Introduction to Algorithm Engineering

Spring 2022

Lecture 1

Agenda for Today

- Welcome back to a new semester !!
  - We are hopefully seeing the last embers of the pandemic.
  - Nevertheless, we have to prepare for a possibly new normal in many ways.
  - We will do in-class teaching as much as possible for Spring 2022.
  - Stay safe, but curious!

Agenda for Today

- Welcome back to a new semester !!
  - We are hopefully seeing the last embers of the pandemic.
  - Nevertheless, we have to prepare for a possibly new normal in many ways.
  - We will do in-class teaching as much as possible for Spring 2022.
  - Stay safe, but curious!

- My details
  - Kishore Kothapalli, Professor, IIIT Hyderabad
  - Email: kkishore@iiit.ac.in (the best way to reach me)
  - Research interests span parallel computing and distributed algorithms.

Agenda for Today

- Rest of Today's Class
  - Syllabus
  - Policies
  - Expectations
  - Actual lecture

Syllabus

- Roughly, a three module course
  - Module 1: Basic Concepts – Algorithm Engineering, parallel computing,
  - Module 2: Algorithm Engineering : Graph Algorithms for connectivity and biconnectivity
  - Module 3: Advanced Topics

Evaluation

- Grading (Tentative)
  - Homeworks: 25% (We will have some lateness policy here)
  - Course Project: 25%
  - End Exam: 35%
  - Quiz : 15%
  - Exceptional Performance: 5% extra, Exception performance in any component gets extra score.

- Any submission that is graded and evaluated should not be copied from any source.
- Copied submissions will get zero for the first instance and negative for repeat offences.

Textbooks

- Textbooks: Do not own these books just for the class!
  - Book available at http://faculty.iiit.ac.in/~kkishore/book.pdf
  - Introduction to Algorithms, Leiserson et al.
  - Randomized Algorithms, Motwani and Raghavan
  - Introduction to Parallel Algorithms, J. JaJa
  - Other material to be posted on the course website

- Most welcome to write to me if you have any questions.

Expectations

- Utilize class time effectively.
  - Starts with all of you settling by the class time.
  - Ask any question you may have. No question is small to ask.
  - Do not show up late.

Agenda for Today

- On to the actual lecture….
- We will see how algorithm engineering can help in designing and analyzing algorithms.
- Starting with a very simple example...

Module 1

- We will start by answering some important questions wrt Algorithm Engineering.

Algorithm Engineering

- The (typical) algorithm design process
  - Design
  - Correctness
  - Theoretical Analysis
  - Implementation
  - Experimental Studies

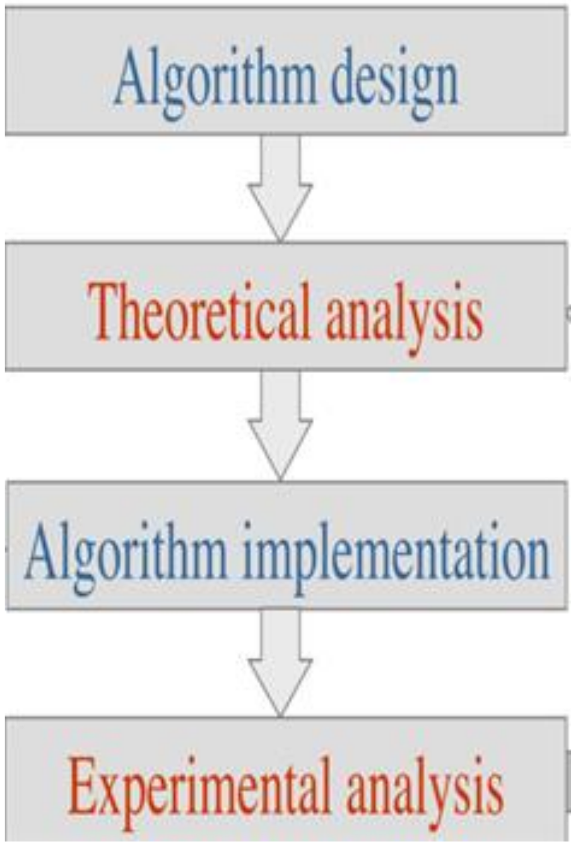


Fig 1: Algorithm Engineering

Picture credits to G. Italiano, 2010

Algorithm Engineering

- The (typical) algorithm design process
  - Design
  - Correctness
  - Theoretical Analysis
  - Implementation
  - Experimental Studies
- We are aware of the above steps
  - Designed multiple algorithms
  - Argued about their correctness
  - Understood their asymptotic run time behavior
  - Implemented some of these algorithms

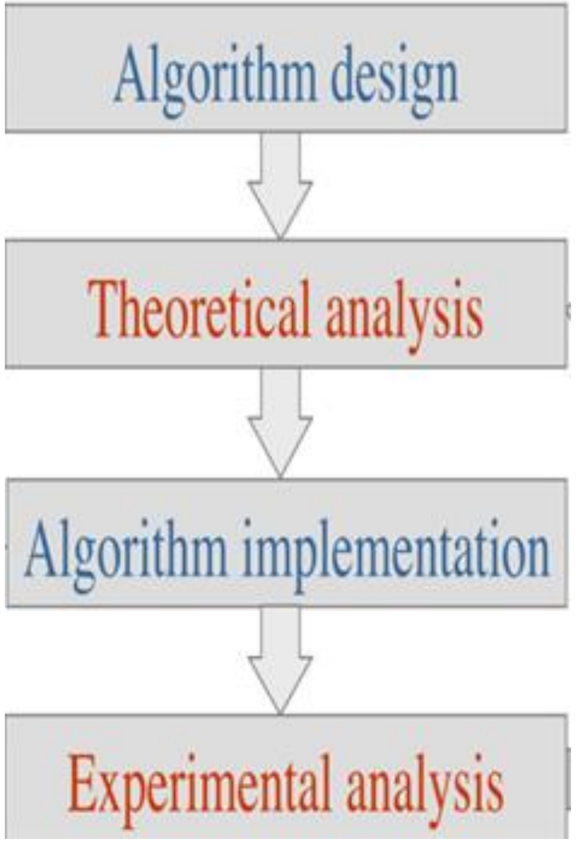


Fig 1: Algorithm Engineering

Picture credits to G. Italiano, 2010

Algorithm Engineering

- A discipline closely aligned with algorithm design
- This discipline weaves the theoretical aspects of algorithms and their practical implementation related aspects.
- Offers interesting insights into the working details of algorithms while not compromising on correctness.

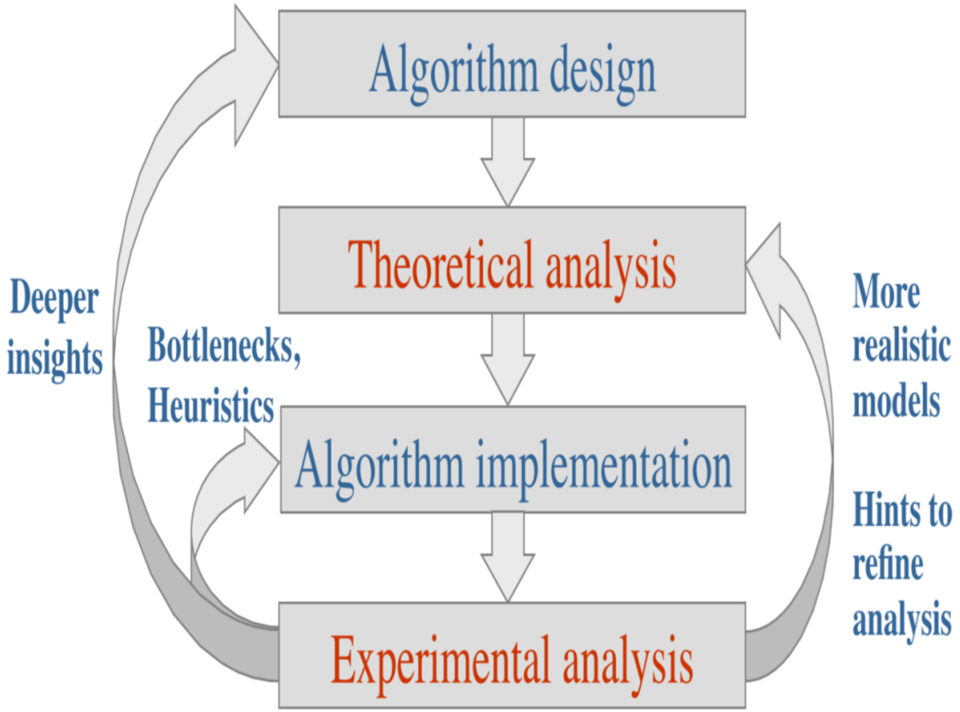


Fig 1: Algorithm Engineering

Picture credits to G. Italiano, 2010

Algorithm Engineering

- Complete the loop from algorithms to experiments to heuristics to algorithms
- Practitioners get to take a deep dive into in both algorithms and systems!

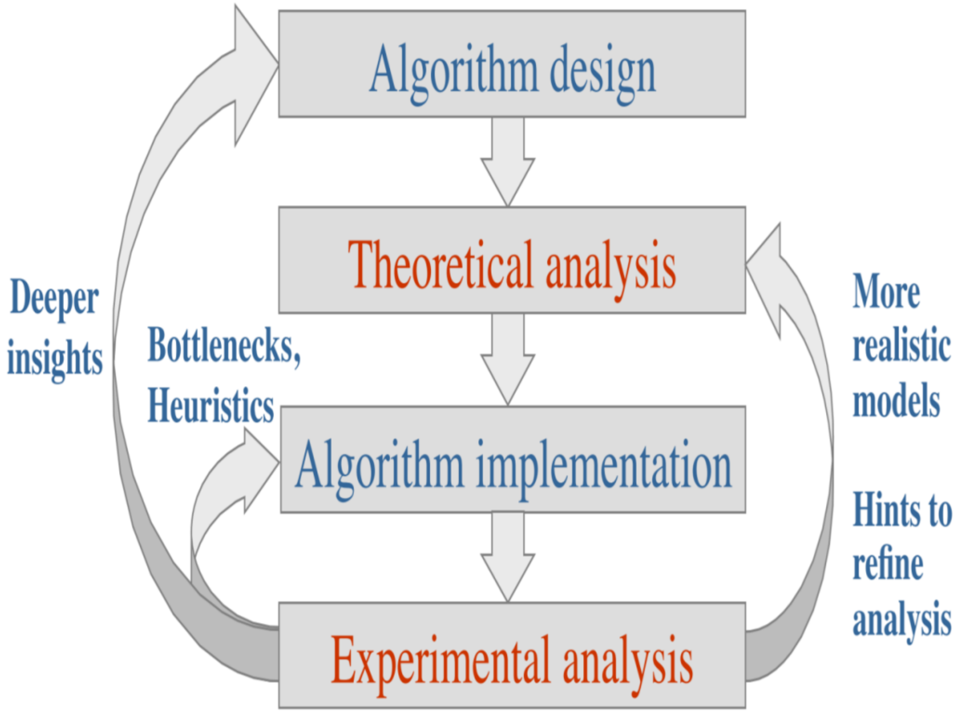


Fig 1: Algorithm Engineering

Picture credits to G. Italiano, 2010

An Example

- Consider any popular algorithm such as the insertion sort.
- We know that it has a worst case run time of $O(n^2)$.
- But its run time has a direct dependence on the number of inversions in the input.
- Recall that an inversion in an array A of n numbers is a pair of indices i and j such that $A[i] > A[j]$ and $i < j$.
- So, we can choose to use bubble sort if we believe that the input is nearly sorted.
- This observation was the key behind the design of a sorting algorithms called “library sort”.

A Better Example

- Let G = (V, E) be an undirected graph on n vertices and m edges.
- The diameter of G is $\max_v \max_w d(v,w)$ where $d(v,w)$ is the shortest path distance between v and w.
  - Another way to write: $\mathrm{diam}(G) = \max_{v,w} d(v,w)$
  - The longest of the shortest distances.
  - Denoted diam(G).

APSP

BFS from each vertex

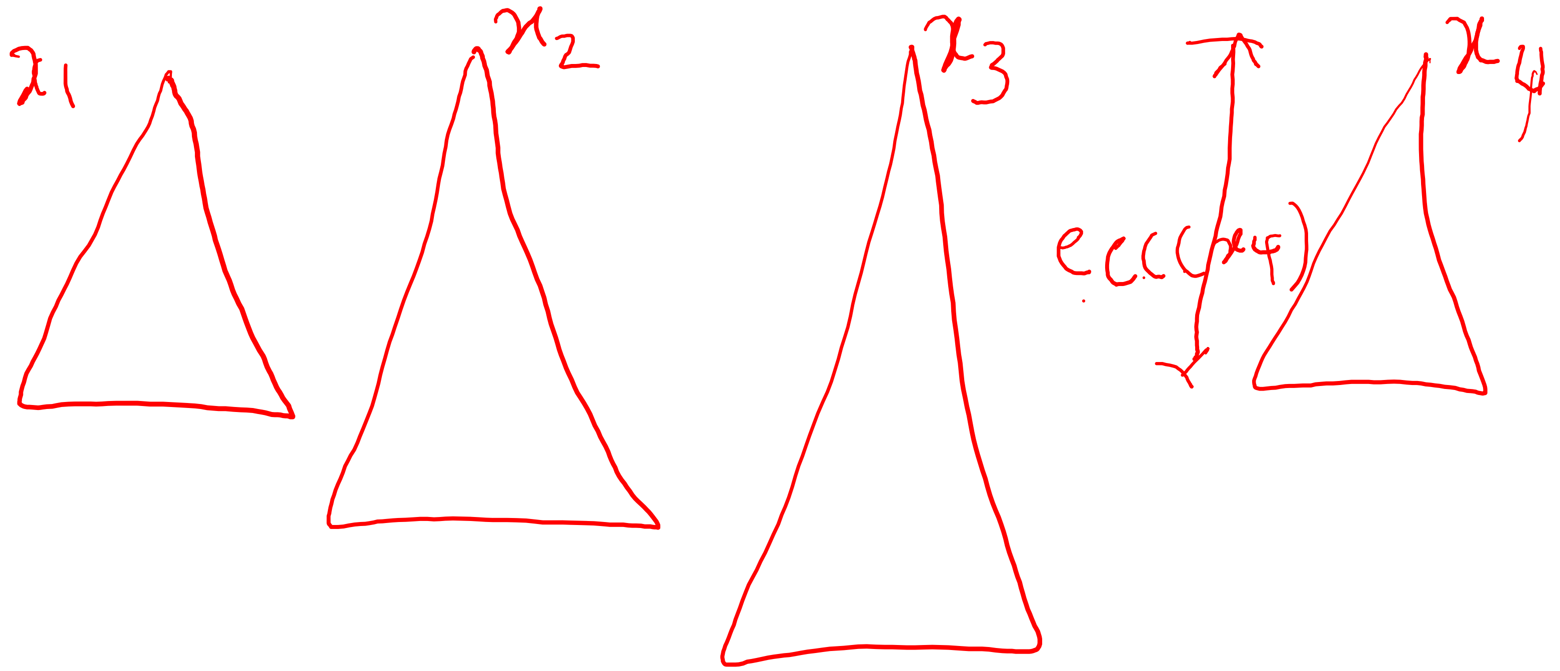
Dynamic Prog.

Diameter of a Graph

- The classical approach would be to use BFS from each vertex.
- The BFS from vertex v can be used find the farthest distance to some vertex w from v.
  - This quantity is also called as the eccentricity of v, denoted ecc(v).
- The diameter of G is the maximum over all v, ecc(v).
- This algorithm has a run time of O(n(n+m)) since each BFS runs in time O(n+m).
- So far, there is no known algorithm that has a better run time.

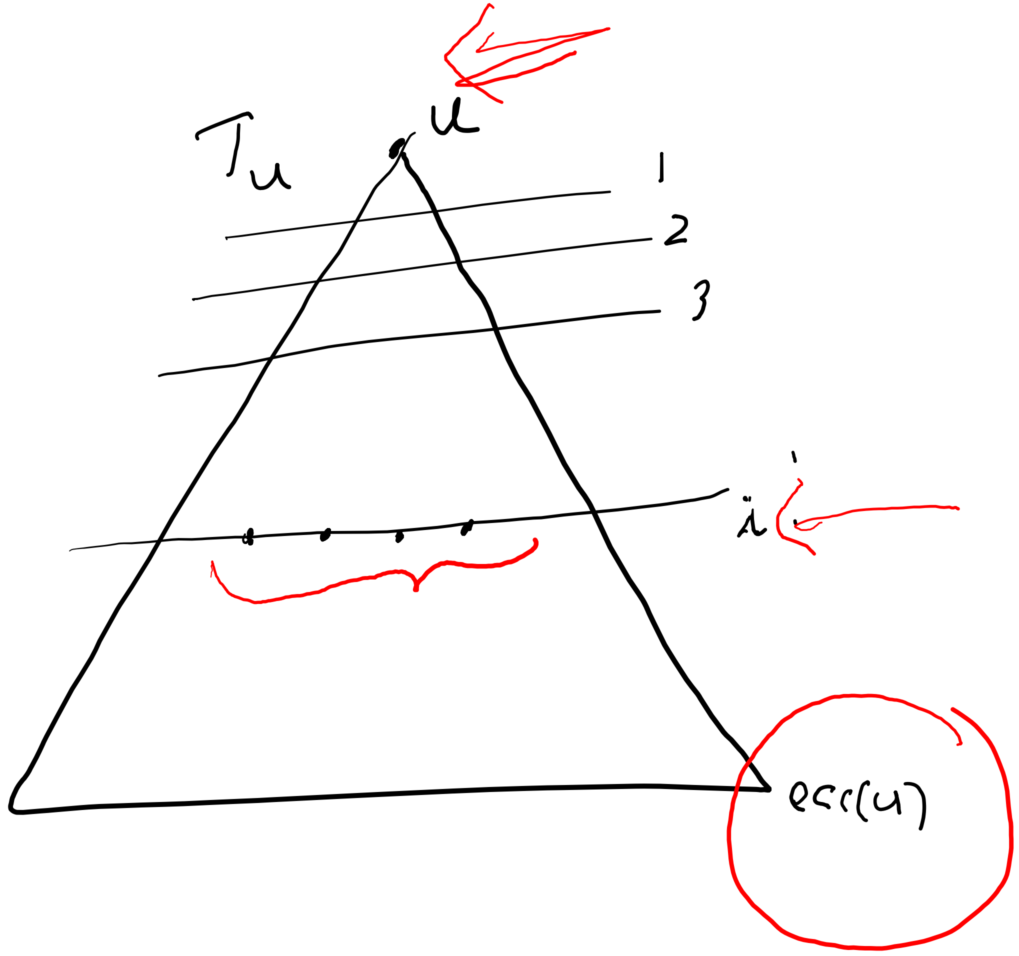
Diameter of a Graph

- It appears that there may be no better algorithm.
- However, a key question is to see if all the BFS are really required.
- A key observation by Crescenzi et al. helps improve the algorithm.



Diameter of a Graph

- Let us do a BFS from a node u in G.
- Let T_u be the BFS tree obtained.
- T_u has nodes in levels 0, 1, 2, etc. with node u in level 0, the neighbors of u in level 1, and the farthest nodes from u in level ecc(u).
- Let $F_i(u)$ be the nodes at distance i from u.
- Let $B_i(u) = \max_{z \in F_i(u)} ecc(z)$
- We now show the following claims.

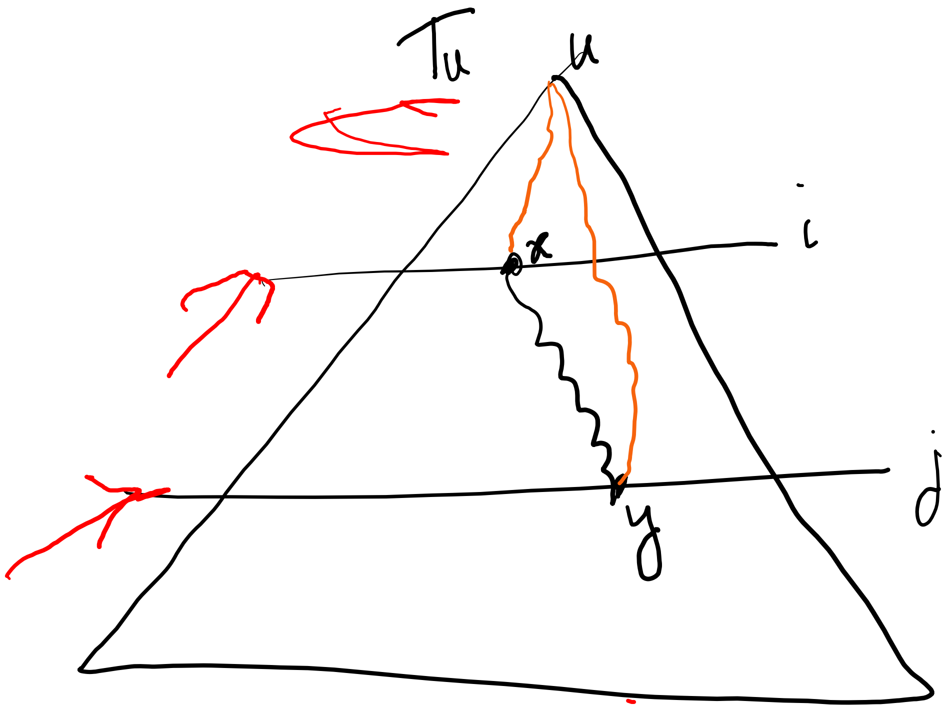


Diameter of a Graph

- Claim 1: Let x and y be any two nodes in G with at least one of them in $F_i(u)$. Then, $d(x, y) \leq B_i(u)$.
- Proof: Notice that $d(x, y) \leq \min \{ecc(x), ecc(y)\}$ by definition of ecc(.).
- Suppose that x is in $F_i(u)$. Then, $ecc(x) \leq B_i(u)$. Same holds if y is also in $F_i(u)$ ending the proof.
- If y is not in $F_i(u)$, then x must be in $F_i(u)$ allowing us to conclude that $d(x,y) \leq ecc(x) \leq B_i(u)$.

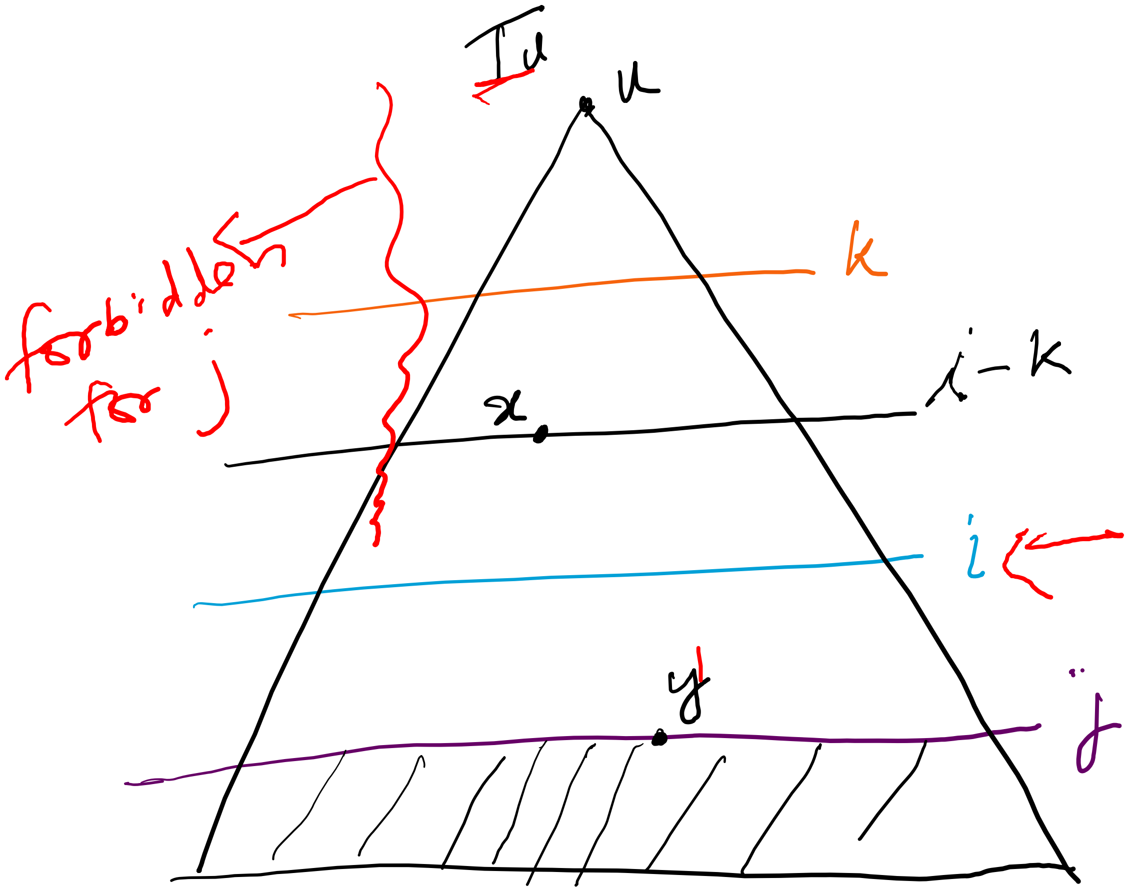
Diameter of a Graph

- Claim 2: Let x be in $F_i(u)$ and y be in $F_j(u)$. Then, $d(x, y) \le i + j \le 2 \max\{i, j\}$.
- Proof: Use the triangle inequality of distances.
- We note that $d(x,y) \le d(x,u) + d(u, y) = i + j \le 2 \max \{ i, j\}$.



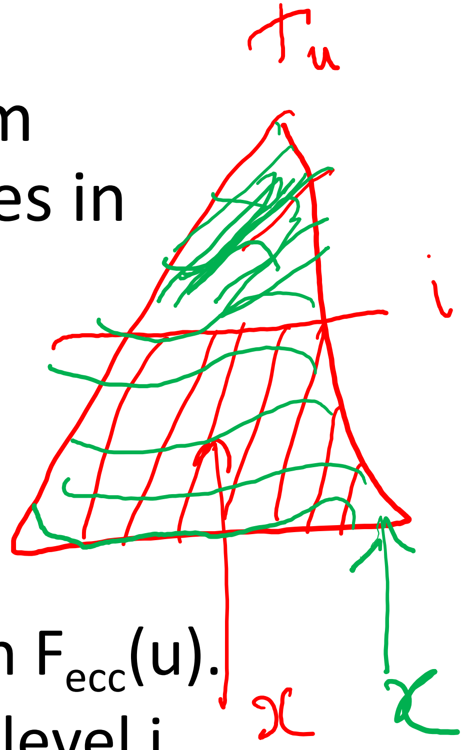
Diameter of a Graph

- Theorem 1: For any $1 \le i < ecc(u)$ and $1 \le k < i$, and for any $x \in F_{i-k}(u)$ such that $ecc(x) > 2(i-1)$, there exists $y_x \in F_j(u)$ such that $d(x, y_x) = ecc(x)$ with $j \ge i$.
- Proof: Given an x, there always exists a y_x such that $d(x, y_x) = ecc(x) > 2(i-1)$ by the hypothesis.
- Suppose that y_x is in $F_j(u)$ with $j < i$.
- Then, using Claim 2, $d(x, y_x) \le 2(i-1)$, which is a contradiction.
  - In this case, $\max\{i, j\} = i$.
- Hence, y_x in $F_j(u)$ with $j > i$.



Diameter of a Graph

- We can reinterpret Theorem 1 as follows.
- If x is a node in $F_i(u) \cup F_{i+1}(u) \cup \cdots \cup F_{ecc(u)}(u)$ with maximum eccentricity $ecc(x) > 2(i - 1)$, then the eccentricity of all nodes in $F_1(u) \cup F_2(u) \cup \cdots \cup F_{i-1}(u)$ is not greater than ecc(x).
- An algorithm can be derived from the above interpretation.
  - Pick a node u in G.
  - Perform a BFS in G from u and call the BFS tree as T_u.
  - Traverse T_u in a bottom-up fashion, starting from the nodes in $F_{ecc}(u)$.
  - At each level i, compute the eccentricities of all the nodes at level i.
    - If the maximum eccentricity e is greater than 2(i − 1) then we can discard traversing the remaining levels above level i.
      - The eccentricities of all such nodes cannot be greater than e.



Diameter of a Graph

- ALGORITHM 1: ifub
- Input: A graph G, a node u, a lower bound ℓ for the diameter, and an integer k
- Output: A value M such that $D - M \leq k$
- $i \leftarrow ecc(u)$;
- $lb \leftarrow \max\{ecc(u), \ell\}$; $ub \leftarrow 2ecc(u)$;
- while $ub - lb > k$ do
  - if $\max\{lb, B_i(u)\} > 2(i - 1)$ then return $\max\{lb, B_i(u)\}$;
  - else $lb \leftarrow \max\{lb, B_i(u)\}$; $ub \leftarrow 2(i - 1)$;
  - end
  - $i \leftarrow i - 1$;
- end
- return lb;

Diameter of a Graph

- What is the run time of the algorithm?
- In the worst case, it is possible that the while loop will go till $i = 0$.
- In that case, one has to do all n BFS's.
- So, the worst case run time is $O(n.m)$.
- What did we achieve then?

Diameter of a Graph

- In practice, the algorithm terminates much quickly.
- The authors study close to 200 graphs of a very large size and show that for several of them, only 0.1n BFS traversals are issued.
- A big improvement over the worst case run time.

Results

(a) Results obtained by ten executions of *i*FUB+4-SWEEP*r*, i.e. *i*FUB by using 4-SWEEP*r*.

Networks	Total	Number of networks having performance ratio v/n			
		$v/n \le 0.001$	$0.001 < v/n \le 0.01$	$0.01 < v/n \le 0.1$	$0.1 < v/n \le 1$
Autonomous systems graphs	2	2	0	0	0
Biological networks	48	4	22	22	0
Citations networks	5	4	0	1	0
Collaboration networks	13	4	7	2	0
Communication networks	38	26	7	4	1
Internet peer-to-peer networks	1	0	0	0	1
Meshes and electronic circuits	34	5	8	8	13
Product co-purchasing networks	4	4	0	0	0
Road networks	3	1	0	2	0
Social networks	11	9	0	2	0
Synthetic graphs	18	3	5	5	5
Web graphs	9	8	1	0	0
Words adjacency networks	4	1	3	0	0
Others	6	2	1	1	2
Total	196	73	54	47	22

Diameter of a Graph

- How to choose u?
- Several ways:
  - Random choice made uniformly across the n vertices.
  - 4-Sweep: Use four BFSs as follows.
    - Let r_1 be a node in V.
    - Let a_1 be one of the farthest nodes from r_1
    - Let b_1 be one of the farthest nodes from a_1.
    - If r_2 is the node halfway between a_1 and b_1, then we define analogously a_2 and b_2.
    - Our node u is then defined as the middle node of the path between a_2 and b_2.
    - Choose r_1 uniformly at random or as a node with the highest degree.

Diameter of the Graph

- This algorithm of Crescenzi et al. from the Theoretical Computer Science Journal in 2012 is a good example of algorithm engineering.
- We observe that theoretical claims lead to better algorithms but not in all cases.
- The paper has a couple of examples of graphs on which the proposed algorithm reaches a worst case time of O(n.m).