

## Announcement

- **Exam II** , Nov 17th, Tuesday
- **Review** , Nov 15th, 7pm, Cupples II, Room 100
- **Hw4** will be graded by Monday, pick up at office hours of TAS or in class on Tue
- **Crib sheet** : from last exam (3x5in) plus a new (8.5x11), one side of the new is the B-tree delete diagram

## Graphs

- Read Ch 23.4
- Topological sort
- Application example: make

## Topological Sort

- The *input* to the problem is a directed graph, the *goal* is to produce *an* ordered list of vertices such that if  $E$  contains the edge  $(u, v)$ , then  $u$  appears before  $v$  in the list. We call such list/ordering/scheduling *valid*. If no such ordering exist, report so.

The valid list, may not be unique.

*The topological sort problem has a solution iff the graph is acyclic.*

- **Input** digraph  $G = (V, E)$

- **Output**

- If the graph has a cycle, report “no valid ordering”.
- If the graph is acyclic, output a valid scheduling.

- **Example**

		Adj list:			
a	b	a:	b	d	
		b:	c	d	
		c:	d		
		d:			
c	d	a	b	c	d

■ *A digraph is acyclic iff DFS of  $G$  yields no back edges.*

## ■ Topological sort algorithm

1. Run DFS to get finishing times (also check for cycles)
2. if there are no cycles, *a valid schedule is given by using decreasing order of finishing times.* The schedule can be implemented as a list. Each time a vertex is finished it is inserted at the front of the list.

## ■ Time Complexity

- Step 1: has the time complexity of DFS:  $O(n + m)$ .
- Step 2: linear in the number of vertices:  $O(n)$ .
- Step 1 dominates, overall time:  $O(n + m)$ .

