

COM- 201

Data Structures and Algorithms

Assist. Prof. Dr. Özge ÖZTİMUR KARADAĞ

ALKÜ

Department of Computer Engineering

COM 201

- Course material will be available at lms.alanya.edu.tr
- Suggested Books:
 - Mark A. Weiss; “Data Structures and Algorithm Analysis in C++”, 4th Ed., Pearson, 2013.
 - Theory and Problems of Data Structures, Seymour Lipschutz, McGraw-Hill
- Grading
 - Programming Assignments: %20 (10+10)
 - Midterm: %30
 - Final Exam: %50

Syllabus

- Tentative outline of the course

Week	Topic	
1	--	
2	--	
3	Introduction and Overview	
4	Arrays, Records and Pointers	
5	Arrays, Records and Pointers	
6	Linked Lists	
7	Linked Lists	Programming Assignment – 1
8	Midterm	
9	String Processing	
10	Stack, Queues	
11	Recursion	
12	Trees	
13	Trees	Programming Assignment – 2
14	Graphs	
15	Graphs	

What have you covered?

- Construct an algorithm to solve a problem,
 - Represent the algorithm as a dataflow diagram,
 - Given a dataflow diagram derive the algorithmic steps.
 - C programming language
 - ...
-
- Data structures course introduces concepts in a general perspective,
 - Concepts introduced in this course can be generalized /adapted to all programming languages.
 - We will do applications in C.

Data

- Data are simply values or sets of values.
- *A data item*: single unit of values
- *Group items*: data items that are divided into subitems
- *Elementary items*: data items that are not divided into subitems
 - Ex:
 - an employee's name may be divided into
 - First name
 - Middle name
 - Last name
 - Social security number is a single item

Collection of Data

- is usually organized as:
 - Fields,
 - Records,
 - Files
- An entity is smth. that has certain attributes/properties which can be assigned values.
- Values can be either numeric or nonnumeric
- Example to attributes and their corresponding values:

Attributes	Name	Age	Sex	SSN
Values	Genç, Ayşe	32	F	134-23-5544

Collection of Data

- Entities with similar attributes form an entity set.
- Each attribute of an entity set has a *range* of values, the set of all possible values that could be assigned to the particular attribute.
- A *field* is a single elementary unit of information representing an attribute of an entity.
- A *record* is the collection of field values of a given entity,
- A *file* is the collection of records of the entities in a given entity set.

Collection of Data

- Each record in a file may contain many field items, but the value in a certain field may uniquely determine the record in the file. Such a field K is called a *primary key*, and the values k_1, k_2, \dots in such a field are called *keys* or *key values*.
- Example: inventory of an automobile dealership
 - Serial number,
 - Type,
 - Year,
 - Price,
 - Accessories

Organization of Data

- Data are also organized into more complex types of structures. The study of such data structures, which form the subject matter of this course includes the following steps:
 - Logical or mathematical description of the structure,
 - Implementation of the structure on a computer,
 - Quantitative analysis of the structure, which includes determining the amount of memory needed to store the structure and the time required to process the structure.
- In the scope of this course, we mainly deal with the data stored in the main memory, while the database course deals with the data in the secondary storage.

Data Structures

- The logical or mathematical model of a particular organization of data is called a *data structure*.
- The choice of a particular data model depends on two considerations:
 - it must be rich enough in structure to mirror the actual relationships of data in the real World.
 - Structure should be simple enough that one can effectively process the data when necessary.

Data Structures

- Arrays

- Linear (One dimensional) array: list of a finite number n of similar data elements referenced respectively by a set of n consecutive numbers, usually $1, 2, 3, \dots, n$
- If the name of the array is A , then the elements of A are denoted by
 - The subscript notation: $a_1, a_2, a_3, \dots, a_n$
 - The parenthesis notation: $A(1), A(2), A(3), \dots, A(n)$
 - The bracket notation: $A[1], A[2], A[3], \dots, A[n]$
- Regardless of the notation, the number K in $A[K]$ is called a subscript and $A[K]$ is called a subscripted variable.

Data Structures

- Arrays
 - Example: A linear array STUDENT consists of the names of six students

STUDENT	
1	John Brown
2	Sandra Gold
3	Tom Jones
4	Jane Kelly
5	Mary Reed
6	Alan Smith

- Linear arrays are called one-dimensional arrays because each element in such an array is referenced by one subscript.
- A two dimensional array is a collection of similar data elements where each element is referenced by two subscripts.

Data Structures

- Arrays

- Example: A chain of 28 stores, each store having 4 departments, may list its weekly sales as follows:

Store \ Dept.	1	2	3	4
	1	2	3	4
1	2872	805	3211	1560
2	2196	1223		1744
3	3257	1017		1951
...
28	2618	931	2333	982

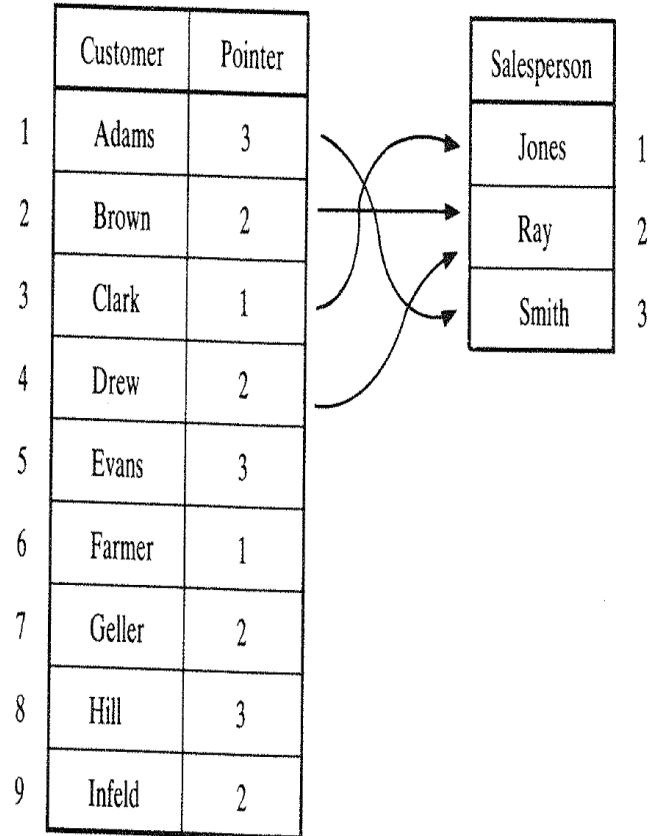
- Can be stored using a two dimensional array, first subscript denotes the store and the second subscript denotes the department.
 - If the name of the array is SALES;
 - $\text{SALES}[1,1]=2872$, $\text{SALES}[1,2]=805$, ... $\text{SALES}[28,4]=982$

Data Structures

- Example: A brokerage firm maintains a file where each record contains a customer's name and his or her salesperson, and suppose the file contains the following data:
- How to store this data?
- Is it the most useful way?

	Customer	Salesperson
1	Adams	Smith
2	Brown	Ray
3	Clark	Jones
4	Drew	Ray
5	Evans	Smith
6	Farmer	Jones
7	Geller	Ray
8	Hill	Smith
9	Infeld	Ray

Data Structures



- What if the firm needs the list of customers for a given salesperson??
 - Have to search through the entire customer file.
- Alternative?

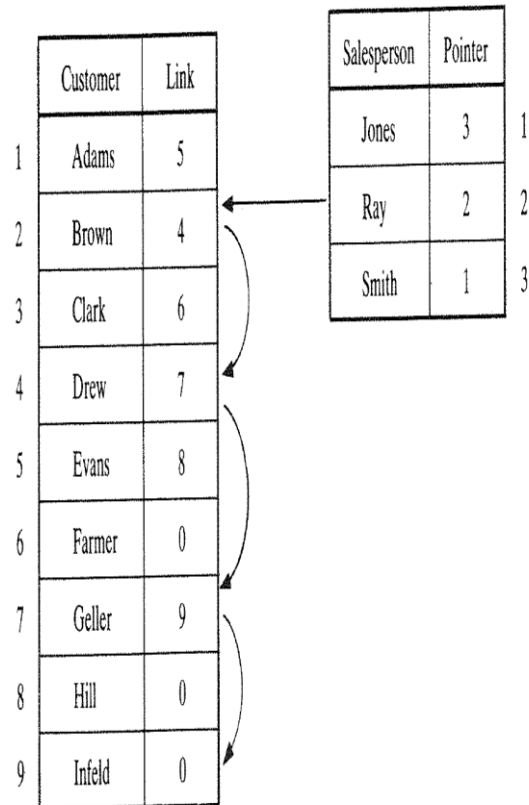
Data Structures

	Salesperson	Pointer
1	Jones	3, 6
2	Ray	2, 4, 7, 9
3	Smith	1, 5, 8

- Each salesperson now have a set of pointers giving the position of his/her customers.
- Disadvantage?
 - Each salesperson may have several pointers and the set of pointers will change as customers are added and deleted.

Data Structures

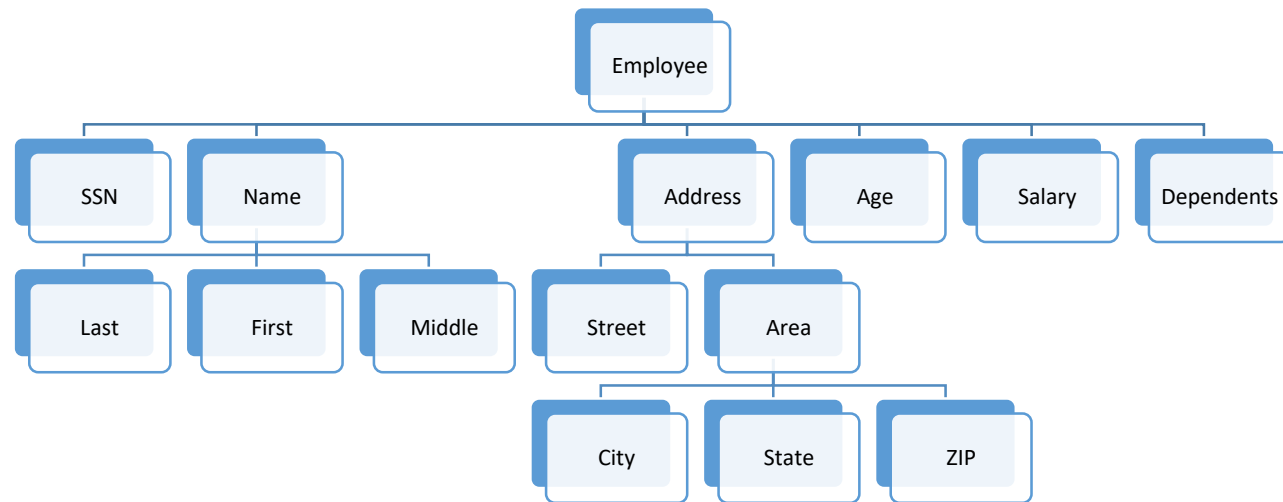
- **Linked Lists**



- Each salesperson has one pointer which points to his/her first customer, whose pointer in turn points to the second customer, and so on, the salesperson's last customer indicated by a 0.
- Advantage:
 - Easy to obtain the entire list of customers for a given salesperson,
 - Easy to insert and delete customers.

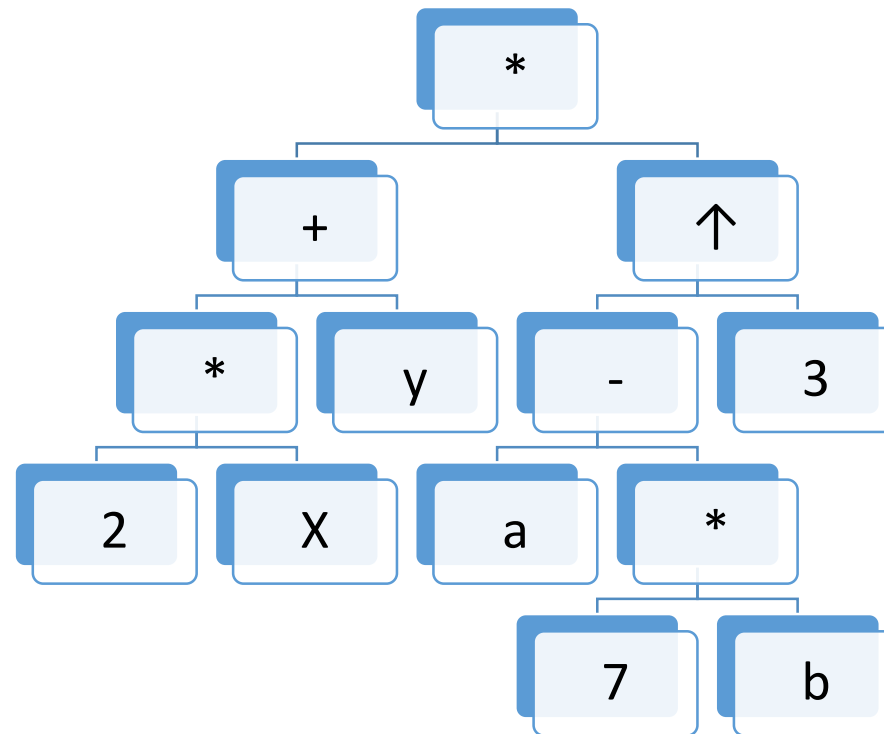
Data Structures

- Trees
 - The data structure which reflects a hierarchical relationship between various elements, a rooted tree graph.



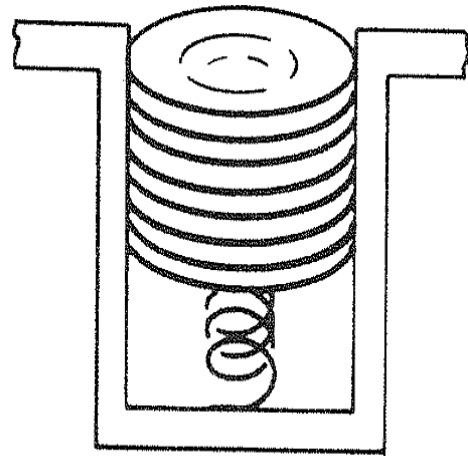
Data Structures

- Trees
 - Ex: Algebraic expression
 - $(2x + y)(a - 7b)^3$



Data Structures

- Stack (Last in First Out – LIFO - System)
 - A linear list in which insertions and deletions can take place only at one end, called the top.



(a) Stack of dishes.

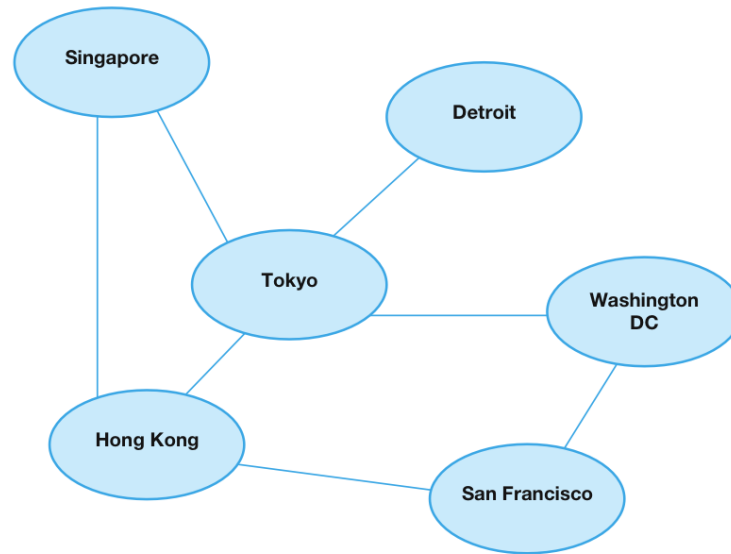
Data Structures

- Queue (First in first out – FIFO – system)
 - is a linear list in which
 - deletions can take place only at one end of the list, the 'front' of the list,
 - Insertions can take place only at the other end of the list, the 'rear' of the list.



Data Structures

- Graph
 - Pairs of data have relationship other than hierarchy.
 - Ex: Airline flights



Data Structure Operations

- The particular data structure that one chooses for a given situation depends largely on the frequency with which specific operations are performed.
- Most frequently used operations:
 - **Traversing**: Accessing each record exactly once so that certain items in the record may be processed. (This accessing and processing is sometimes referred to as 'visiting' the record.)
 - **Searching**: Finding the location of the record with a given key value, or finding the locations of all the records which satisfy one or more conditions.
 - **Inserting**: Adding a new record to the structure.
 - **Deleting**: Removing a record from the structure.
- Some other operations:
 - **Sorting**: Arranging the records in some logical order (e.g. alphabetically according to some NAME key, or in numerical order according to some NUMBER key, such as social security number or account number)
 - **Merging**: Combining the records in two different sorted files into a single sorted file

Algorithms: Complexity, Time-Space Tradeoff

- Algorithm is a well-defined list of steps for solving a particular problem.
- Time and space it uses are two major measures of the efficiency of an algorithm.
- The complexity of an algorithm is the function which gives the running time and/or space in terms of the input size.
- Depending on the data structure the efficiency of the algorithm varies.
- Choice of the data structure depends on:
 - Type of data,
 - Frequency of data operations etc.
- Sometimes the choice of data structure involves a time-space tradeoff: by increasing the amount of space for sorting the data, one may be able to reduce the time needed for processing the data, or vice versa.

Algorithms: Complexity, Time-Space Tradeoff

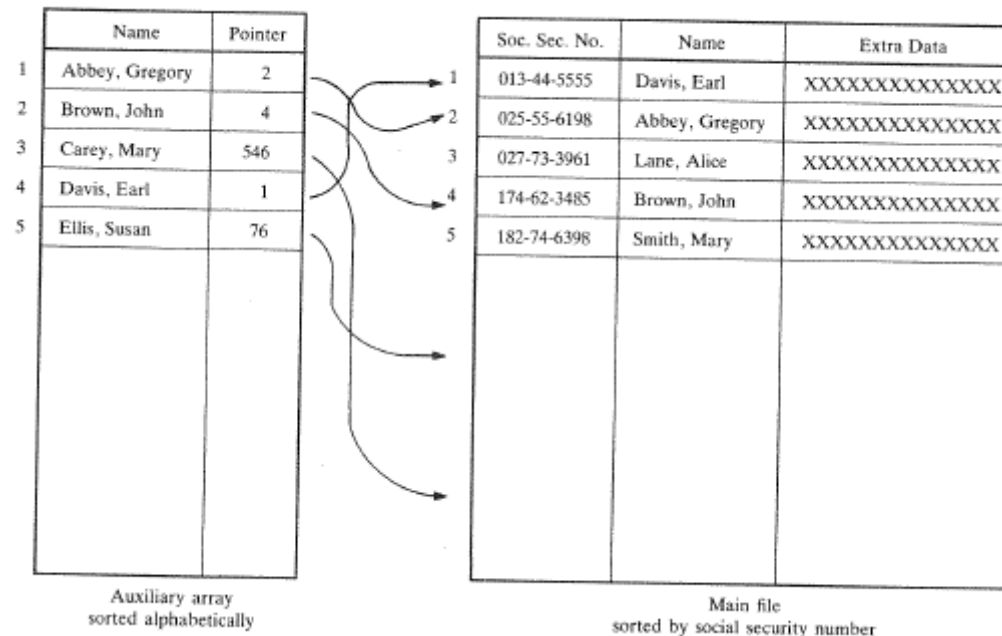
- Ex: Searching algorithms
 - An organization contains a membership file in which each record contains the following data for a given member:
 - Name, address, telephone number, age, sex
 - For a given name, find his/her telephone number
 - Linear search: Search each record of the file, one at a time, until finding the given Name and hence the telephone number. $\rightarrow C(n)=n/2$
 - Binary search: If the names are sorted alphabetically, compare the Name with the name in the middle of the list; this tells which half of the list contains Name. Then compare Name with the name in the middle of the correct half to determine which quarter of the list contains Name. Continue the process until finding Name in the list. $\rightarrow C(n)=\log_2 n$
 - Disadvantage of Binary search:
 - It assumes direct access to the middle of the list, which means that the list is stored as array. Unfortunately, inserting an element in an array requires elements to be moved down the list, and deleting an element from an array requires elements to be moved up the list.

Algorithms: Complexity, Time-Space Tradeoff

- An example of time-space tradeoff: A file of records contains names, SSN and additional information.
 - Sorting the file alphabetically and using Binary search is a very efficient way to find the record for a given name.
 - If we are given only the SSN, we would have to do a linear search which is extremely time consuming for a very large number of records.
- Solution??
 - Have another file which is sorted numerically according to SSN →
 - doubles the space required

Algorithms: Complexity, Time-Space Tradeoff

- Have the main file sorted numerically by SSN, have an auxiliary array with only two columns, the first column containing the alphabetized list of the names and the second column containing the pointers which give the locations of the corresponding records in the main file. This is the frequently used solution, as it additional space is minimal.



Algorithms: Complexity, Time-Space Tradeoff

- Suppose a file is sorted numerically by SSN. As new records are inserted into the file, data must be constantly moved to new locations in order to maintain the sorted order. One simple way to minimize the movement of data is to have the SSN serve as the address of each record. There will be no movement of data while inserting, there will be instant Access to any record.
- What is the disadvantage?
 - It would require one billion (10^9) memory locations for only hundreds or thousands of records. → tradeoff of space for time is not worth the expense.

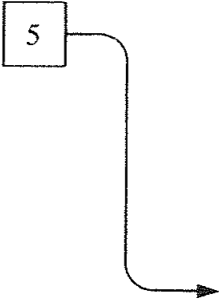
Sample Questions

- Consider the linear array NAME which is sorted alphabetically
 - Find NAME[2], NAME[7]
 - Suppose Davis is to be inserted into the array. How many names must be moved to new location?
 - Suppose Gupta is to be deleted from the array. How many names must be moved to new locations?

NAME	
1	Adams
2	Clark
3	Evans
4	Gupta
5	Jones
6	Lane
7	Pace
8	Smith

Sample Questions

- Consider the linear array NAME in the figure. The values of FIRST and LINK[K] in the figure determine a linear ordering of the names as follows. FIRST gives the location of the first name in the list, and LINK[K] gives the location of the name following NAME[K], with 0 denoting the end of the list. Find the linear ordering of the names.



The diagram shows a box labeled 'FIRST' containing the number '5'. An arrow points from this box to the row index '5' in the NAME array table.

	NAME	LINK
1	Rogers	7
2	Clark	8
3		
4	Hansen	10
5	Brooks	2
6	Pitt	1
7	Walker	0
8	Fisher	4
10	Leary	6

Sample Questions

- Consider the algebraic expression $(7x+y)(5a-b)^3$. Draw the corresponding tree diagram.
 - Find the scope of the exponential operation. (the scope of node v in a tree is the subtree consisting of v and the nodes following v .)
 - Use a \uparrow for exponential and $*$ for multiplication.

Sample Questions

- The Daily flights of an airline company is given CITY lists the cities, ORIG[K] and DEST[K] denote the cities of origin and destination, respectively, of the flight NUMBER[K]. Draw the corresponding directed graph of the data.

	CITY
1	Atlanta
2	Boston
3	Chicago
4	Miami
5	Philadelphia

(a)

	NUMBER	ORIG	DEST
1	701	2	3
2	702	3	2
3	705	5	3
4	708	3	4
5	711	2	5
6	712	5	2
7	713	5	1
8	715	1	4
9	717	5	4
10	718	4	5

(b)

Sample Questions

- For the previous question, how the should be stored:
 - Find the origin and destination of a flight, given the flight number.
 - Given city A and city B, find whether there is a flight from A to B, and if there is find its flight number.

References

- Theory and Problems of Data Structures, Seymour Lipschutz, McGraw-Hill