Big Data
Algorithmic Introduction

Prof. Yuval Shavitt

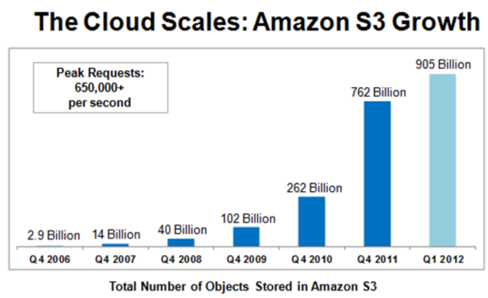
Logistics

- Contact: shavitt@eng.tau.ac.il
- Final grade:
  - 4-6 home assignments (will try to include programing assignments as well): 20%
  - Exam 80%

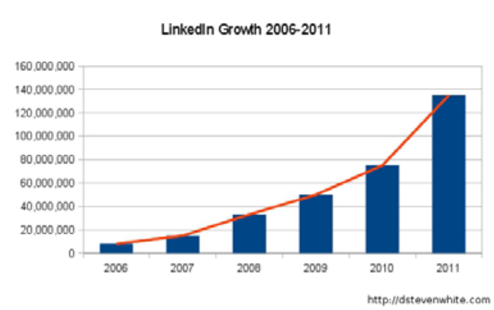
Big Data

- Today we have huge datasets:
  - Social networks
  - Biological networks
  - Consumer data
  - Transportation data
  - Internet data
- Their analysis require new approaches

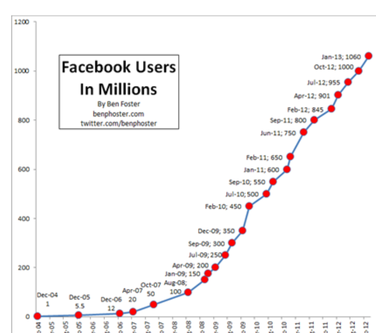
How many *objects* do we store?



LinkedIn

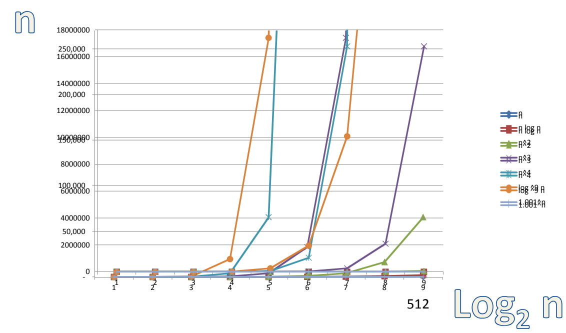


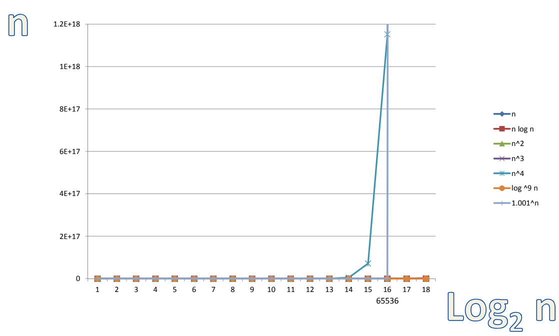
Analyzing the Facebook graph?

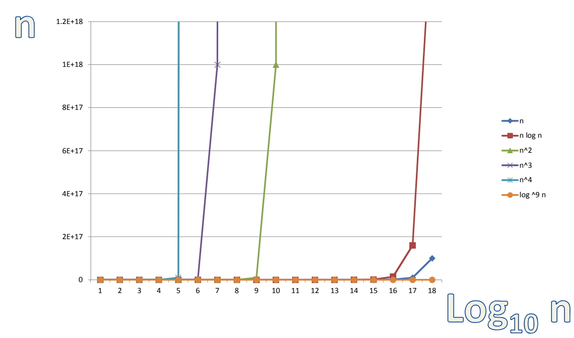


Complexity Basics

- We are interested in algorithms that are efficient
  - O(n) is better than O(n^2)
  - O(n) is better than O(n log n)
  - O($\log^5$ n) is better than O(n)
  - O(n^3) is better than O($(1+\varepsilon)^n$)







When Data is Large?

Definition 1: When data cannot be stored on the machine RAM

- If the algorithm is not sequential

Definition 2: When “regular” algorithms fail

- Too slow
  - An O(n^2) algorithm when n=2,000,000 will run for 8 hours if the O constant is 1
- Too deep stack

Where is Our Data?

- Centralized
  - oracle
- Distributed
- Streamed

Thinking about Algorithms

- Our thinking about algorithm is 80 years old
- Worst case analysis
  - Average case is marginalized
  - No attempt to define working regimes
  - Constants are ignored (even logs $\tilde{O}()$)

Shortest Path

- Given a graph G(V,E) with non-negative edge weights, find the shortest path between two vertices.
- Best algorithm?
- Dijkstra with complexity O(V log V+E)
  - |V| times we select the node closest to source
    - Costs O(V)

Shortest Path

- Given a graph G(V,E) with non-negative edge weights, find the shortest paths from a vertice to all others.
- Best algorithm?
- Dijkstra with complexity O(V log V+E)

Dijkstra

- Complexity O(V log V+E)
  - |V| times we select the node closest to source
    - Costs O(V)
  - |E| we relax and edge
- Why not O(V^2+E)?
  - Q can be implemented with a *Binary heap*
    - O(E log V)
  - Q can be implemented with a *Fibonacci heap*
    - O(V log V+E)

```
DIJKSTRA(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
S ← ∅
Q ← V[G]
while Q ≠ ∅
      do u ← EXTRACT-MIN(Q)
         S ← S ∪ {u}
         for each vertex v ∈ Adj[u]
             do RELAX(u, v, w)
```

But wait, what does the bible say?

heaps.

From a practical point of view, however, the constant factors and programming complexity of Fibonacci heaps make them less desirable than ordinary binary (or *k*-ary) heaps for most applications. Thus, Fibonacci heaps are predominantly of theoretical interest. If a much simpler data structure with the same amortized time bounds as Fibonacci heaps were developed, it would be of great practical use as well.

Like a binomial heap, a Fibonacci heap is a collection of trees. Fibonacci

This leaves us with a complexity of O(E log V)

and we still have to implement the heap

Shortest Path with Bellman-Ford

procedure BellmanFord(*list* vertices, *list* edges, *vertex* source)
//Step 1: initialize graph
for each vertex v **in** vertices:
    if v **is** source **then** distance[v] := 0
        else distance[v] := **infinity**
            predecessor[v] := **null**
// Step 2: relax edges repeatedly
for i **from** 1 **to** size(vertices)-1:
    for each edge (u, v) **with** weight w **in** edges:
        if distance[u] + w < distance[v]:
            distance[v] := distance[u] + w
            predecessor[v] := u

O(V E)

B-F Discussion

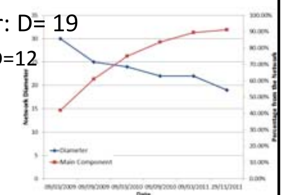
- O(V E) is worse even than O(E log V)

// Step 2: relax edges repeatedly
for i **from** 1 **to** ~~size(vertices)~~-1:   **D+1**
    for each edge (u, v) **with** weight w **in** edges:
        if distance[u] + w < distance[v]:
            distance[v] := distance[u] + w
            predecessor[v] := u

What is the diameter of a *real* graph?
    log V (or less)
Easy to implement – no hidden costs

Example: The STEAM Network

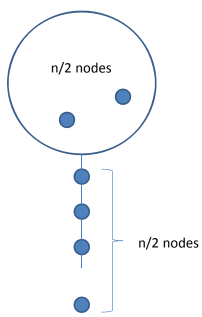
- A network of gamers
- |V|=9,000,000 |E|= 82,000,000
  |E|/|V|=18.2
- LCC has 8,244,178 nodes
- Approximated Diameter: D= 19
  - Removing 22 nodes ⇒ D=12



It is easy to build a bad graph

- But do such graphs exist?
  - In our facebook graph?
  - LinkedIn graph?
  - Any real data graph?

  - Maybe in a future graph?



Algorithmic Approaches

- Find better exact algorithms
  - Maybe for special cases: sparse graphs, bounded degree, bounded diameter, …
- Approximation algorithms
  - ε, constant, logarithmic approximation
  - Polynomial running time
- Probabilistic approximation
  - Sometimes sublinear
- heuristics

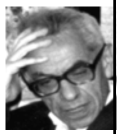
(ε,δ) – Approximation

- An algorithm for a estimating f is an (ε,δ)-approximation if it takes an input instance and two real values ε, δ and produces an output y such that

$$\Pr[(1-\varepsilon)\cdot f \le y \le (1+\varepsilon)\cdot f] \ge 1-2\delta$$

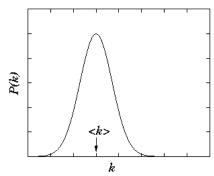
Graphs

Random graphs in Mathematics
The Erdös-Rényi model



- Generation:
  - create n nodes.
  - each possible link is added with probability p.
- Number of links: np
- If we want to keep the number of links linear, what happen to p as $n \to \infty$?

Poisson distribution



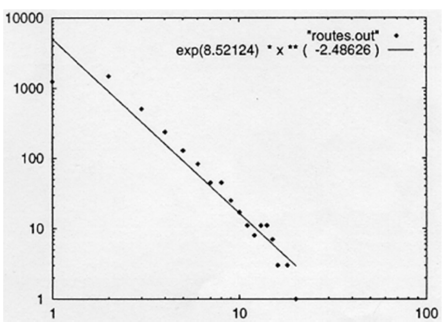
The Barabasi-Albert Model

- Noticed that many graphs have a similar structure
  - No characteristic degree
  - Most nodes have small degree
  - The number of nodes with high degree declines polynomialy (not exponentially)
    - Long tail

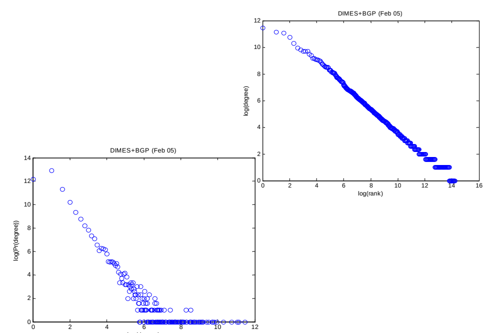
The Faloutsos Graph

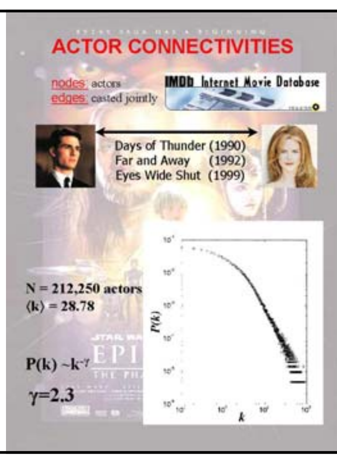
1995 Internet router topology

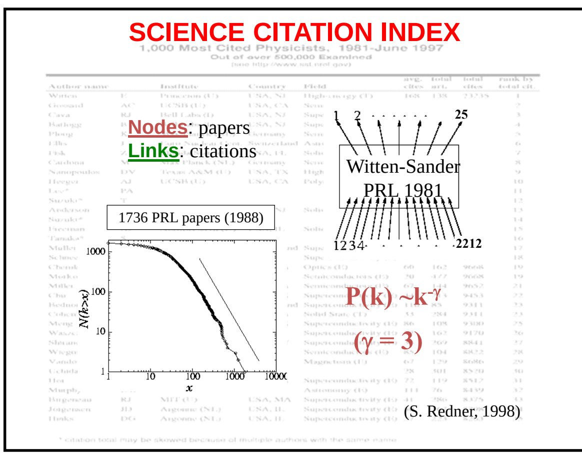
3888 nodes, 5012 edges, <k>=2.57



Degree Dist. & Zipf Plot



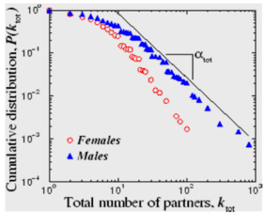




Sex-web

Nodes: people (Females; Males)
Links: sexual relationships

4781 Swedes; 18-74;
59% response rate.

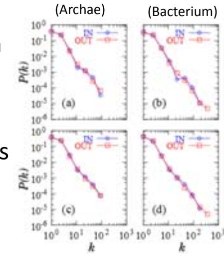


Liljeros et al. Nature 2001

Metabolic Network

- A graph representation of the biochemical reaction in a metabolic network
- nodes= substrates
- Edges = metabolic reactions
- Node degree
  - In = participate as product
  - Out = participate as educt

A. fulgidus (Archae) | E. Coli (Bacterium)



C. Elegans (Eukaryyote) | average of 43 organism

Jeong et al. Nature 2000

SCALE-FREE NETWORKS

(1) The number of nodes (N) is NOT fixed.

Networks continuously expand by the addition of new nodes

Examples:
WWW : addition of new documents
Citation : publication of new papers

(2) The attachment is NOT uniform.

A node is linked with higher probability to a node that already has a large number of links.

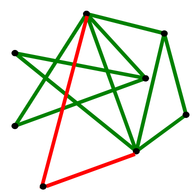
Examples :
WWW : new documents link to well known sites (CNN, YAHOO, NewYork Times, etc)
Citation : well cited papers are more likely to be cited again

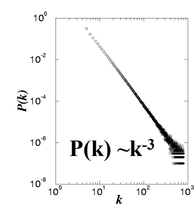
Scale-free model

(1) GROWTH :
At every timestep we add a new node with m edges (connected to the nodes already present in the system).

(2) PREFERENTIAL ATTACHMENT :
The probability Π that a new node will be connected to node i depends on the connectivity k_i of that node

$$\Pi(k_i) = \frac{k_i}{\Sigma_j k_j}$$





A.-L.Barabási, R. Albert, Science **286,** 509 (1999)