Computer Science 112
Data Structures

Lecture 26:
Review for Final Exam

Final Exam

- **Monday, May 11**
- **4 – 7 PM**
- **Our sections: Tillet 232**

Note on topsort

- **You are responsible for knowing about both breadth-first and depth-first topological sort, even though breadth-first has only been covered in recitation.**

BFS Topsort Algorithm

- **Keep a "predecessor count" for each vertex**
  - **Initially: in degree**
  - **When a predecessor is numbered, decrement count**

BFS Topsort Algorithm

- **enqueue sources (nodes with 0 in-degree)**
- **while not queue.isEmpty( )**

  v = queue.dequeue( )

  number v (increasing numbers)

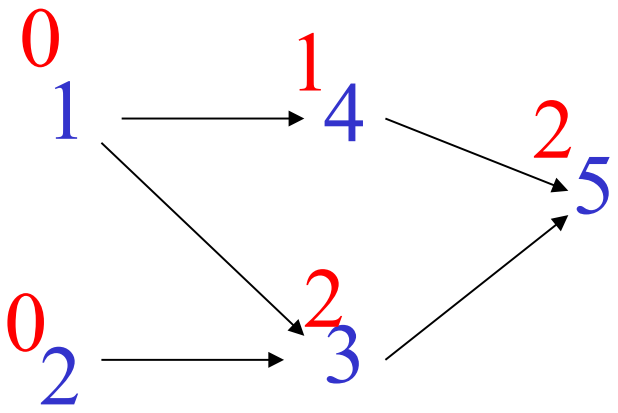
  decrement predecessor counts of v's neighbors

    if neighbor count becomes 0, enqueue neighbor

BFS Topsort Algorithm

- **Keep predecessor count for each vertex**
- **Enqueue vertices with count = 0**

queue
← front
2 1

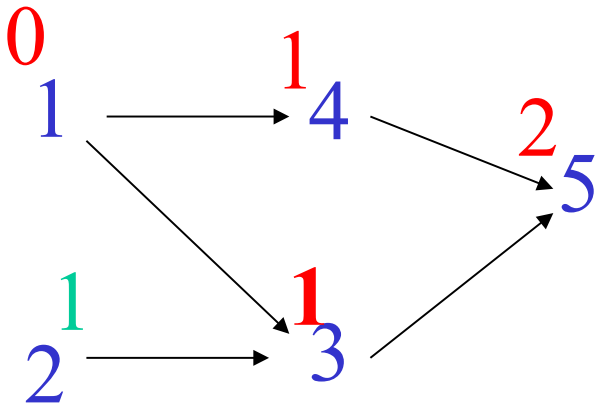


BFS Topsort Algorithm

- **Dequeue vertex 2**
  - **number it**
  - **decrement counts of neighbors – if now 0 enqueue**

queue
← front
1

v=2

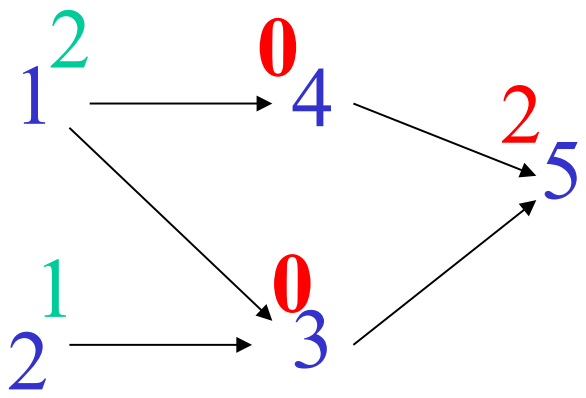


BFS Topsort Algorithm

- **Dequeue vertex 1**
  - **number it**
  - **decrement counts of neighbors – if now 0 enqueue**

queue
← front
4 3

v=1

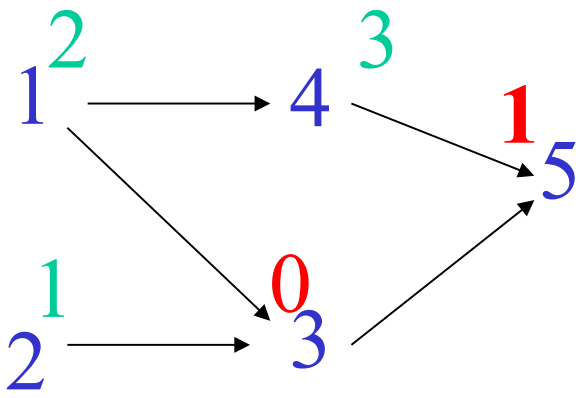


BFS Topsort Algorithm

- **Dequeue vertex 4**
  - **number it**
  - **decrement counts of neighbors – if now 0 enqueue**

queue
← front
3

v=4

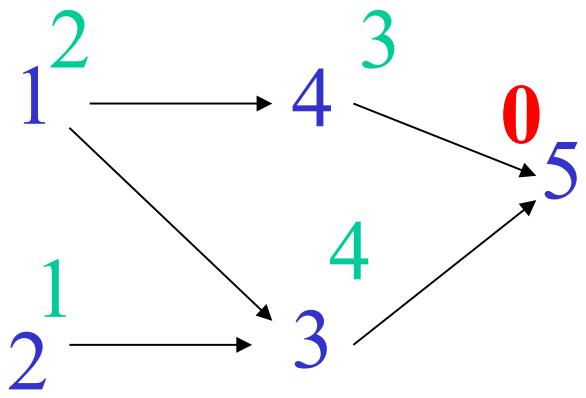


BFS Topsort Algorithm

- **Dequeue vertex 3**
  - **number it**
  - **decrement counts of neighbors – if now 0 enqueue**

queue
← front
5

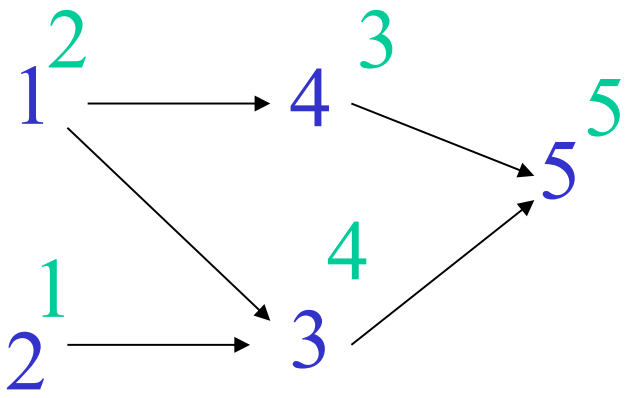
v=3



BFS Topsort Algorithm

- **Dequeue vertex 5**
  - **number it**
  - **decrement counts of neighbors – if now 0 enqueue**

queue
← front



v=5

Topics for Final Exam

Everything from exams 1 and 2

Heaps

Graphs

Sorting

Topics for Final Exam
since Exam 2

heaps

- **heap structure and order, implementation as an array**
- **heap insert / sift up and delete / sift down**

graphs

- **directed/undirected, weighted, path, cycle, connected**
- **representation as adjacency matrix and adjacency list**
- **Depth First Search, Breadth First Search**
- **DFS / BFS Topsort**
- **Dijkstsra's shortest path algorithm**

Sorting

- **quicksort, heapsort, (treesort), [merge, insertion sorts]**

Topics for exam 2

Binary search trees

AVL trees

Huffman codes

Hashing

Binary search trees

Ordering, Search, Insertion, Deletion,

Depth as a function of number of nodes

AVL trees

Balance factor, Rotation operation, Insertion and rebalancing [NOT deletion], Big-O

Huffman codes

Varying length codes, Huffman trees,

Decoding, Encoding, Building the tree

Hashing

Insertion, Chaining, Searching,

Load factor and rehashing, Big-O:

search and insert, worst and expected

Topics for Exam 1

- **Linked Lists**
- **Exceptions**
- **Generics**
- **Stacks**
- **Queues**
- **Search**

Linked lists

- add-front, delete-front, add-after, delete-after, length, last, etc
- circular linked lists, doubly linked lists

Exceptions

Generics

Stacks, queues

- as linked lists, as ArrayLists
- big-O

Search

- sequential, best/worst/average big-O
- binary