SC1007
Data Structures and Algorithms

Week 8: Hash Table &
Graph Representation

Instructor: Luu Anh Tuan

Email: anhtuan.luu@ntu.edu.sg

Office: #N4-02b-66

College of Engineering

School of Computer Science and Engineering

Course schedule

Week	Lecture Topic	Tutorial	Lab	Assignment Deadline
1	Introduction To Data Structure and Algorithm			
2	Linked List (LL) - Linear Search			
3	Analysis of Algorithm	T1	L1 - LL	
4	Stack and Queue (SQ) - Arithmetic Expression			AS1: LL
5	Tree Traversal - Binary Search	T2	L2 - SQ	AS2: SQ
6	AVL, Huffman coding		L3 - Tree 1	AS3: Tree
7	Revision	T3	L4 - Tree 2	AS4: Tree 2
	Recess Week – **Lab Test 1 – 3 March 2022 (Thu)**			
8	Hash Table + Graph Representation			
9	BFS, DFS		L5 - Hash Table + Graph	
10	Backtracking	T4	L6 - BFS, DFS	AS5 : Hash Table
11	Dynamic Programming	T5	L7 - Backtracking	AS6 : Graph + BFS, DFS
12	Bipartite Graph - Matching Problem	T6	L8 - DP	AS7 : Backtracking
13	Revision			AS8 : DP
14	**Lab2 Test + Quiz – 21 April 2022 (Thu)**			

Overview

- Hash Table
- Graph Terminology
- Graph Representation
  - Adjacency Matrix
  - Adjacency List

Hashing

- A typical space and time trade-off in algorithm
- To achieve search time in O(1), memory usage will be increased

What is hashing?

Direct-Address Table

- Assume that the keys of elements K drawn from the universe of possible keys U
- No two elements have the same key

- Search time is O(1) but …
  - The array size is enormous
  - |U| >> |K|

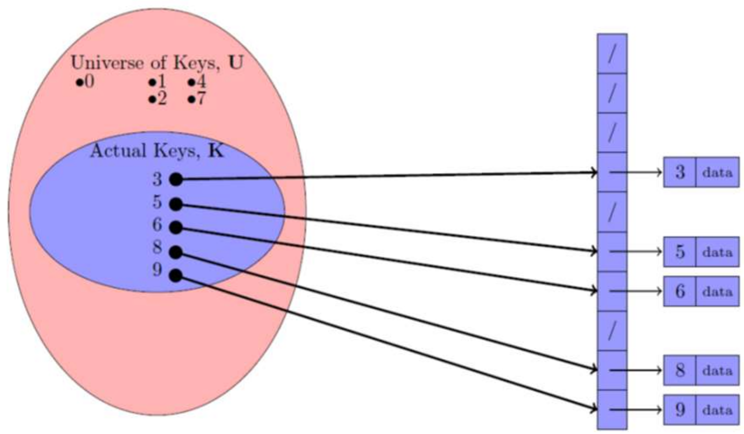


Figure 4.1: Direct Address Table

What is hashing?

- To reduce the key space to a reasonable size
- Each key is mapped to a unique index (hash value/code/address)
- Search time remains O(1) on the average

hash function: {all possible keys} → {0, 1, 2, …, h-1}

- The array is called a hash table
- Each entry in the hash table is called a hash slot
- When multiple keys are mapped to the same hash value, a collision occurs
- If there are n records stored in a hash table with h slots, its load factor is $\alpha = \frac{n}{h}$

Hash Functions

- Must map all possible value within the range of the hash table uniquely
- Mapping should achieve an even distribution of the keys
- Easy and fast to compute
- Minimize collision

1. Modulo Arithmetic
2. Folding
3. Mid-square
4. Multiplicative Congruential Method
5. Etc.

Hash Functions

1. Modulo Arithmetic: $\mathrm{H}(k) = k \bmod h$
   - E.g. h *= 17 & k = 37699* → *H(k) = 37699 mod 13 = 12*
   - In practice, h should be a prime number, but not too close to any power of 2

2. Folding
   - Partition the key into several parts and combine the parts in a convenient way
   - Shift folding: Divide the key into a few parts and added up these parts
   - X = abc → H(X) = (a + b + c) mod h
   - E.g. *H(123456789) =(123 + 456 + 789) mod 13 = 3*

Hash Functions

3. Mid-square
   - The key is squared and the middle part of the result is used as the hash address
     - E.g. $k=3121$, $k^2 = 3121^2 = 9740641 \rightarrow H(k) = 406$

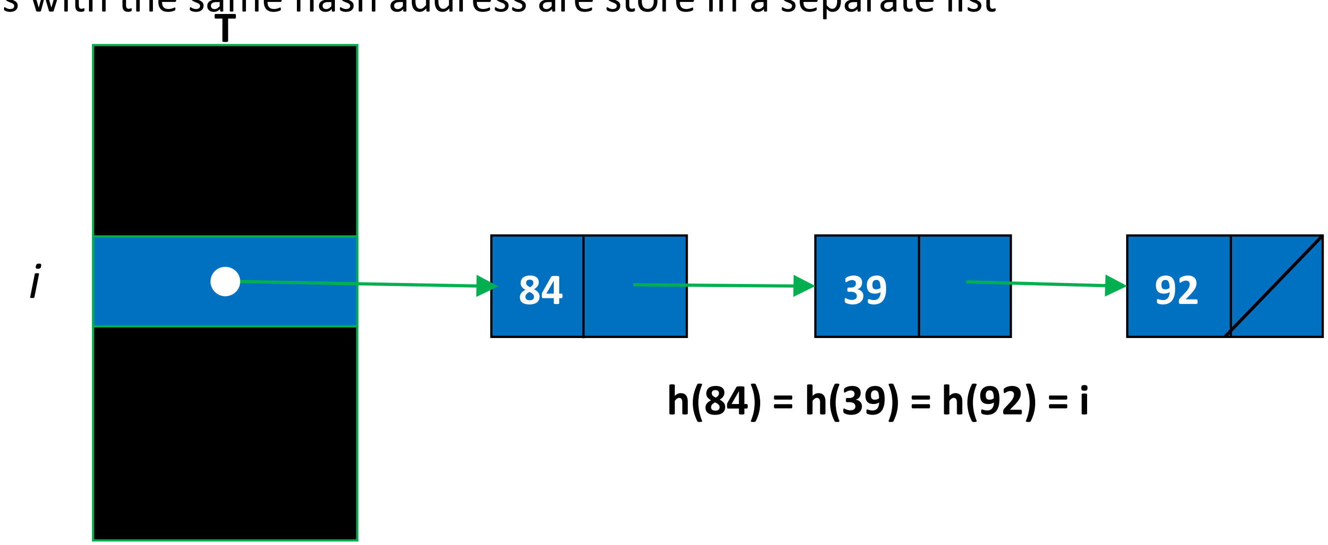
4. Multiplicative Congruential Method
   - Pseudo-random number generator
     - $H(k) = (a \times k)\ mod\ h$
     - E.g. $k = 5$, $a = 13 \rightarrow H(k) = (5 \times 6)\ mod\ 13 = 4$

Collision Resolutions

- Closed Addressing Hashing – a.k.a separate chaining
- Open Addressing Hashing
  - Linear Probing
  - Quadratic Probing
  - Double Probing

Closed Addressing: Separate Chaining

- Keys are not stored in the table itself
- All the keys with the same hash address are store in a separate list



- During searching, the searched element with hash address i is compared with elements in linked list H[i] sequentially
- In closed address hashing, there will be α number of elements in each linked list on average.

Closed Addressing: Separate Chaining

Time complexity in the worst-case analysis:

- When all elements are hashed to the same slot
- A linked list contains all n elements
- Its unsuccessful search takes n key comparisons, $\Theta(n)$
- Its successful search, assuming the probability of searching for each item is $\frac{1}{n}$

  $\frac{1}{n}\sum_{i=1}^{n} i = \frac{n+1}{2} = \Theta(n)$

  - It is just like a sequential search

Closed Address Hashing: Separate Chaining

Time complexity in the average-case analysis:

- All elements are equally likely hashed into h slots.
- Its unsuccessful search takes $\frac{n}{h}$ key comparisons, $\Theta(\alpha)$
- Its successful search takes 1 more than the number of comparisons done when the sought after item was inserted into the hash table
  - Before the i-th item is inserted into the hash table, the average length of all lists is $\frac{i-1}{h}$
  - When the ith item is sought for, the no. of comparisons is $(1+\frac{i-1}{h})$
  - The average number of key comparisons over n items
  - If α is constant (n is proportional to h), then $\Theta(1)$
  - Searching takes constant time averagely

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{n}\left(1+\frac{i-1}{h}\right) &= \frac{1}{n}\left(\sum_{i=1}^{n}1+\frac{1}{h}\sum_{i=1}^{n}(i-1)\right)\\ &= 1+\frac{1}{nh}\left(\frac{n}{2}(n-1)\right)\\ &= 1+\frac{n-1}{2h}=\Theta(1+\alpha)\end{aligned}$$

Open Addressing

- Keys are stored in the table itself
- α cannot be greater than 1
- When collision occurs, probe is required for the alternate slot
  - Ideally, the probing approach can visit every possible slot.

1. Linear Probing: probe the next slot

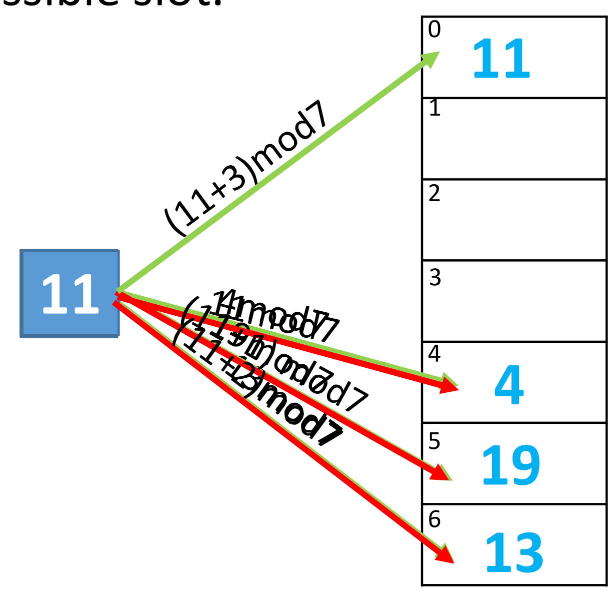
$H(k,i) = (k+i) \bmod h$ where $i \in [0, h-1]$

eg. $H(k,i) = (k+i) \bmod 7$

$k \in \{4, 13, 19, 11\}$

Primary clustering:

- A long runs of occupied slots
- Average search time is increased



Open Addressing

2. Quadratic Probing

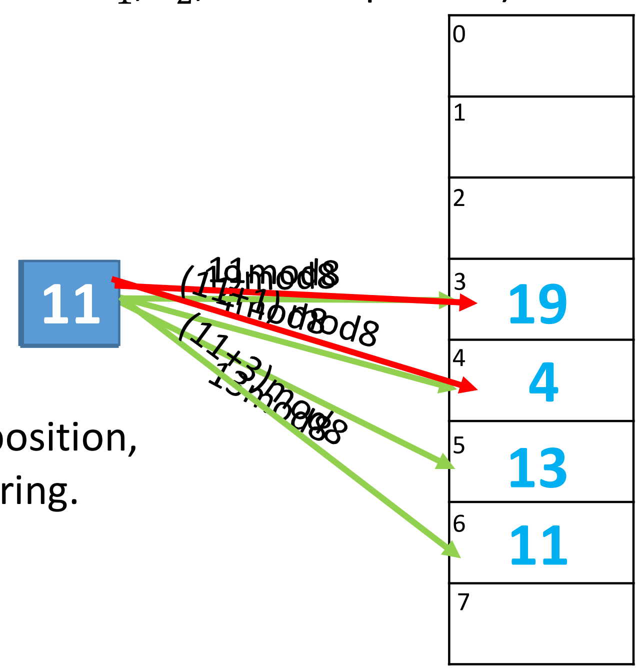
$$H(k,i) = (k + c_1 i + c_2 i^2) \bmod h \quad \text{where } c_1 \text{ and } c_2 \text{ are constants, } c_2 \neq 0$$

- May not all hash table slots be on the probe sequence (selection of c_1, c_2, h are important)
  - Any prime number h choice will only have at most h/2 distinct probes
- For $h = 2^n$, a good choice for the constants are $c_1 = c_2 = \frac{1}{2}$

eg. $H(k,i) = \left(k + \frac{1}{2}i + \frac{1}{2}i^2\right) \bmod 8$

$k \in \{4, 13, 19, 11\}$

$(\frac{1}{2}i + \frac{1}{2}i^2)$mod 8
1
3
6
2
7
5
4



- Secondary Clustering: if two keys have the same initial probe position, their probe sequences will be the same. This will form a clustering.
  - Inserting k=3 in the previous example.

Open Addressing

3. Double Hashing: a random probing method

$H(k,i) = (k + iD(k)) \bmod h$ where $i \in [0, h-1]$ and $D(k)$ is another hash function

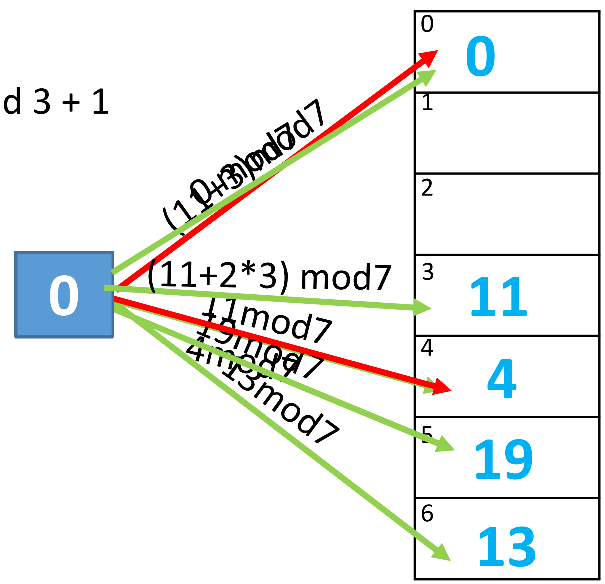
- The hash table size h should be a prime number

eg. $H(k,i) = (k + iD(k)) \bmod 7$

$\mathrm{D}(k) = (k) \bmod 3 + 1$

$k \in \{0, 4, 13, 19, 11\}$

D(11) = 11 mod 3 + 1



Time Complexity

Linear Probing

- Successful Search: $\frac{1}{2}(1 + \frac{1}{1-\alpha})$
- Unsuccessful Search: $\frac{1}{2}(1 + \left(\frac{1}{1-\alpha}\right)^2)$

Double Hashing

- Successful Search: $\frac{1}{\alpha}\ln\frac{1}{1-\alpha}$
- Unsuccessful Search: $\frac{1}{1-\alpha}$

*Proof can be found in The Art of Computer Programming by Knuth Donald (1973)

Delete A Key Under Open Addressing

- Leave the deleted key in the table
- Make a marker indicating that it is deleted
- Overwrite it when a new key is inserted to the slot
- May need to do a “garbage collection” when a large number of deletions are done
    - To improve the search time

Rehashing: Expanding the Hash Table

- As α increases, the time complexity also increases

Solution:

- Increase the size of hash table (doubled)
- Rehash all keys into new larger hash table

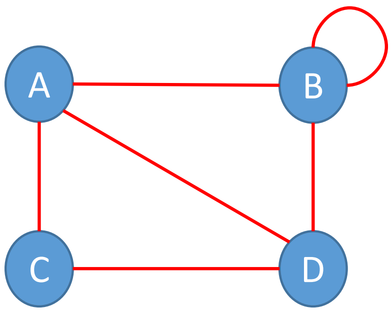
Hash Table: Summary

- Closed-address Hashing : Separate Chaining: $O(\alpha)$ on average
- Open-address Hashing: $O(f(\frac{1}{1-\alpha}))$
    - Linear Probing
    - Quadratic Probing
    - Double Hashing
- Delete keys
- Rehashing

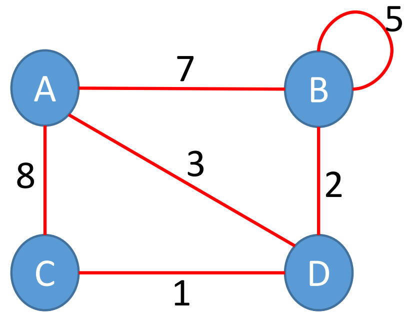
Graph Terminology

- A graph $G = (V,\ E)$ consists of two finite sets:
  - A set V of vertices/ nodes
  - |V| is the number of vertices
  - A set E of edges/arcs/links that connect the vertices
  - $E = \{(x, y)|x, y \in V\}$
  - $|\ E\ |$ is the number of edges ranged from 0 to $\frac{|V|(|V|-1)}{2}$
  - Degree of a vertex is the number of edges incident to it
  - A tree is a special graph with no cycle

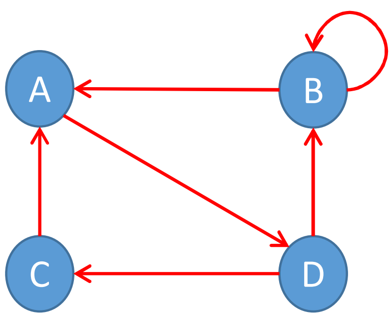
Graph Terminology



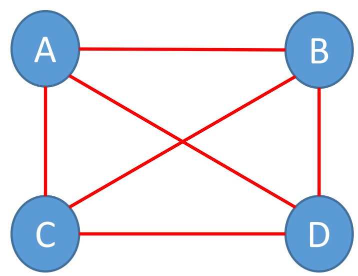
Undirected Graph



Weighted Graph



Directed Graph



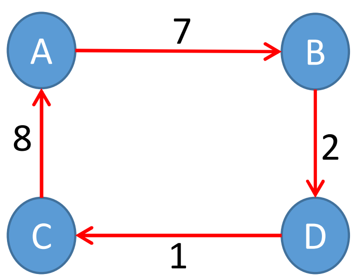
Complete Graph

Graph Terminology

- If e = (*x, y*) is an edge in an undirected graph, then *e* is **incident** with *x* and *y*; *x* is **adjacent** to *y* and vice versa.
- If E is unordered, then G is undirected; otherwise, G is a directed graph.
- If e = (x, y) is an edge in a directed graph, then y can be reached from x through one edge, so target y is adjacent to source x (but it doesn’t mean x is adjacent to y).

Graph Terminology

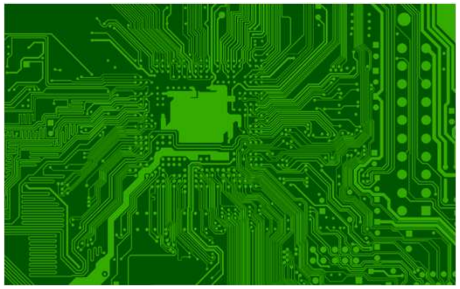
- A path is a sequence of distinct vertices, each adjacent to the predecessor (except for the first vertex). |V| = |E|+1
  - ABDC
- A cycle is a path containing at least three vertices such that the last vertex on the path is the same as the first. |V| = |E|
  - ABDCA



Graph Terminology

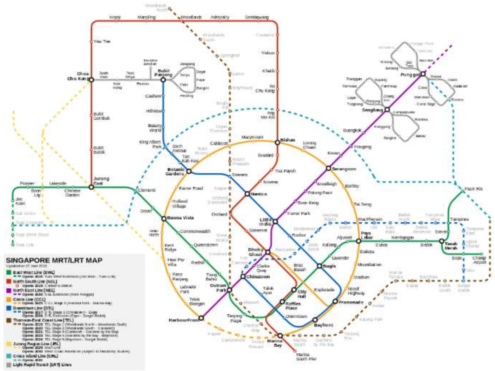
- An undirected graph is connected if there is a path from any vertex to any other vertex.
- A directed graph is strongly connected if there is a path from any vertex to any other vertex.
- A graph is cyclic if it contains one or more cycles; otherwise it is acyclic.
- A complete graph on n vertices is a simple undirected graph that contains exactly one edge between each pair of distinct vertices.
  - $|\mathrm{E}| = \frac{|V|(|V|-1)}{2}$

Graph Applications



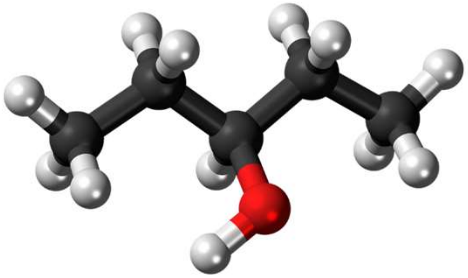
Electrical circuits

- V = {electrical devices}
- E = {linkage between devices}



Maps

- V = {stations}
- E = {underground route}



Organic Chemistry

- V = {atoms}
- E = {bonds between atoms}

Computer Networks

V = {computers}

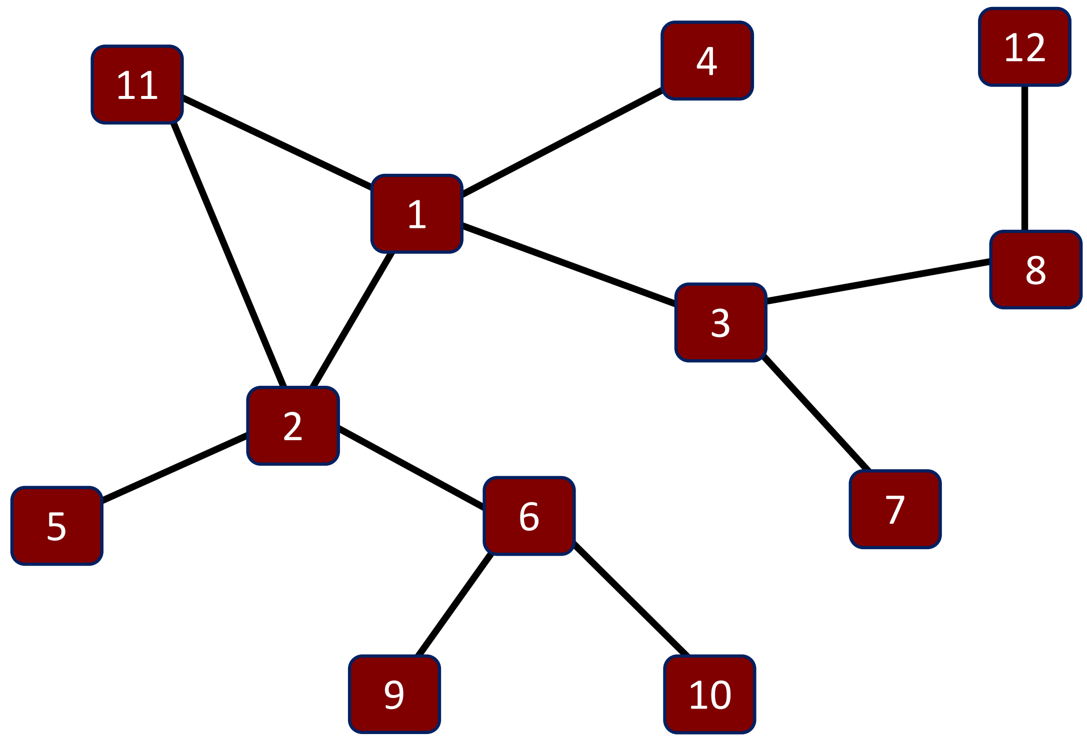
E = {connections between computers}

- Aforl. (2014). A map of Singapore's Mass Rapid Transit (MRT) and Light Rail Transit (LRT) systems [Image]. Retrieved from https://commons.wikimedia.org/wiki/File:Singapore_MRT_and_LRT_System_Map.svg
- File: Electric circuit [Image]. (2013). Retrieved from https://pixabay.com/en/board-chip-circuit-electric-158973
- Chemistry-atoms [Image]. (2015). Retrieved from https://pixabay.com/en/pentanol-molecule-chemistry-atoms-867210/

Graph Representation

- Adjacency Matrix

- Adjacency List



Adjacency Matrix

- Use a matrix (2-D array) with size $|V| \times |V|$

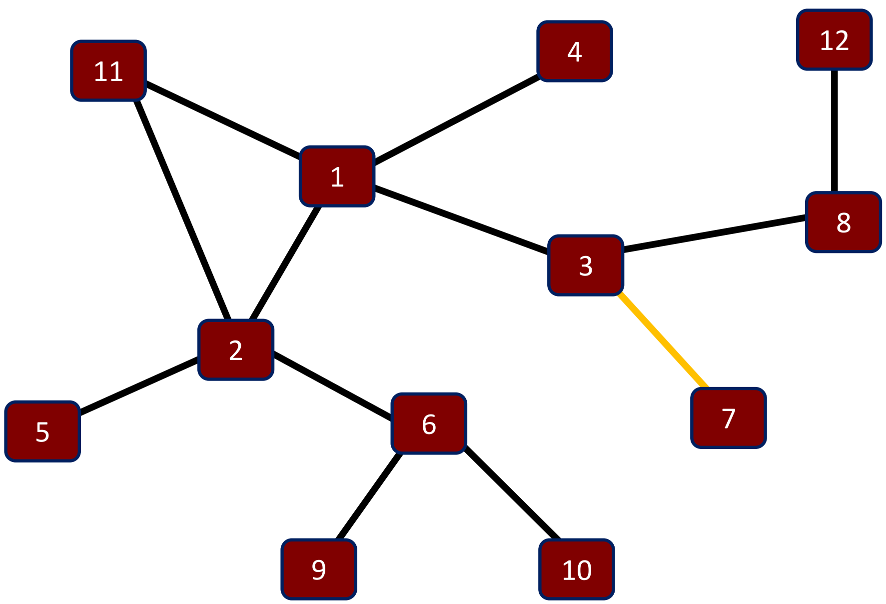
```
typedef struct _graph{
        int vSize;
        int eSize;
        int **AdjM;
}Graph;
```

- $(u, v) \in E$ implies AdjM[u][v] = 1; Otherwise AdjM[u][v] = 0.
- If a graph is undirected, then AdjM is symmetric
  - AdjM[u][v] = AdjM[v][u]
- If a graph is directed, then AdjM[u][v] = 1 iff $(u, v) \in E$ but it does not imply $(v, u) \in E$ and AdjM[v][u] = 1.

Adjacency Matrix

```
typedef struct _graph{
        int vSize;
        int eSize;
        int **AdjM;
}Graph;
```

- access time for AdjM[u][v] is constant
- when graph is sparsely connected, most of the entries in AdjM are zeros



	1	2	3	4	5	6	7	8	9	10	11	12
1	0	1	1	1	0	0	0	0	0	0	1	0
2	1	0	0	0	1	1	0	0	0	0	1	0
3	1	0	0	0	0	0	1	1	0	0	0	0
4	1	0	0	0	0	0	0	0	0	0	0	0
5	0	1	0	0	0	0	0	0	0	0	0	0
6	0	1	0	0	0	0	0	0	1	1	0	0
7	0	0	1	0	0	0	0	0	0	0	0	0
8	0	0	1	0	0	0	0	0	0	0	0	1
9	0	0	0	0	0	1	0	0	0	0	0	0
10	0	0	0	0	0	1	0	0	0	0	0	0
11	1	1	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	1	0	0	0	0

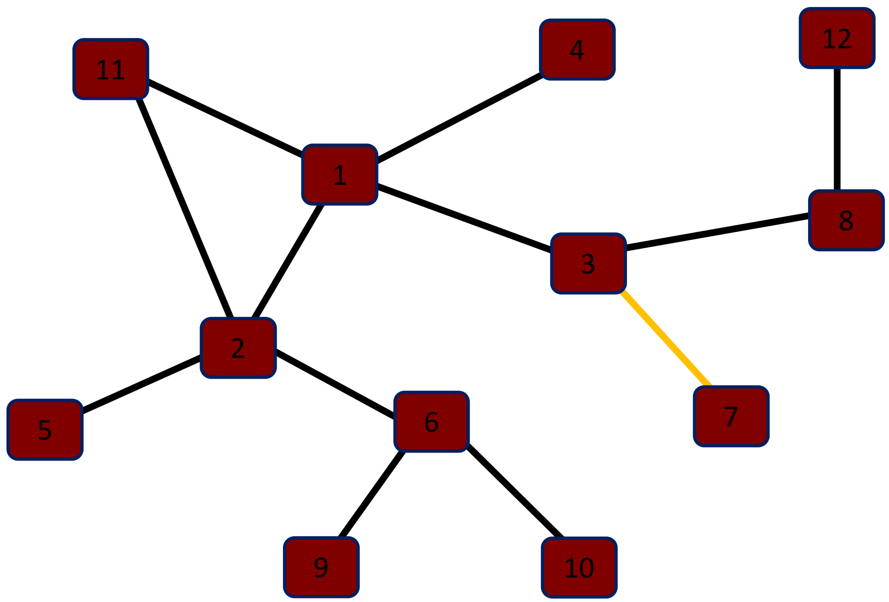
Adjacency List

```
struct _listnode
{
        int id; //or weight
        struct _listnode *next;
};
typedef struct _listnode ListNode;
typedef struct _graph{
       int vSize;
       int eSize;
       ListNode **AdjL;
}Graph;
```

- Use an array to represent the vertices
- For each vertex, use a linked list to represent the connections to other vertices
- Access time for AdjM[u][v] is linear
- Space complexity is lower, O(|V|+|E|)

Adjacency List

- Array size is |V|.
- Total number of nodes in link lists is 2|E|



1	→ 2 → 3 → 4 → 11
2	→ 11 → 1 → 5 → 6
3	→ 1 → 8 → 7
4	→ 1
5	→ 2
6	→ 10 → 9 → 2
7	→ 3
8	→ 12 → 3
9	→ 6
10	→ 6
11	→ 2 → 1
12	→ 8

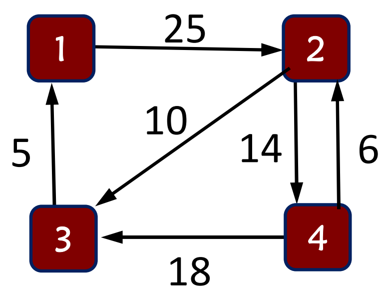
Represent Weighted Graphs

- In the array of adjacency lists, the weight can be stored as a data field in each list node
- In the adjacency matrices, the weight can be stored
  - The element at the *u*-th row and the v-th column can be defined as:

$$AdjM[u][v] = \begin{cases} W(u, v) & \text{if } (u, v) \in E \\ c & \text{otherwise} \end{cases}$$

- Constant c can be defined as 0 (weight as capacity) or some very large number ∞ (weight as cost)

Represent Weighted Graphs



	1	2	3	4
1	0	25	0	0
2	0	0	10	14
3	5	0	0	0
4	0	6	18	0

1 → (2, 25)

2 → (3, 10) → (4, 14)

3 → (1, 5)

4 → (2, 6) → (3, 18)

Summary

- Concepts and terminologies of graph, such as
  - A graph consists of a set of vertices and a set of edges
  - Directed vs. undirected graphs
  - The definitions of path and cycle, etc.
- Two data structures used to represent graphs:
  - Adjacency matrix
  - Array of adjacency lists
  - Their advantages and disadvantages for different applications


Q&A