CS1020 Data Structures and Algorithms I
Lecture Note #17

Mix-and-Match

Data Structures with
Multiple Organisation

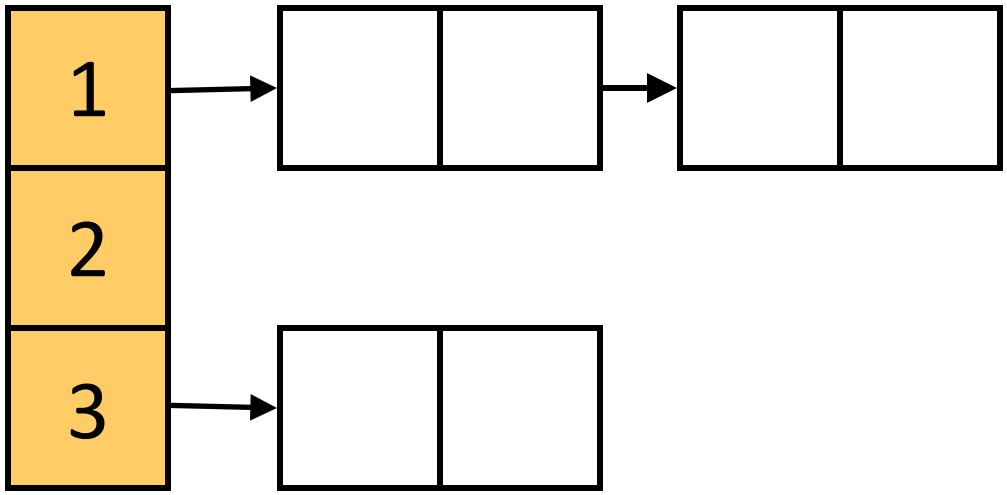
Basic Data Structures

- Arrays
- Linked Lists
- Trees (to be covered in CS2010)

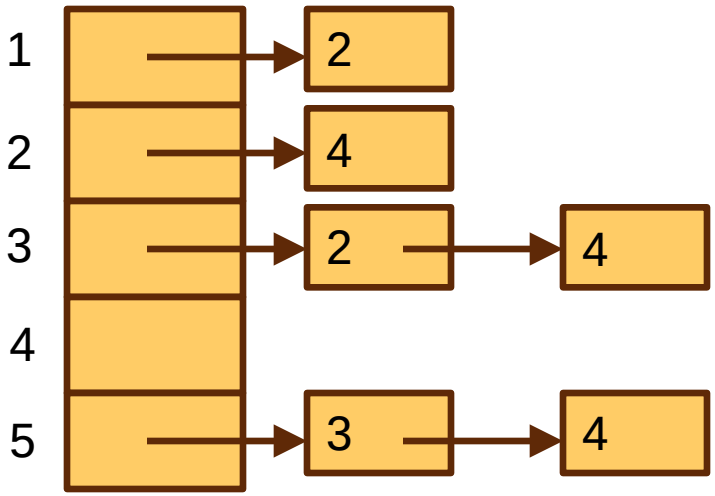
We can combine them to implement different data structures for different applications.

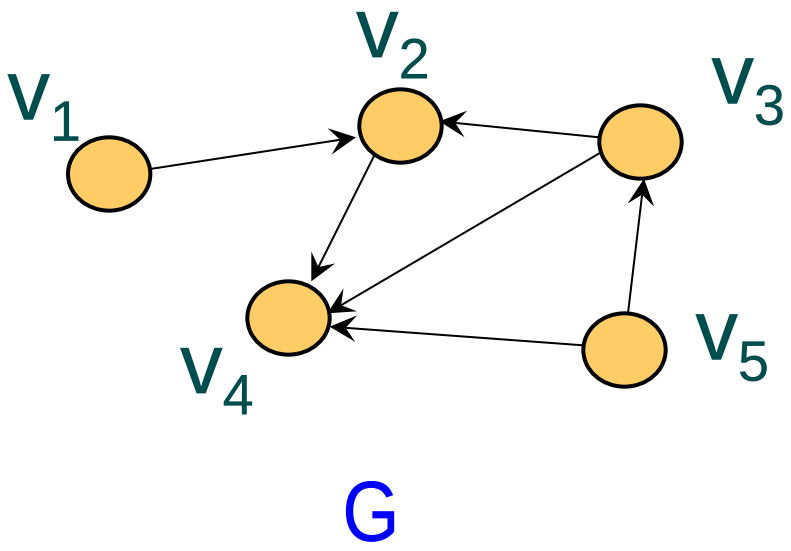
Mix-and-Match

- Array of Linked Lists
  - E.g.: Adjacent list for representing graph
  - E.g.: Hash table with separate chaining



Adjacency List for Directed Graph



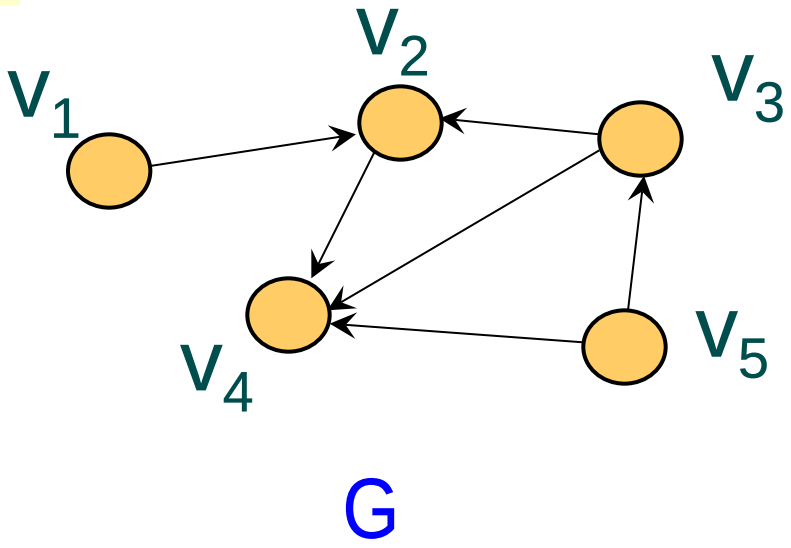


G

Adjacency Matrix for Directed Graph

Matrix[i][j] = 1 if $(v_i, v_j) \in E$
0 if $(v_i, v_j) \notin E$

		1 (v_1)	2 (v_2)	3 (v_3)	4 (v_4)	5 (v_5)
1	v_1	0	1	0	0	0
2	v_2	0	0	0	1	0
3	v_3	0	1	0	1	0
4	v_4	0	0	0	0	0
5	v_5	0	0	1	1	0



CS1102 AY2003

17. (16 points) Let n_i be the number of vertices adjacent to a vertex i. Suppose we want to support the following four operations on a directed graph:

- insert(i, j), which adds an edge (i, j) into the graph;
- delete(i, j), which removes the edge (i, j) from the graph;
- exists(i, j), which checks if edge (i, j) exists in the graph; and
- neighbours(i), which returns the list of vertices adjacent to i.

Describe a data structure that supports insert(i, j), delete(i, j) and exists(i, j) in $O(1)$ time, and neighbours(i) in $O(n_i)$ time. You may use diagrams to illustrate your data structure. You may simply quote data structures taught in this class without going into details.

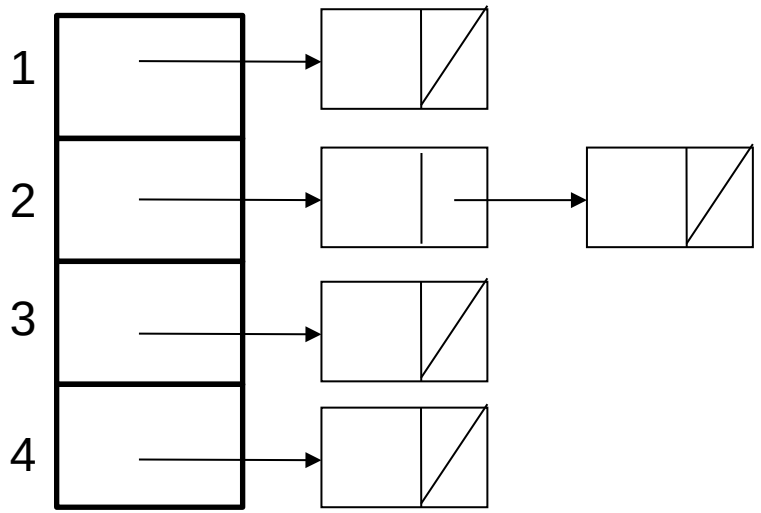
CS1102 AY2003: Use Adjacency Matrix

Operation	Big-O
insert(i, j)	O(1)
delete(i, j)	O(1)
exists(i, j)	O(1)
neighbours(i)	O(n)

	1	2	3	4
1		T		
2	T		T	
3		T		
4	T			

CS1102 AY2003: Use Adjacency List

Operation	Big-O
insert(i, j)	O(1)
delete(i, j)	O(n)
exists(i, j)	O(n)
neighbours(i)	$O(n_i)$



Problem

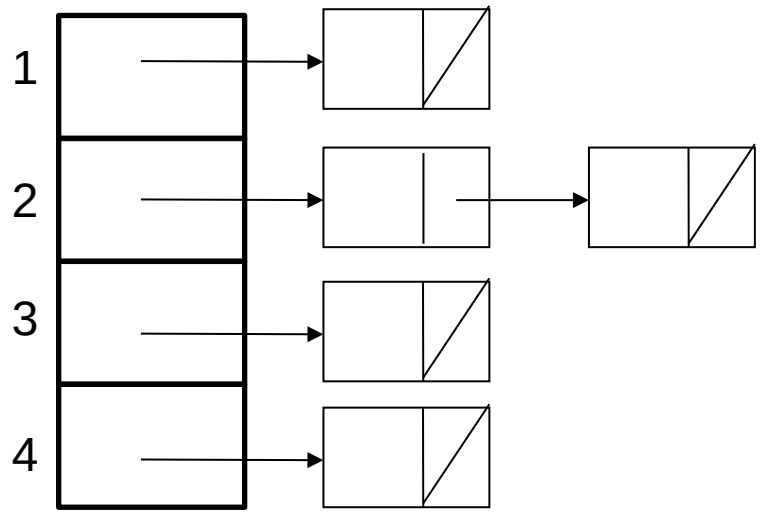
- Searching on an unsorted linked list is O(n)
- How to improve it to O(1)?

Use hashing.

(i, j) as key and the hash value returned by hash function to be index to a hash table where (i, j) is stored together with the reference to the node in the linked list.

Use Adjacency List

Operation	Big-O
insert(i, j)	O(1)
delete(i, j)	O(n)
exists(i, j)	O(1)
neighbours(i)	$O(n_i)$



Is delete (i, j) O(1)?

CS1102 AY2003

17. (16 points) Let n_i be the number of vertices adjacent to a vertex i. Suppose we want to support the following four operations on a directed graph:

- insert(i, j), which adds an edge (i, j) into the graph;
- delete(i, j), which removes the edge (i, j) from the graph;
- exists(i, j), which checks if edge (i, j) exists in the graph; and
- neighbours(i), which returns the list of vertices adjacent to i.

Describe a data structure that supports insert(i, j), delete(i, j) and exists(i, j) in $O(1)$ time, and neighbours(i) in $O(n_i)$ time. You may use diagrams to illustrate your data structure. You may simply quote data structures taught in this class without going into details.

Build an adjacency list of the graph, where the lists are doubly linked.

Build a hash table with (i, j) as key, and a reference to the node representing (i, j) in the adjacency list as value.

End of file