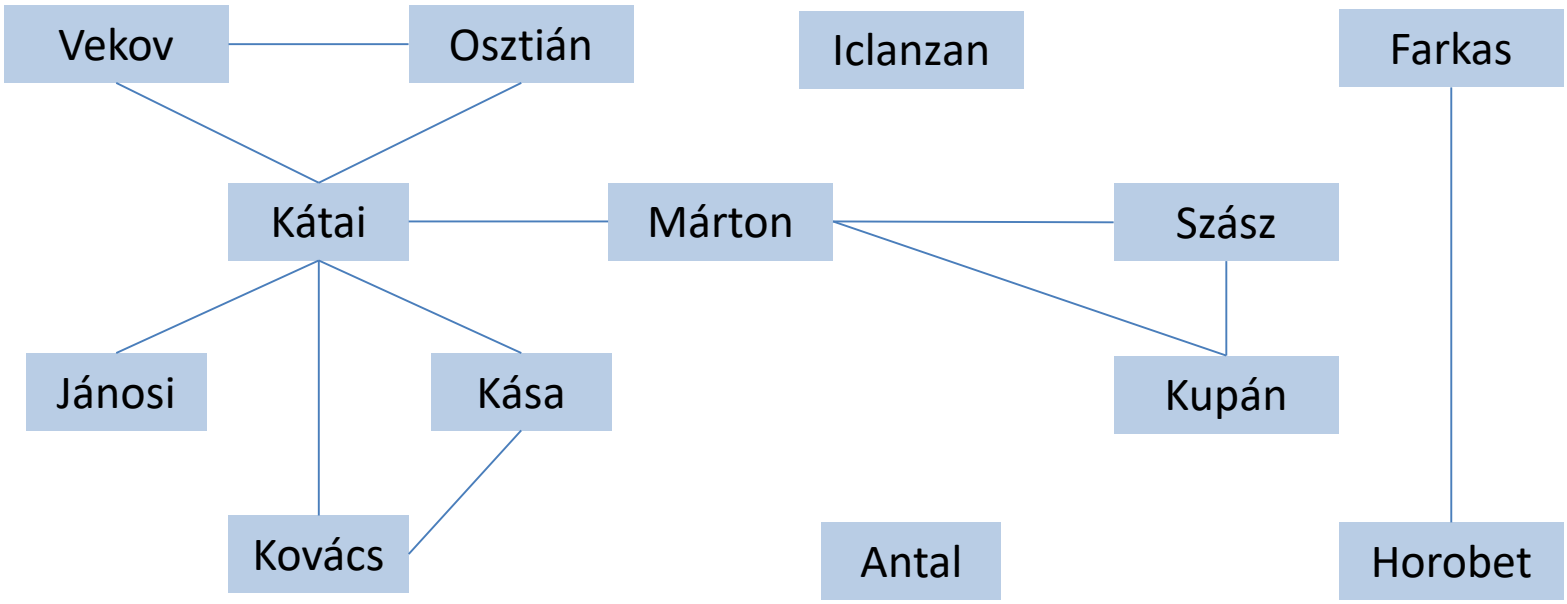


Euler

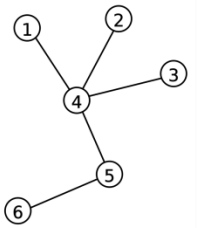


Hamilton

Néhány alapfogalom
összefüggőség (komponensek)

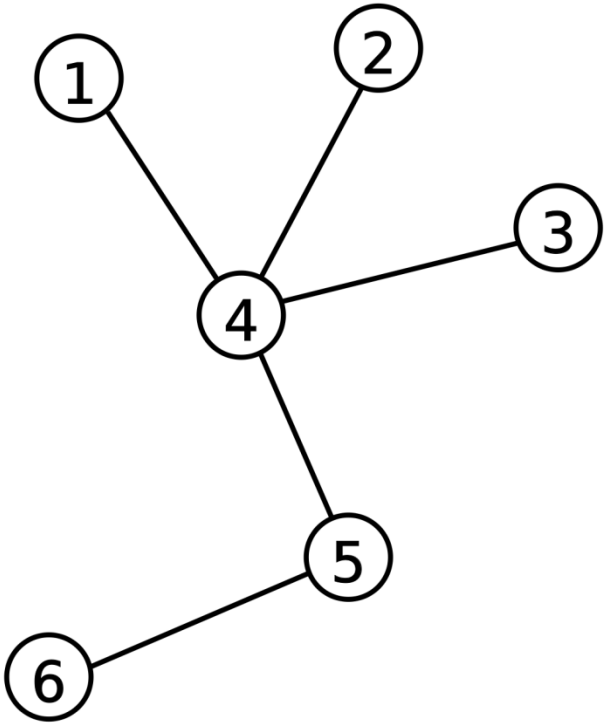


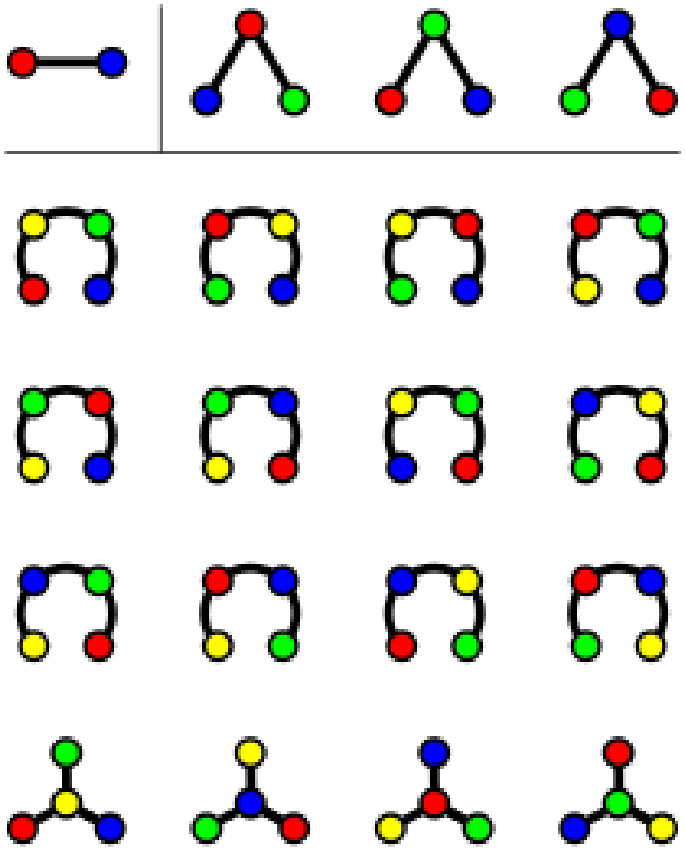
Néhány alapfogalom
fa (erdő)

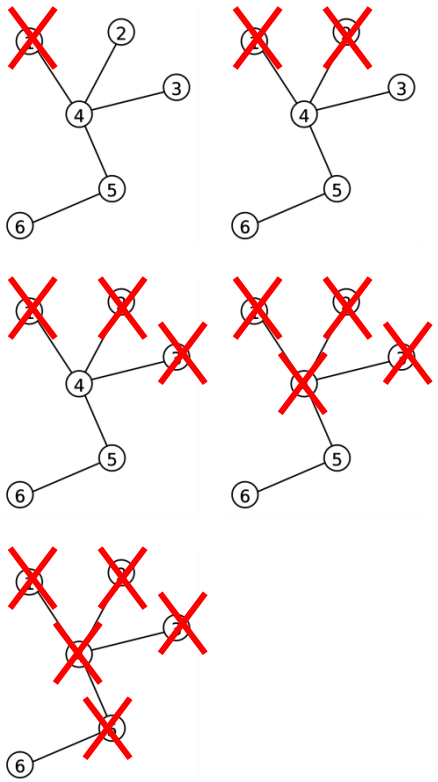


Cayley: n^{n-2}

Prüfer kód



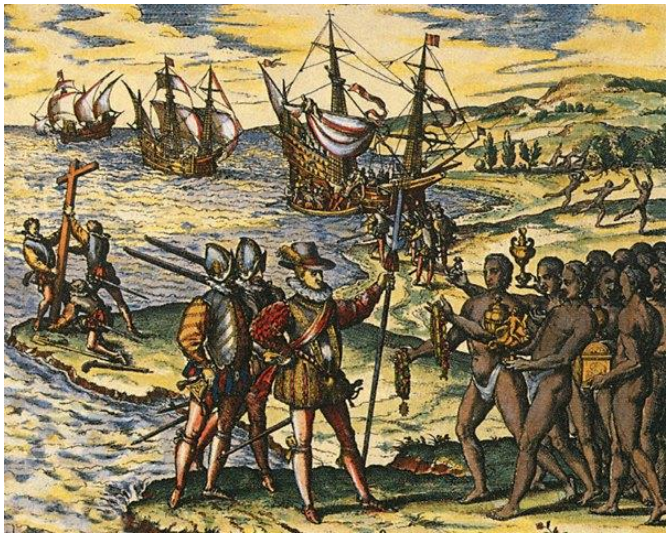


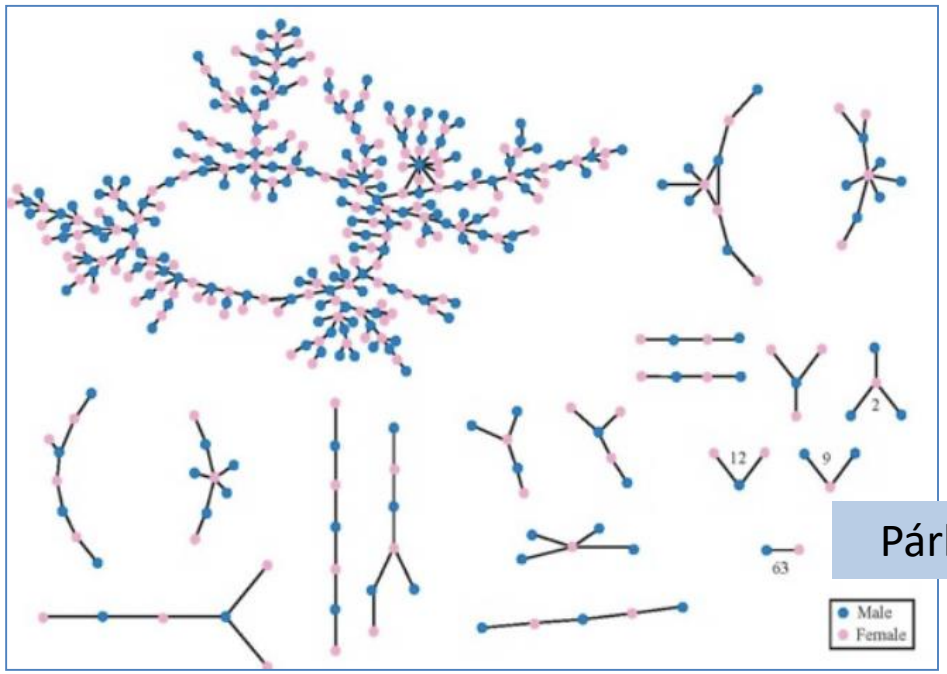


4,4,4,5,~~6~~

Óriás összefüggő komponens

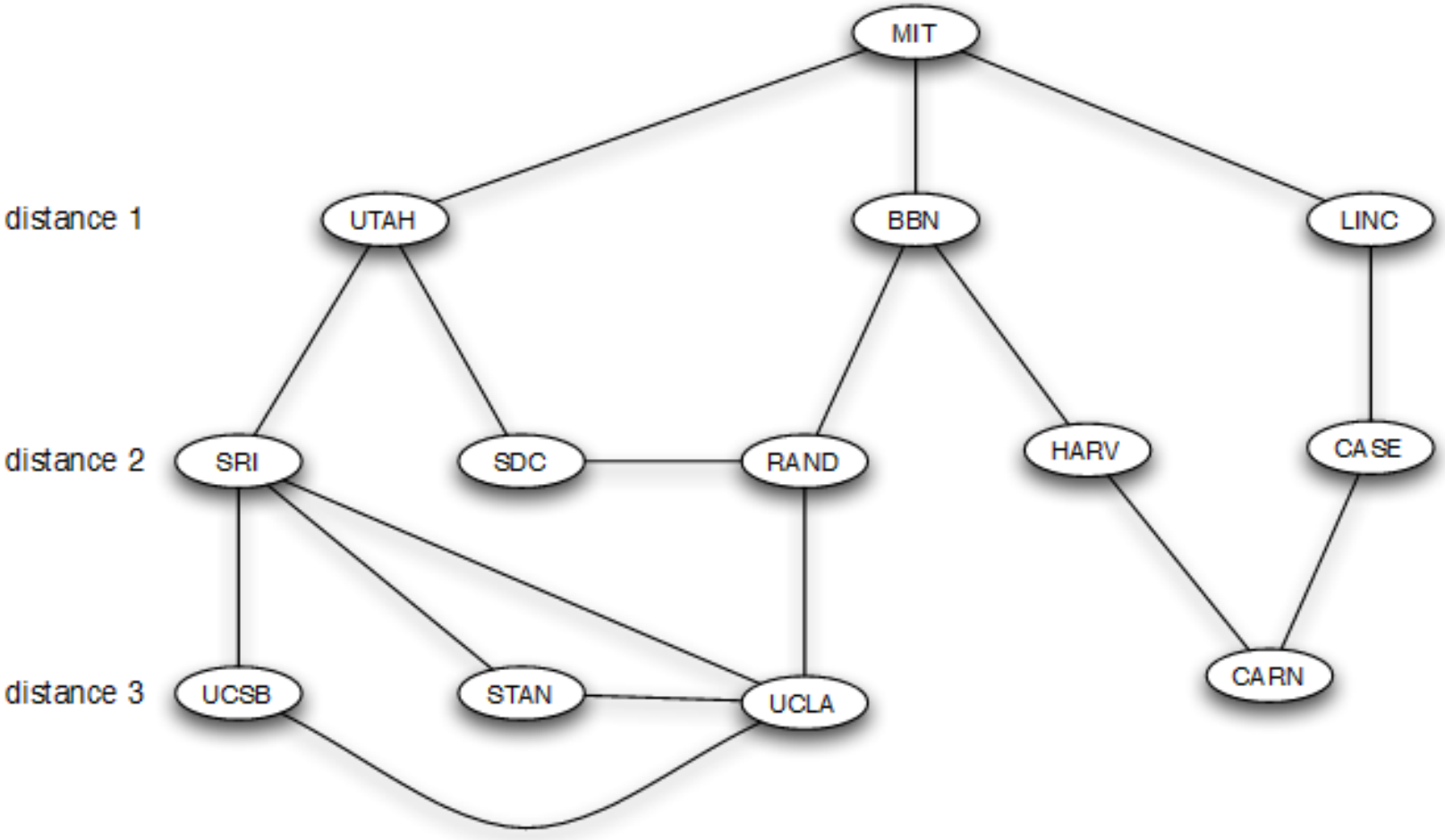
- Baráti kapcsolatok világhálója
  - A németországi barátom szüleinek barátai…
- Két óriás komponens
  - Eurázsia vs. Amerika



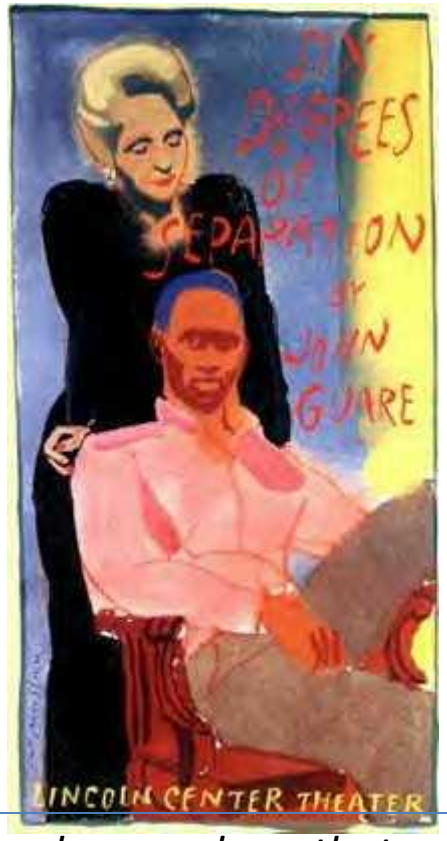


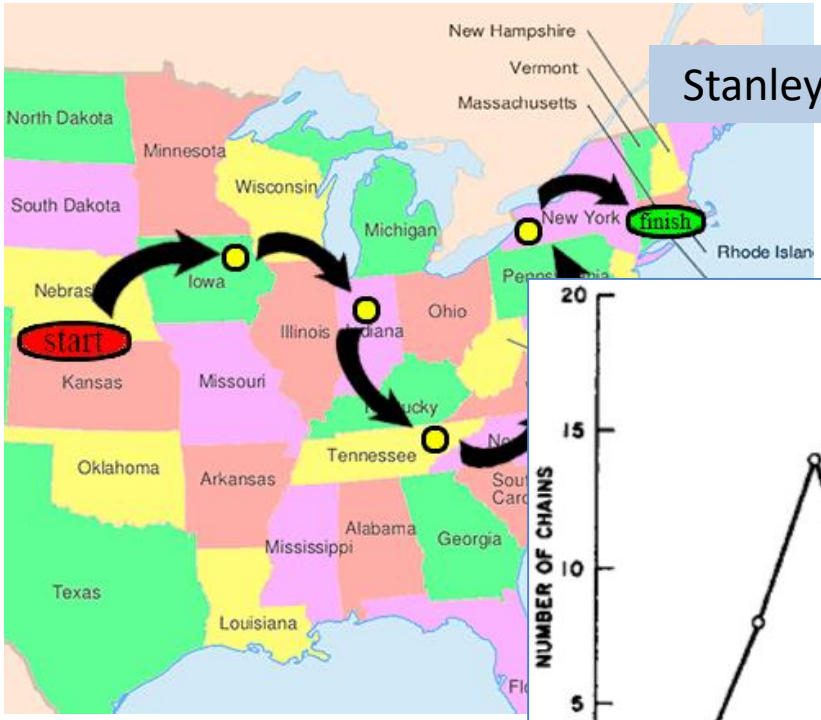
Párkapcsolatok: amerikai líceum (18 hónap)

Néhány alapfogalom
távolság (legrövidebb utak, BFS)

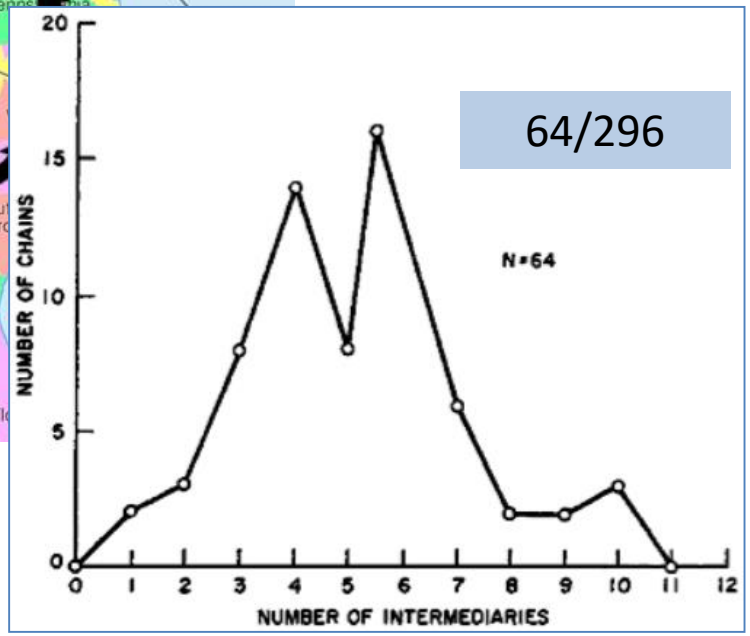


Kisvilág-tulajdonság (1)



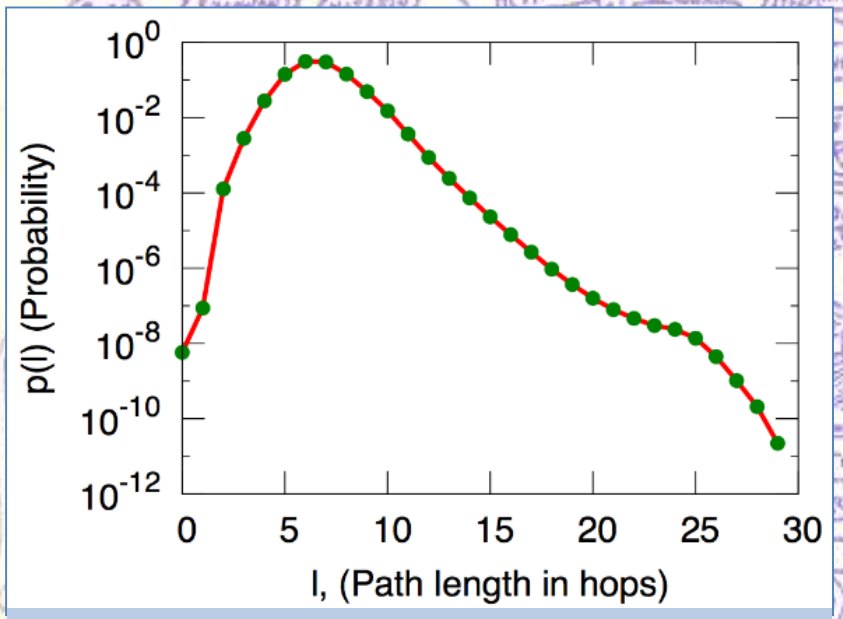


Stanley Milgram et al., 1960s



I read somewhere that everybody on this planet is separated by only six other people. Six degrees of separation. Between us and everybody else on this planet. The president of the United States. A gondolier in Venice.

Kisvilág-tulajdonság (2)



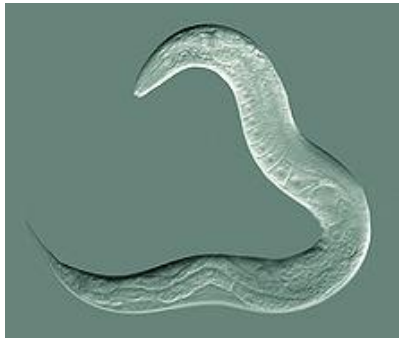
Microsoft Instant Messenger (1 hónap)
1.000 / 240.000.000 aktív felhasználó

Erdős Pál, ~1500 dolgozat
Erdős szám: ~4-5
Einstein: 2
Kása Zoltán: 3
http://www.oakland.edu/enp/

Bacon number (Kevin Bacon, 1994)
Internet Movie Database (átlag: 2.9)
Plenniki Morya (7, 1928 szovjet kalóz-film)

Hálózati adatbázisok

- Collaboration Graphs
  - társszerző / társszínész
  - World-of-Warcraft collaboration graph
- Who-talks-to-Whom Graphs
  - Sapis MIT projekt (Kátai-Jánosi-Bogosi)
- Information Linkage Graphs
  - Wikipedia / facebook
  - idézések elemzése
- Technological Networks
- Networks in the Natural World
  - mi-fogyaszt-mit kapcsolatok (veszélyeztetett fajok)
  - Neuron-hálók (C. elegans: 302 pont, 7000 él)

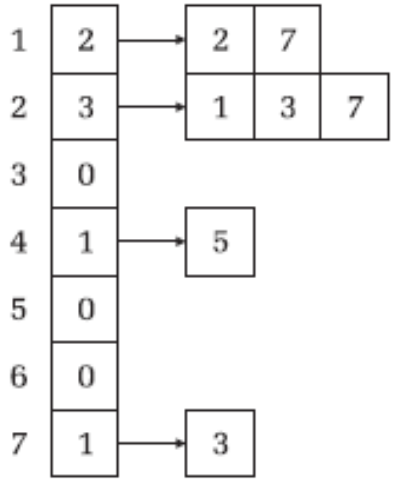


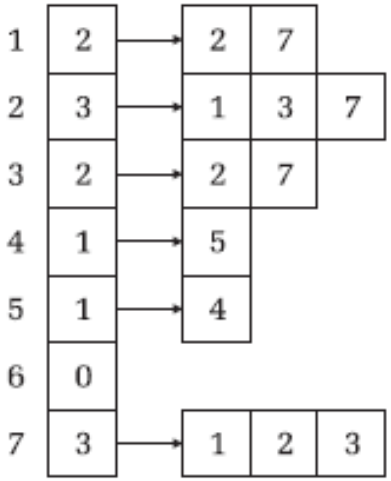
Gráfok ábrázolása a memóriában

- Él-lista
- Szomszédsági-mátrix
- Szomszédsági-lista

1	2	3	4	5	6	7
1	**2**	**7**	**1**	**4**	**2**	**2**
2	**3**	**3**	**7**	**5**	**7**	**1**

1	2	3	4	5	6
1	**2**	**7**	**1**	**4**	**2**
2	**3**	**3**	**7**	**5**	**7**

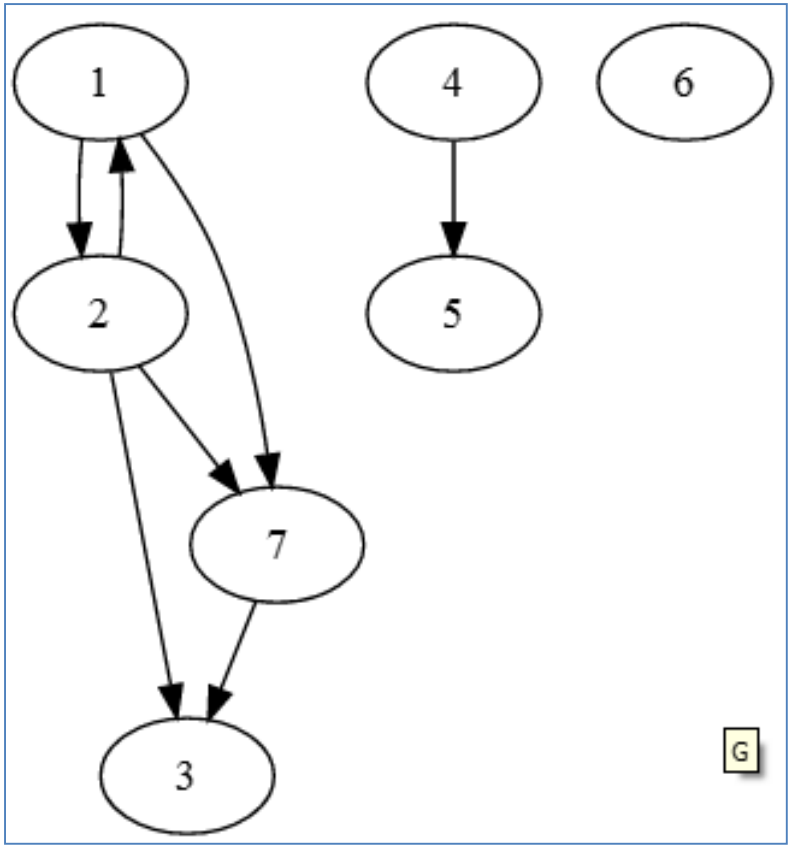


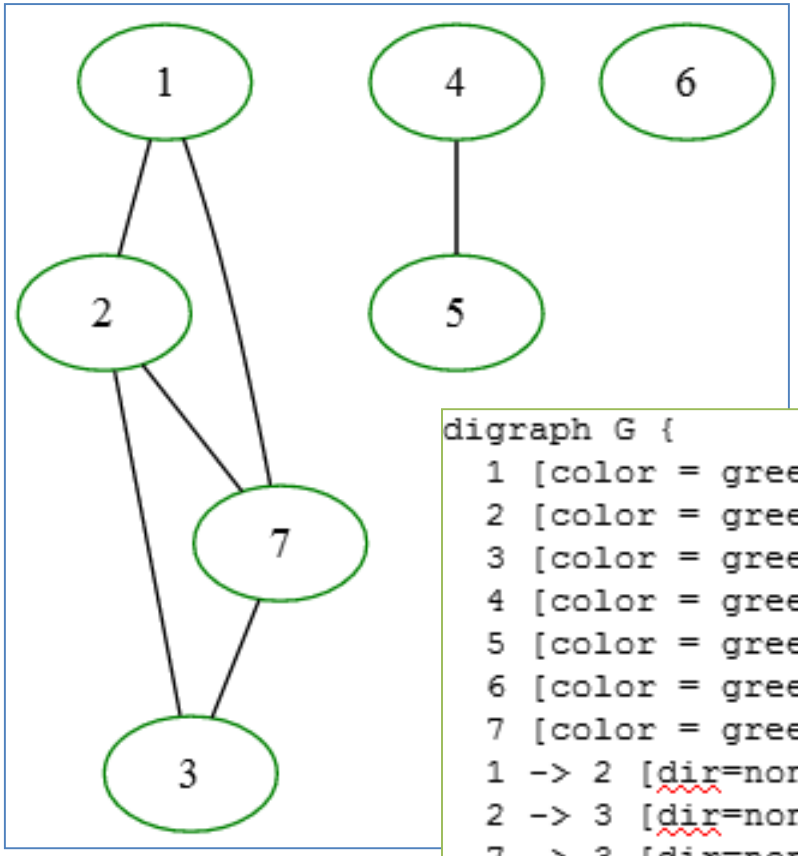


	1	2	3	4	5	6	7
1	0	1	0	0	0	0	1
2	1	0	1	0	0	0	1
3	0	0	0	0	0	0	0
4	0	0	0	0	1	0	0
5	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0
7	0	0	1	0	0	0	0

	1	2	3	4	5	6	7
1	0	1	0	0	0	0	1
2	1	0	1	0	0	0	1
3	0	1	0	0	0	0	1
4	0	0	0	0	1	0	0
5	0	0	0	1	0	0	0
6	0	0	0	0	0	0	0
7	1	1	1	0	0	0	0

http://www.webgraphviz.com/






```
digraph G {
  1 [color = green]
  2 [color = green]
  3 [color = green]
  4 [color = green]
  5 [color = green]
  6 [color = green]
  7 [color = green]
  1 -> 2 [dir=none]
  2 -> 3 [dir=none]
  7 -> 3 [dir=none]
  1 -> 7 [dir=none]
  4 -> 5 [dir=none]
  2 -> 7 [dir=none]
}
```

Mini-projekt (3 hét)

- A MIT tagok publikációs listái alapján építsd fel az alábbi együttműködési gráfot:
  - Ha két tagnak van közös cikke, akkor van köztük él
  - Az élek súlya legyen a közös cikkek száma
- Szomszédsági listából határozd meg a gráf összefüggő komponenseit
- Generáld a „graphviz kódot”

Továbbiakban…

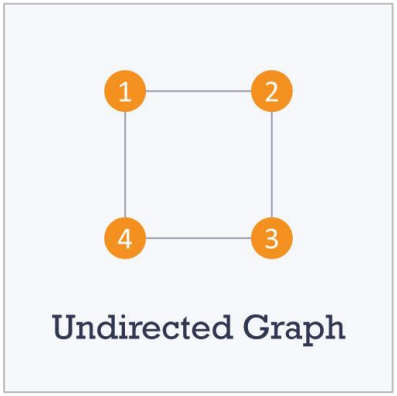
2. Fókuszban az élek (hidak)
3. Fókuszban a pontok
4. Artikulációs pontok
5. Klikkek (teljes gráf)

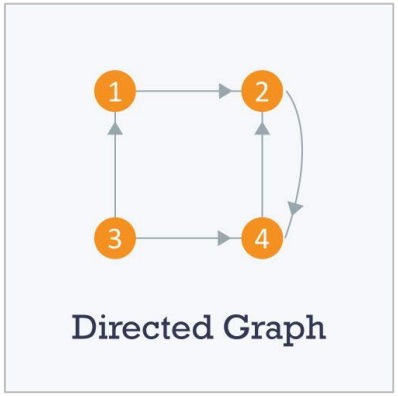

NETWORKISM
Cultural Meme
© Tomas Saraceno (2008)

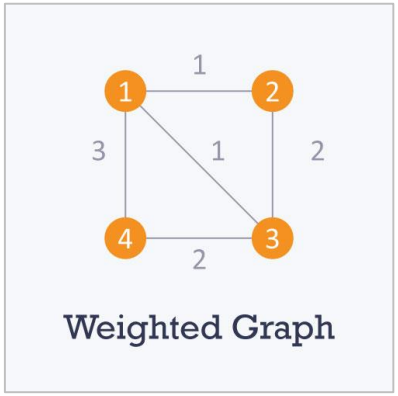
DFS-BFS

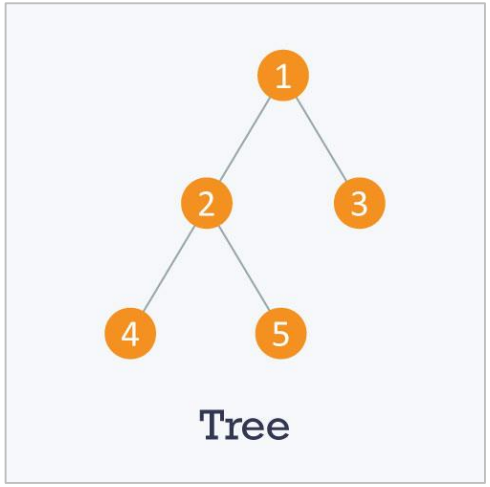
ismétlés

Gráfok

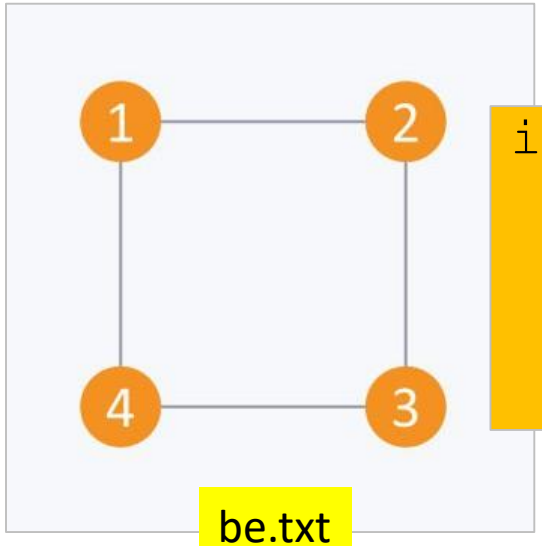

1
2
4
3
Undirected Graph


1
2
3
4
Directed Graph


1
1
2
3
1
2
4
3
2
Weighted Graph


1
2
3
4
5
Tree

Szomszédsági MÁTRIX (A)

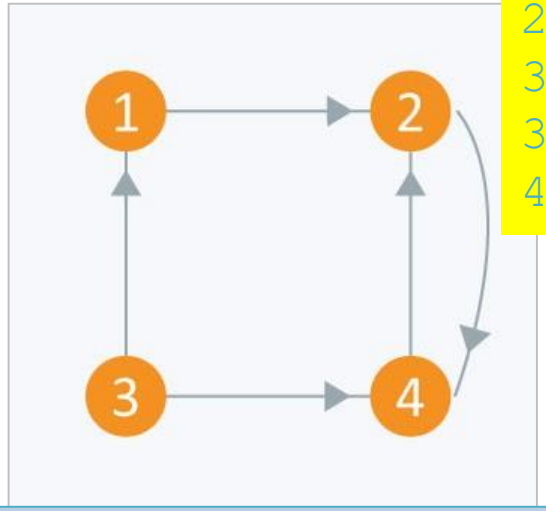


i/j	:	1	2	3	4
1	:	0	1	0	1
2	:	1	0	1	0
3	:	0	1	0	1
4	:	1	0	1	0

be.txt

nodes **4** **4** edges

```
1 2
2 3
3 4
1 4
```

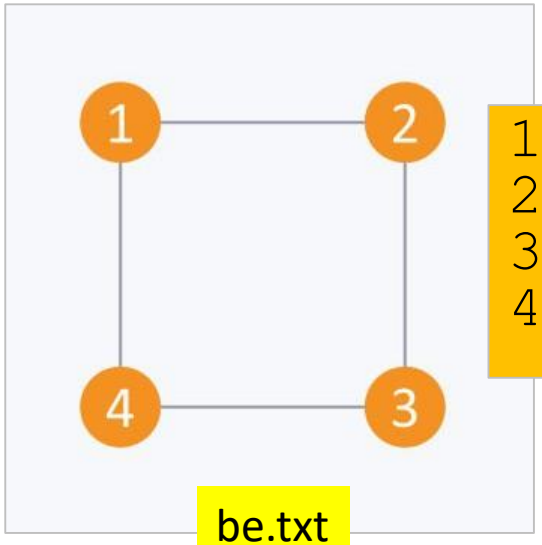


be.txt

```
4 5
1 2
2 4
3 1
3 4
4 2
```

```
for( i = 0 ; i < edges ; ++i ){
   cin >> x >> y;
   A[x][y] = 1;
}
```

Szomszédsági LISTA (adjList)

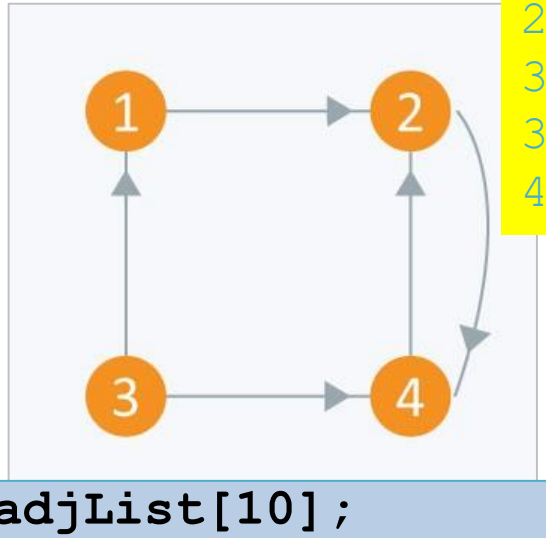



```
1: → 2 → 4
2: → 1 → 3
3: → 2 → 4
4: → 1 → 3
```

be.txt

nodes **4** **4** edges

```
1 2
2 3
3 4
1 4
```

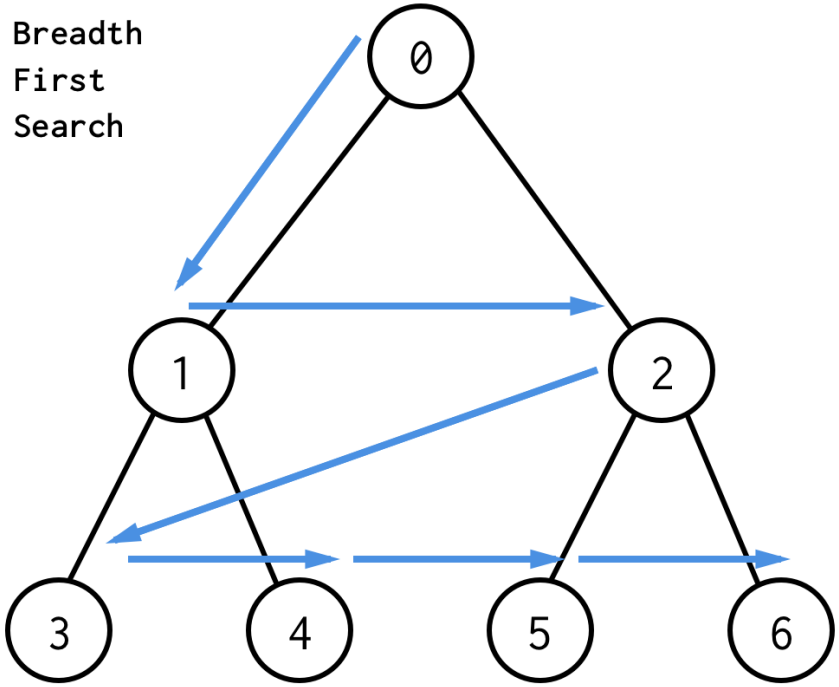


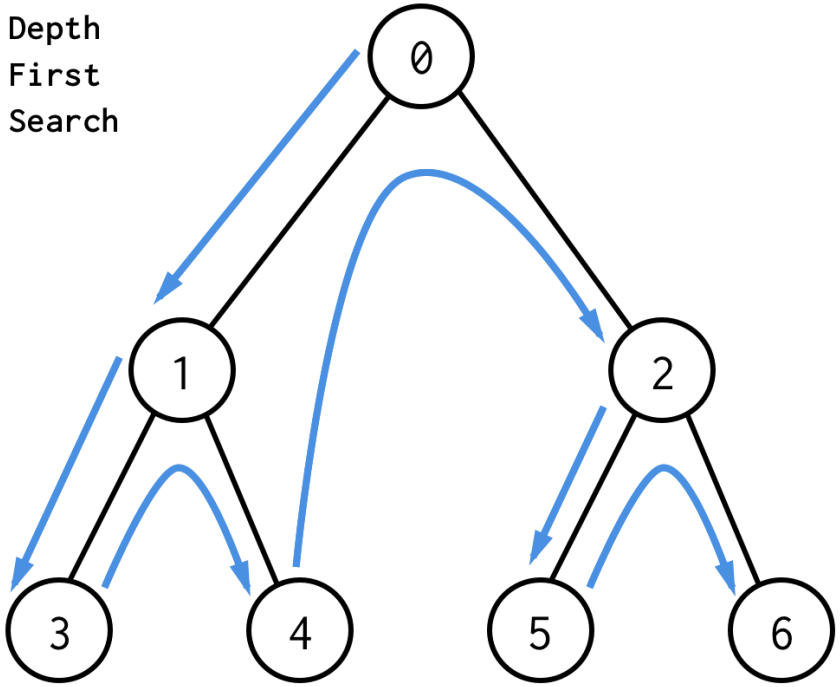
be.txt

```
4 5
1 2
2 4
3 1
3 4
4 2
```

```
vector <int> adjList[10];
for( i = 0 ; i < edges ; ++i ){
   cin >> x >> y;
   adjList[x].push_back(y);
}
```

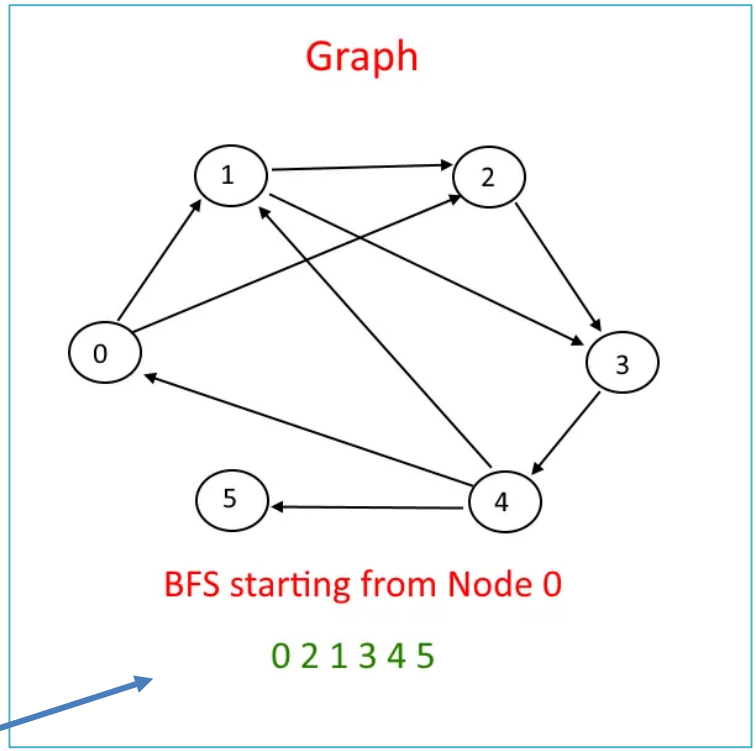
Szélességi/Mélységi bejárás (BFS/DFS)





BFS

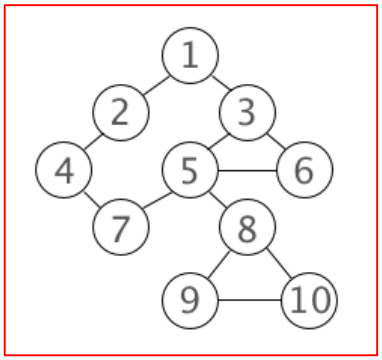
- Queue
  - **0**
  - 0 **1** 2
  - 0 1 **2** 3
  - 0 1 2 **3**
  - 0 1 2 3 **4**
  - 0 1 2 3 4 **5**
  - 0 1 2 3 4 5



A BFS sorrend nem egyértelmű
Konvenció: a szomszédokat, ID-jük szerint növekvő serrendben tekintjük

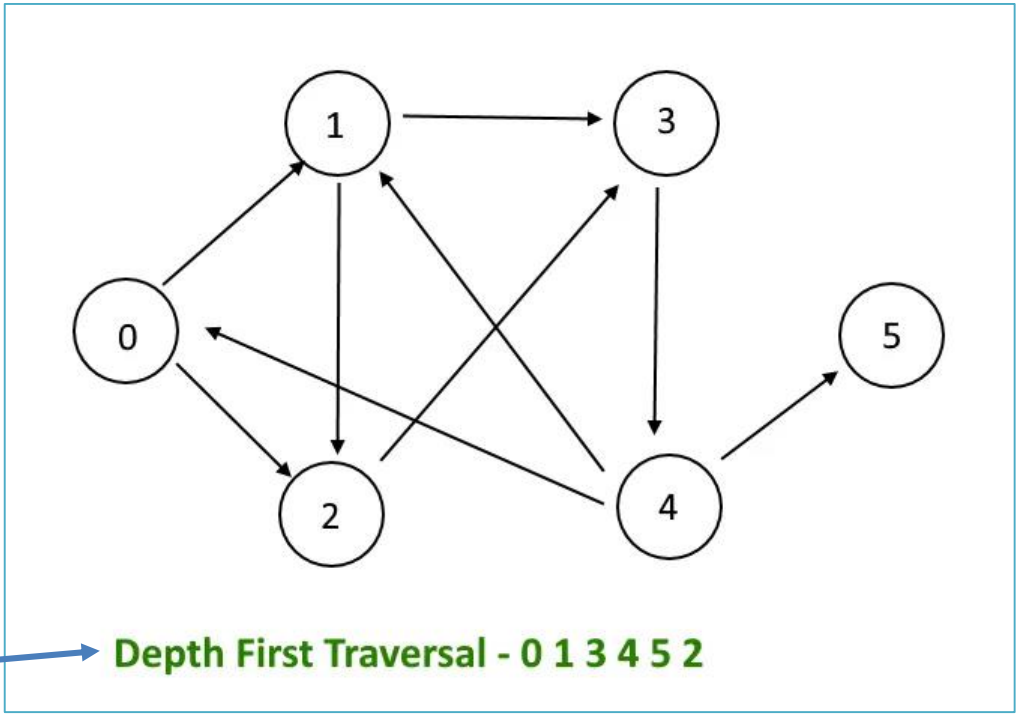
```
vector <int> adjList[10];  //Vector for maintaining adjacency list
bool visited[10];          //Mark the node if visited


void bfs(int s) {
  queue <int> q;
  q.push(s); visited[s] = true;

  while( !q.empty() ){
    int u = q.front(); q.pop();
    for( int i = 0 ; i < adjList[u].size() ; i++ ){
      int v = adjList[u][i];
      if( visited[v] == false ){
        q.push(v); visited[v] = true;
      }
    }
  }
}
```

```
vector <int> adjList[10];  //Vector for maintaining adjacency list
int dist[10];              //To determine the level of each node
bool visited[10];          //Mark the node if visited


void bfs(int s) {
   queue <int> q;
   q.push(s);  dist[s] = 0; visited[s] = true;

   while( !q.empty() ){
      int u = q.front(); q.pop();
      for( int i = 0 ; i < adjList[u].size() ; i++ ){
         int v = adjList[u][i];
         if( visited[v] == false ){
            dist[v] = dist[u]+1;
            q.push(v); visited[v] = true;
         }
      }
   }
}
```



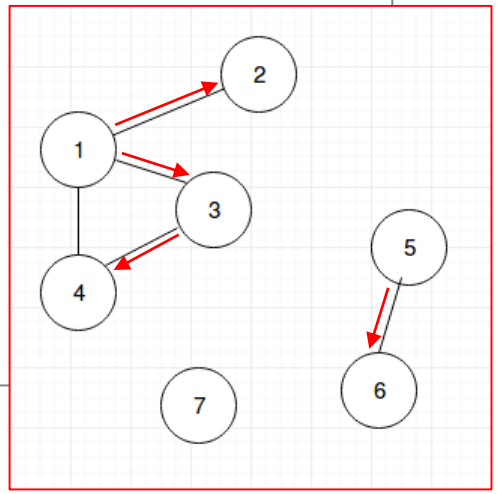
DFS

- Stack (rekurzió)
  - `0 1 2 3 4 5`

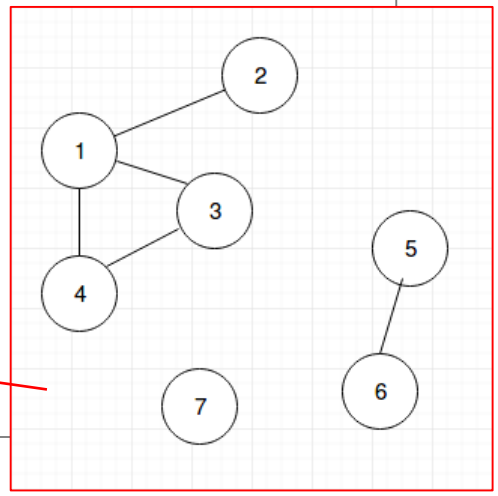


A DFS sorrend nem egyértelmű
Konvenció: a szomszédokat, ID-jük szerint növekvő serrendben tekintjük

```
vector <int> adjList[10];  //Vector for maintaining adjacency list
bool visited[10];          //Mark the node if visited

void dfs(int u) {
   visited[u] = true; // cout << u;
   for( int i = 0 ; i < adjList[u].size() ; ++i ){
      int v = adjList[u][i];
      if( visited[v] == false ){
         dfs(v);
      }
   }
}


…
for( int i = 1 ; i <= nodes ; ++i ){
   if( visited[i] == false ) {
      dfs(i);
   }
}
```



```
vector <int> adjList[10];  //Vector for maintaining adjacency list
bool visited[10];          //Mark the node if visited
connectedComponents = 0;

void dfs(int u) {
   visited[u] = true; // cout << u;
   for( int i = 0 ; i < adjList[u].size() ; ++i ){
      int v = adjList[u][i];
      if( visited[v] == false ){
         dfs(v);
      }
   }
}


…
for( int i = 1 ; i <= nodes ; ++i ){
   if( visited[i] == false ) {
      dfs(i);
      ++connectedComponents;
   }
}
```



```
vector <int> adjList[10];  //Vector for maintaining adjacency list
bool visited[10];          //Mark the node if visited
k = n;

void dfs(int u) {
   visited[u] = true; // cout << u;
   for( int i = 0 ; i < adjList[u].size() ; ++i ){
      int v = adjList[u][i];
      if( visited[v] == false ){
         dfs(v);
      }
   }
   topo_order[k--] = u;
}
…
for( int i = 1 ; i <= nodes ; ++i ){
   if( visited[i] == false ) {
      dfs(i);
   }
}
```