ICOM 4035 – Data Structures

Dr. Manuel Rodríguez Martínez
Electrical and Computer Engineering Department

Lecture 8 – September 13, 2001

Agenda

- Project 1 is out: Algebra and Calculus on Polynomials.
  - Get the source code from class web page.
  - Due Date: 12:00 PM October 4
- Exam1 Date: 3:00PM-4:30PM, September 27, 2001
  - Topics: Arrays, structures, pointers, objects, operator overloading, constructor, copy constructors, destructors, copy assignment, vector class, bag class, set class, bitset class, Big-Oh notation.
  - Coming soon: practice problems & solutions.

Review of bags

- Trace the execution of the following operations on bag b:

```
bag b(3);
b.insert(10);
b.insert(-2);
b.insert(3);
b.find(10);
b.find(-1);
b.erase(20);
b.insert(3);
b.erase(-2);
```

Set Class Container

- Set is data structure used to store elements with the following semantics:
  - Only one copy of an element can be stored in the set S.
  - A find operation is supported to determine if an element x is present in the set S.
  - An erase operation is supported to erase an element x from the set S.
  - The following set operations are supported:
    - Union – elements that appear in at least one of two sets
    - Intersection – elements that are common to two sets
    - Difference – elements that appear in the first set but no in the second one.
    - Sub-set – determines is a set A is a subset of another set B.
- Sets are used in application in which copies are not allowed:
  - Clients in a video store, holydays in a month, social security #

On the Theory of Sets

- A set S is a collection of objects, where there are no duplicates
  - Examples
    - A = {a, b, c}
    - B = {0, 2, 4, 6, 8}
    - C = {Jose, Pedro, Ana, Luis}
- The objects that are part of a set S are called the elements of the set.
  - Notation:
    - 0 is an element of set B is written as $0 \in B$.
    - 3 is not an element of set B is written as $3 \notin B$.

Cardinality of Sets

- Sets might have
  - 0 elements – called the empty set ∅.
  - 1 elements – called a singleton
  - N elements – a set of N elements (called a finite set)
    - Ex: S = {car, plane, bike}
  - ∞ elements – an infinite number of elements (called infinite set)
    - Integers, Real,
    - Even numbers: E = {0, 2, 4, 6, 8, 10, …}
      - Dot notation means infinite number of elements
- The cardinality of a set is its number of elements
  - Notation: cardinality of S is denoted by |S|
  - Could be an integer number or infinity symbol ∞.

Cardinality of Sets (cont.)

- Some examples:
  - A = {a,b,c}
    - |A| = 3
  - R – set of real numbers
    - |R| = ∞
  - E = {0, 2, 3, 4, 6, 8, 10, …}
    - |E| = ∞
  - ∅ the empty set
    - | ∅ | = 0

Set notations and equality of Sets

- Enumeration of elements of set S
  - A = {a,b c}
  - E = {0, 2, 4, 6, 8, 10, …}
- Enumeration of the properties of the elements in S
  - E = {x : x is an even integer}
  - E = {x: x $\in$ I and x/2=0, where I is the set of integers.}
- Two sets are said to be equal if and if only they both have the same elements
  - A = {a, b, c}, B = {a, b, c}, then A = B
  - if C = {a, b, c, d}, then A $\neq$C
    - Because d $\notin$ A

Sets and Subsets

- Let A and B be two sets. B is said to be a subsets of A if and only if every member x of B is also a member of A
  - Notation: $B \subseteq A$
  - Examples:
    - A = {1, 2, 3, 4, 5, 6}, B = {1, 2}, then $B \subseteq A$
    - D = {a, e, i, o, u}, F = {a, e, i, o, u}, then $F \subseteq D$
  - If B is a subset of A, and $B \neq A$, then we call B a proper subset
    - Notation: $B \subset A$
    - A = {1, 2, 3, 4, 5, 6}, B = {1, 2}, then $B \subset A$
  - The empty set $\emptyset$ is a subset of every set, including itself
    - $\emptyset \subseteq A$, for every set A
  - If B is not a subset of A, then we write $B \not\subset A$

Set Union

- Let A and B be two sets. Then, the union of A and B, denoted by $A \cup B$ is the set of all elements x such that either $x \in A$ or $x \in B$.
  - $A \cup B = \{x: x \in A \text{ or } x \in B\}$
- Examples:
  - A = {10, 20 , 30, 40, 100}, B = {1,2 , 10, 20} then $A \cup B$ = {1, 2, 10, 20, 30, 40, 100}
  - C = {Tom, Bob, Pete}, then $C \cup \emptyset = C$
  - For every set A, $A \cup A = A$

Set Intersection

- Let A and B be two sets. Then, the intersection of A and B, denoted by $A \cap B$ is the set of all elements x such that $x \in A$ and $x \in B$.
  - $A \cap B = \{x: x \in A \text{ and } x \in B\}$
- Examples:
  - A = {10, 20 , 30, 40, 100}, B = {1,2 , 10, 20} then $A \cap B$ = {10, 20}
  - Y = {red, blue, green, black}, X = {black, white}, then $Y \cap X$ = {black}
  - E = {1, 2, 3}, M={a, b} then, $E \cap M = \emptyset$
  - C = {Tom, Bob, Pete}, then $C \cap \emptyset = \emptyset$
- For every set A, $A \cap A = A$
- Sets A and B disjoint if and only if $A \cap B = \emptyset$
  - They have nothing in common

Set Difference

- Let A and B be two sets. Then, the difference between A and B, denoted by A - B is the set of all elements x such that $x \in A$ and $x \notin B$.
  - $A - B = \{x: x \in A \text{ and } x \notin B\}$
- Examples:
  - A = {10, 20 , 30, 40, 100}, B = {1,2 , 10, 20} then A - B = {30, 40, 100}
  - Y = {red, blue, green, black}, X = {black, white}, then Y - X = {red, blue, green}
  - E = {1, 2, 3}, M={a, b} then, E - M = E
  - C = {Tom, Bob, Pete}, then C - $\emptyset$ = C
  - For every set A, A - A = $\emptyset$

Sample Application with sets

- Program to keep track of students in courses
  - Example: ICOM 4035 and ICOM 6005.
  - Operations
    - Add a student to a course
    - Remove a student from a course
    - Report all students in all courses
    - Report all students taking both courses
    - Report students taking one course but not the other.
  - Use sets to implement object that represent the students in each class.
  - Use set operations to answer each of these questions

Alternative implementation of sets

- Allows for faster, constant time search and delete operation on the set.
  - Implementation for array require looking up all array.
- Alternative is called bit vector
- Use an array of unsigned char or unsigned int to represent sets of integers.
  - Each bit represents a number.
  - The bit at position k represents integer k.
  - If k is in the set, then the bit is set to 1, otherwise it is set to 0.
- Names, places and other objects to which we can assign a number can also be represented this way.

Example:

- Suppose we have array of size 2 of unsigned char
- Rightmost position is 0, numbers increase to left.
- 0100001100011100
  - Represents set {2,3,4,8,9,14}
  - Storage requirements: 2 bytes
  - Previous implementation would have required at 5 bytes.
  - In fact, with 2 bytes we can represent up to 16 numbers.
- Each byte is a segment of the bit vector.
- To find an element we need to know segment number and its position within segment to locate it
  - Segment is given by: element / segment size
  - Position is given by: element % segment size

Example

- Given 2 byte bit-vector 0100001100011100
- Where is element 1 located?
  - Segment 0, position 1
- Where is element 4 located?
  - Segment 0, position 4
- Where is element 8 located?
  - Segment 1, position 0
- Where is element 10 located?
  - Segment 1, position 2
- Where is element 19 located?
  - Cannot be represented due to lack of space.
  - Bit vector n bytes can only represent (n * 8) – 1 elements

Boolean Algebra

- Needed to inspect bits in the bit vector.
- Supported operations: AND, OR, XOR
- Also need bit-wise shift operators that are used to move bits from a position to another.
- Left shift – move all bits n spots to the left, discarding elements that reach and pass last position.
  - 0101 << 1 = 1010 (shift one)
  - 0101 << 3 = 1000 (shit three)
- Right shift – move all bits n spots to the right, discarding element that reach and pass first position.
  - 0101 >> 1 = 0010 (shift one)
  - 0101 >> 3 = 0001 (shift three)

AND Operator

- Expression in C++: A & B
- Truth Table:

B

A

AND	0	1
0	0	0
1	0	1

Example:

01001101
&
11001100

01001100

OR Operator

- Expression in C++: A | B
- Truth Table:

B

A

OR	0	1
0	0	1
1	1	1

Example:

```
 01001101
     |
 11001100
 ________
 11001101
```

XOR Operator

- Expression A ^ B
- Truth Table:

B

A

XOR	0	1
0	0	1
1	1	0

Example:

```
 01001101
     ^
 11001100
 --------
 10000001
```