Quick Revision

A flashback over what has been learnt with highlights on some important topics covered

Answers to the complete list of Q&As were already provided during the end of each lecture (and included within the video recordings of each and therefore not repeated herewith)

Menu

- Data structures
- Motivation of studying data structures
- Language to study data structures
- **Abstraction**
- **Huffman coding & priority queues**
- **Information hiding**
- **Encapsulation**
- **Efficiency in space & time**
- **Static vs dynamic data structures**

Q&A

- Name three typical type of data values in common data collections.
- Name three typical structures seen in common data collections.
- Name three typical operations seen in common data collections.

- Why do we learn data structure?
- Can we program data structure in Java?
- Can we program data structure in C?

Q&A

- Why do we insist an algorithm must terminate?
- Why do we insist an algorithm must be precise?
- Why instructions in an algorithm are written in a sequence?

- Write down an algorithm to start up IE Explorer on a computer.
  - Input: a computer equipped with Windows which is shut down
  - Output: a computer up & running with Windows IE Explorer

- Write down an algorithm to shutdown a computer safely from Windows.
  - Input: a computer equipped with Windows which is running under Windows
  - Output: a computer which is shutdown

Q&A

- Both space efficiency & time efficiency are metrics used to evaluate the performance of an algorithm (and a data structure). (T or F?)
- Dynamic data structures are more space efficient in general. (T or F?)
- Static data structures are more time efficient in general. (T or F?)
- Information hiding is the principle that users of a software component need to know only the essential details of how to *initialize* and *access* the component, and do not need to know the details of the implementation (T or F?)

Menu

- Overview of Data Structure Programming topics
- Programming with Libraries
- **Collections**
- **Programming with Lists of objects**

Q&A

- Name 3 type of collections that can be implemented under Java Programming with Linear collections
- Name 3 operations that can be implemented under Java Programming with Linear collections
- Name 2 type of collections that can be implemented under Java Programming with Hierarchical collections
- Name 4 operations that can be implemented under Java Programming with Hierarchical collections
- What is a software “library”?
- Define Java “Package”.
- Name Java’s IO library.
- Name Java’s GUI library.
- What is the Java statement to include package or class into your program?

Menu

- Collections and List
- **Using List and ArrayList**
- Iterators

Q&A: How does ArrayList work?

- What does it store inside?
- How does it keep track of the size?
- How does it grow when necessary?

Menu

- More Collections
- **Bags and Sets**
- **Stacks and applications**
- **Maps and applications**

Menu

- **Queues and Priority Queues**
- **Classes/Interfaces that accompany collections**
  - **Iterator**
  - **Iterable**

Q&A

- Java has specified a “Queue” interface. (T or F)
- Java does not have any class support for “Priority Queue”. (T or F)
- peek() operation under the Queue interface will throw an exception if the queue is empty. (T or F)
- poll() operation under the Queue interface will throw an exception if the queue is empty. (T or F)
- There is an element() method under the Queue interface. (T or F)
- Iterable is an interface specification for a class that is equipped with an Iterator.
- Iterator is an interface specification for a class that can generate iterative elements.

Menu

- **Iterators and Iterables**
- Sorting collections
- **Comparators and Comparables**

Q&A

- An object defined under a comparable class will have a “natural ordering”. (T or F)
- Objects declared under a comparable class can be compared using which method?
- What is the signature of the *compareTo* method?
- Which method can be used to sort list of comparable objects?
- Comparator is an object that can compare other objects. (T or F)
- What is the signature for the *compare()* method?
- A comparable class can implement multiple comparators. (T or F)

Menu

- Comparators
- Exceptions
- **Implementing Collections:**
  - **Interfaces, Classes**

Q&A

- A method will do what when something goes wrong?
- An exception provides a local exit when something goes wrong. (T or F)
- Name 3 cases when an exception can be thrown.
- What do we do with exceptions?
- Exceptions can be caught using what Java statement?
- Does exception have a type?
- After the exception handler finishes its work, what will the program do next?
- Can exception handler use information in the exception object?
- What does "**e.getMessage**()" do where **e** is an exception object?

Q&A

- Name 3 common Java exceptions.
- RuntimeExceptions doesn’t have to be handled. (T or F)
- IOException doesn’t have to be handled. (T or F)
- Which Java statement can cause an exception?
- What happens when we try adding an element to an immutable List?
- Does a Java Interface provide a ‘constructor’?
- ArrayList implements which Interface?
- **public interface** List <E> extends which Java Interface?

Menu

- Implementing Collections:
  - Interfaces, **Abstract Classes**, Classes

Q&A

- What are the key features of an abstract class?
- Can an abstract class be instantiated?
- Abstract methods can be defined within a class to save implementation efforts. (T or F)

- What are the key issues of implementation when we remove an element from an ArrayList?
- What are the key issues of implementation when we add an element from an ArrayList?

Menu

- **Implementing ArrayList**:
  - Iterators
  - **Cost of adding and removing**

Q&A

- remove() is compulsory in Iterator implementation. (T or F)
- How does ArrayList make use of the ‘type parameter’ in its implementation?
- Which element will be removed by ArrayList.remove()?
- What does ArrayList.next() check before returning the next element in the list?
- How does ArrayList.remove() ensure only 1 element can be removed after each call to next()?
- What can happen if 2 or more Iterators running concurrently under the same ArrayList? Name 2 scenarios.

Menu

- Cost of operations and measuring efficiency
- **ArrayList: remove, add**
- **ArraySet: contains, remove, add**

Q&A

- O(log(n)) < O(sqrt(n)) (T or F)
- O(n^n) < O(n!) (T or F)
- O(2^n) < O(n^n) (T or F)
- When analysing the cost of an algorithm, loop usually is the focus. (T or F)
- Which of the following operations is more expensive?
  - Reading a line from a file
  - Reading a line from a user
- Worst case cost analysis is usually more difficult than average cost analysis. (T or F)

Menu

- **recursive functions**
- factorial function
- fibonacci function
- recursion vs iteration

Q&A

- What is the key step in designing a recursive function?
- Every recursive function can be rewritten as an iterative function. (T or F)

Menu

- Testing collection implementations
- **Queues**
- Motivation for linked lists
- **Linked structures for implementing Collections**

Q&A

- When do you write test cases for “black box” testing? Before or after implementation?
- Explain why array implementations of queue are slow.
- Linked list allows data removal by?
- Define references/pointers.
- What is the purpose of garbage collection in memory management?

Menu

- **Linked structures for implementing Collections**
- A collection class
- Adding/Removing from the front

Menu

- **A Stack using a Linked List with a header**
- **A Queue using a Linked List with a header**

Menu

- **Cost of ArraySet operations**
- **Binary Search**
- **Cost of SortedArraySet with Binary Search**

Menu

- Binary Search
- Sorting
  - approaches
  - selection sort
  - insertion sort
  - bubble sort
  - analysis
  - fast sorts

Menu

- Sorting
  - **Design by Divide and Conquer**
  - **Merge Sort**
  - **QuickSort**

Menu

- Analysing Fast Sorting Algorithms
  - MergeSort
  - QuickSort

Menu

- Introduction to Trees
  - What are trees?
- **Binary Tree**
- **General Tree**
- Terminology
- Different Types of Tree
- **Tree Ordering**
- **Trees and Recursion**
- What are they useful for?
  - Tic Tac Toe example
  - Chess
  - Taxonomy Tree
  - Decision Tree

Q&A

- Tree represents an efficient 1-dimensional data structure. (T or F?)
- A leaf node in a tree has no children. (T or F?)
- Binary tree has no ordering upon its sibling nodes. (T or F?)
- Name 3 applications for tree.
- Relationship among recursive calls can be expressed in what type of tree?

Menu

- **Maps**
- Search lists
- **Binary search trees**
- **Tree traversal**
  - **Preorder**
  - **Inorder**
  - **Postorder**
- **Balanced Search Trees**
  - **AVL Trees**

Menu

- **Implementing Binary Trees**
- **Implementing General Trees**

Menu

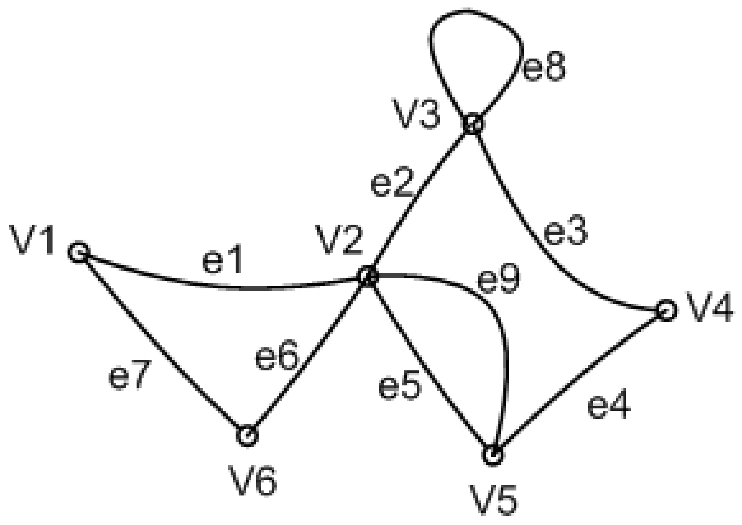
- **Hash tables**
- **Comparison among various search mechanisms**
- Table size
- **Hash function**
- Modular hash function
- Hash function examples
- **Collisions**
- Applications

Menu

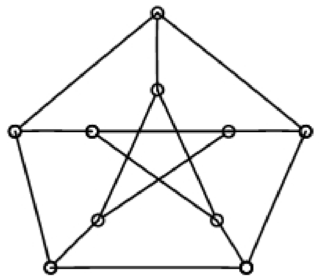
- Basic definitions of graph theory
- **Properties of graphs**
- **Paths**
- **Trees**
- Digraphs and their applications, network flows

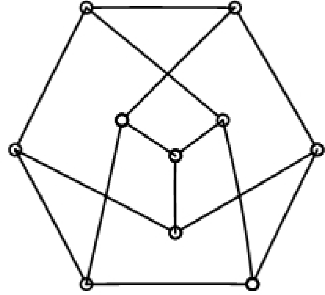
Self test

- 1. Write down the vertex set and edge set of the graph in:

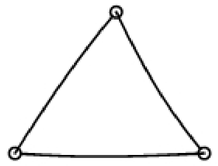


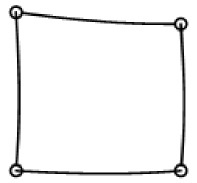
- The vertex set is {V1,V2,V3,V4,V5,V6},
- The edge set is {e1,e2,e3,e4,e5,e6,e7,e8, e9}.

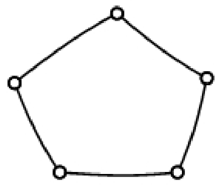

(a)

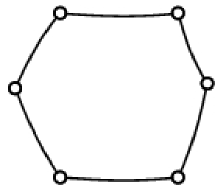

(b)

- 2. Which graphs below are subgraphs of those shown above.


(a)

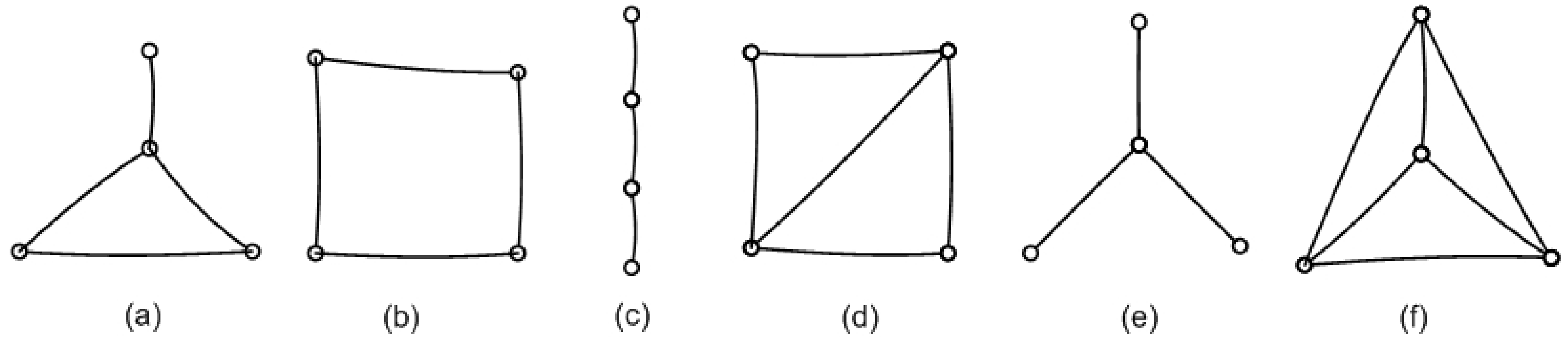

(b)


(c)


(d)

(c), (d)

- 3. Write down the degree sequence in the graphs below. Verify that the sum of the values of the degrees are equal to twice the number of edges in the graph.



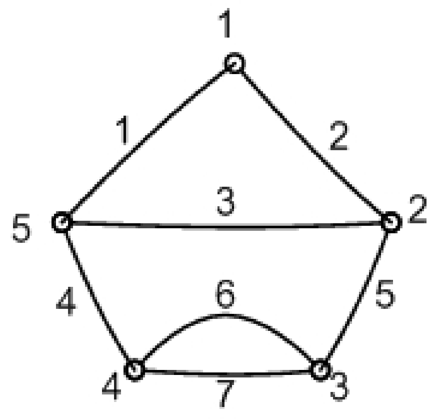
(a) (3, 2, 2, 1);
(b) (2, 2, 2, 2);
(c) (2, 2, 1, 1);

(d) (3, 3, 2, 2);
(e) (3, 1, 1, 1);
(f) (3, 3, 3, 3);

Menu

- Paths
- Connected graphs
- **Incidence matrix and adjacency matrix of a graph**

- **Self test**
- 1. Write down the adjacency and incidence matrices of the graph below.



$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 1 & 0 & 0 & 1 \\ 2 & 1 & 0 & 1 & 0 & 1 \\ 3 & 0 & 1 & 0 & 2 & 0 \\ 4 & 0 & 0 & 2 & 0 & 1 \\ 5 & 1 & 1 & 0 & 1 & 0 \end{array}$$

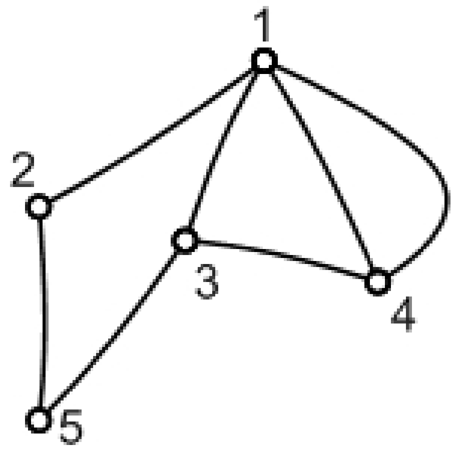
$$\begin{array}{c|ccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \hline 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 4 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 5 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \end{array}$$

- **Self test**
- 2. Draw the graph whose adjacency matrix is given in (a)

$$\begin{pmatrix} 0 & 1 & 1 & 2 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 2 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

(a)

$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 1 & 1 & 2 & 0 \\ 2 & 1 & 0 & 0 & 0 & 1 \\ 3 & 1 & 0 & 0 & 1 & 1 \\ 4 & 2 & 0 & 1 & 0 & 0 \\ 5 & 0 & 1 & 1 & 0 & 0 \end{array}$$

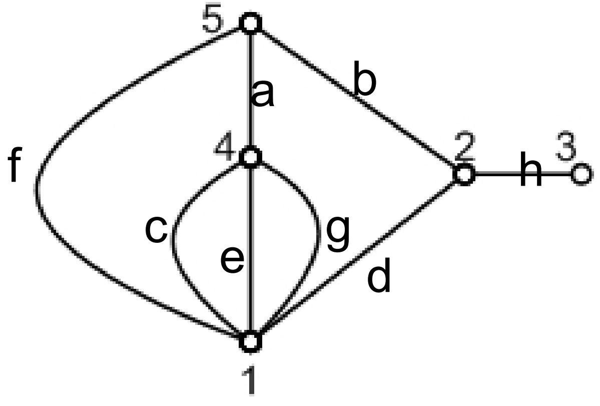


- **Self test**
- 3. Draw the graph whose incidence matrix is given in (b)

$$\begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

$$\begin{array}{c|cccccccc} & a & b & c & d & e & f & g & h \\ \hline 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 2 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 4 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 5 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \end{array}$$

(b)



Menu

- Trees and forests
- Spanning trees
- **Minimum spanning tree**
- **Greedy algorithm for determining a minimum spanning tree**
- **Shortest path problem**