ENSF 593/594
Data Structures – Arrays

Mohammad Moshirpour

Outline

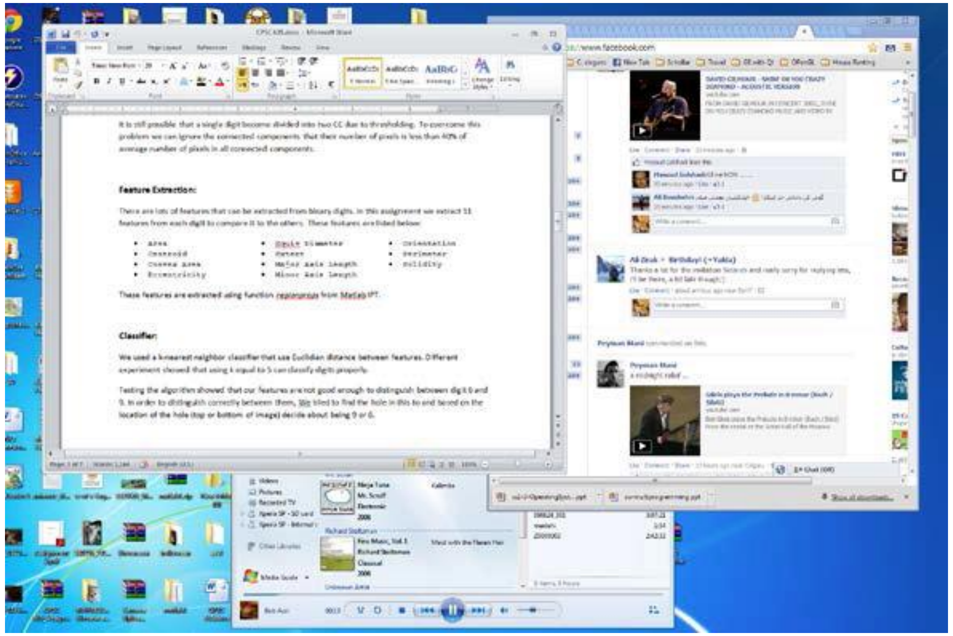
- List
- Arrays
  - Insertion
  - Deletion
- Vector

Goal

- In this lecture we will introduce you to lists and how to implement lists as arrays.

Multitasking in Operating System

- Your computer can only preform one task at a time, but how can you listen to musing with checking your Facebook notifications meanwhile writing your assignment at MS Word?



It is still possible that a single digit become divided into two CC due to thresholding. To overcome this problem we can ignore the connected components that their number of pixels is less than 40% of average number of pixels in all connected components.

Feature Extraction:

There are lots of features that can be extracted from binary digits. In this assignment we extract 11 features from each digit to compare it to the others. These features are listed below:

- Area
- Centroid
- Convex Area
- Eccentricity
- EquivDiameter
- Extent
- Major Axis Length
- Minor Axis Length
- Orientation
- Perimeter
- Solidity

These features are extracted using function regionprops from Matlab IPT.

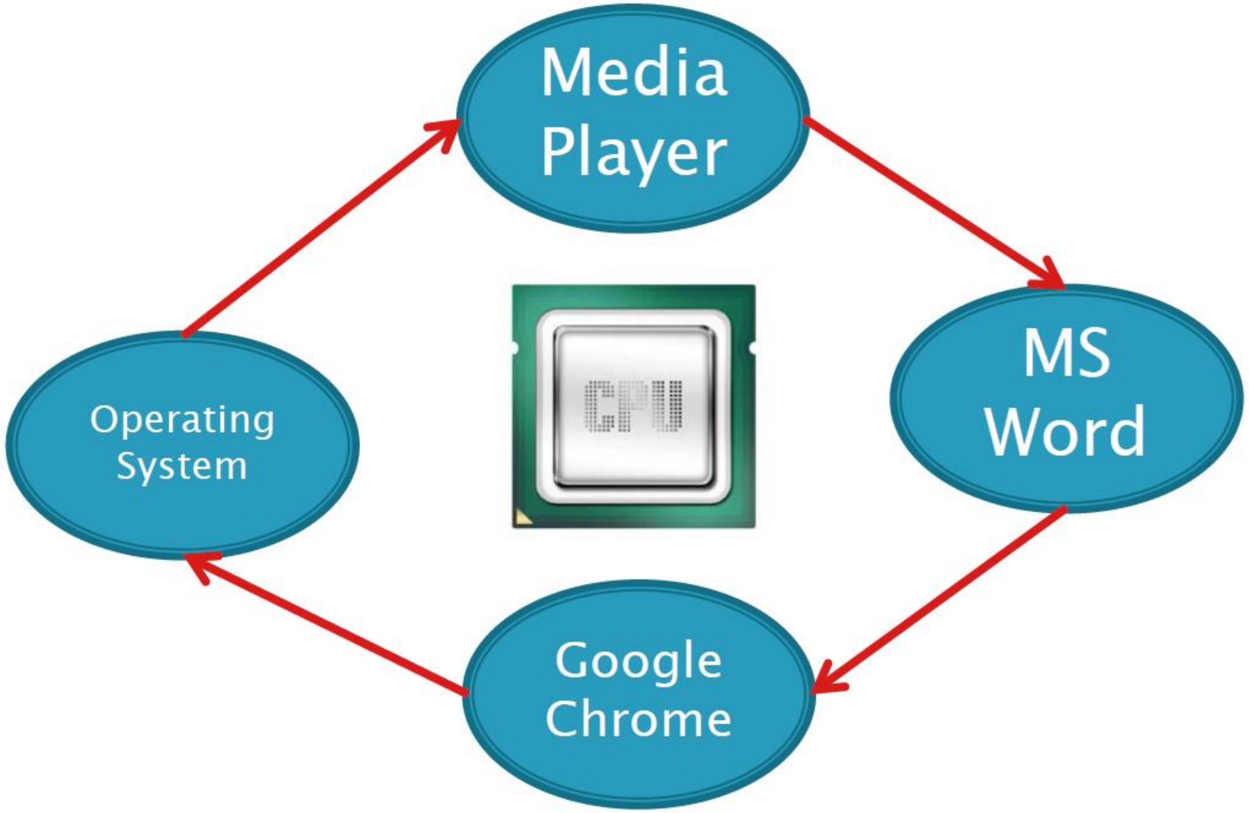
Classifier:

We used a k-nearest neighbor classifier that use Euclidian distance between features. Different experiment showed that using k equal to 5 can classify digits properly.

Testing the algorithm showed that our features are not good enough to distinguish between digit 6 and 9. In order to distinguish correctly between them, We tried to find the hole in this to and based on the location of the hole (top or bottom of image) decide about being 9 or 6.

Multitasking in Operating System (Cont'd)

- Circular Link List



Lists

- Are classified as linear data structures
- May be one-dimensional, or multi-dimensional
- Each element of a list might consist of:
  - A single data item, or
  - A record or object (compound data)
- Lists may be ordered (sorted) or unordered

Lists (cont’d)

- A list is an Abstract Data Type (ADT) that supports these operations:
  - add(newEntry)
    - Adds an item to the end of the list
  - insert(newEntry, position)
    - Inserts an item into a list at the specified position
  - delete(position)
    - Deletes the item at the specified position
  - clear()
    - Deletes all items from the list

Lists (cont’d)

- getEntry(position)
  - Return the item at the specified position
    - May be done by value or by reference
- replaceEntry(position, newEntry)
  - Overwrite the item at the specified position with a new item
- getLength()
  - Returns the number of items currently in the list
- isEmpty()
  - Returns true if no items in the list, false otherwise
- isFull()
  - Returns true if the list is full, false otherwise

Lists (cont’d)

- display()
  - Prints out all items in the list
- contains(itemKey)
  - Returns true if the list contains the item, false otherwise
- search(itemKey)
  - Returns the item that matches the key (or nil if no match)

Lists (cont’d)

- Lists may be implemented in many ways
  - E.g. Arrays, linked lists
  - In RAM, or in secondary storage (file on disk)

Arrays

- Are also called *physically ordered lists*
- Definition: Are linear, random access data structures, whose elements are accessed by a unique identifier called an *index* or *subscript*. Elements are stored contiguously in RAM or in secondary storage.

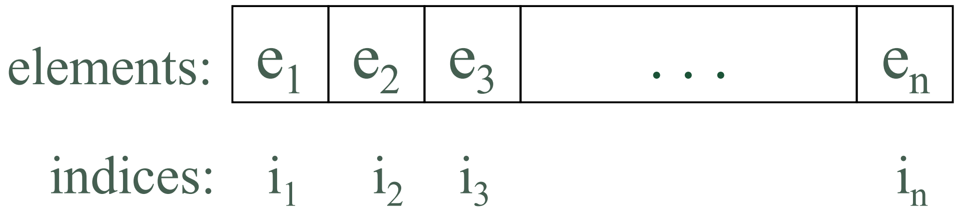
Arrays (cont’d)

- Most modern programming languages directly support arrays
  - Java example: int[] array = new int[10];
- Structure:
  - Formal view:

$$\begin{array}{rcccccl}
\text{Set of array elements } E = \{ & e_1, & e_2, & e_3, & \ldots, & e_n & \} \\
\text{Mapping function } f: & \uparrow & \uparrow & \uparrow & & \uparrow & \\
\text{Set of array indices } I = \{ & i_1, & i_2, & i_3, & \ldots, & i_n & \}
\end{array}$$

Arrays (cont'd)

- Programming language view:



elements:	e_1	e_2	e_3	. . .	e_n
indices:	i_1	i_2	i_3		i_n

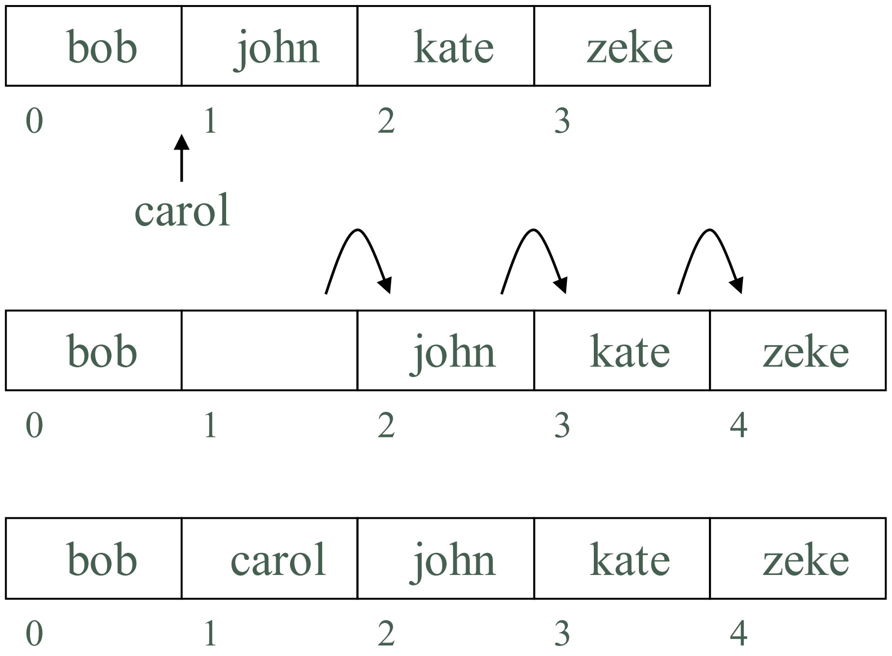
- Languages like C, C++, and Java require indices to start at 0
  - Others, like Pascal and FORTRAN are more flexible

Arrays (cont’d)

- Arrays are fixed in length:
  - At compile-time for most languages
  - At run-time for languages like Java
- Imposes a maximum size for a list
  - May run out of room as we add items
  - Wastes space if we use only part of the array

Arrays (cont'd)

- Inserting an item into an array may require shifting elements to make room for it



Arrays (cont'd)

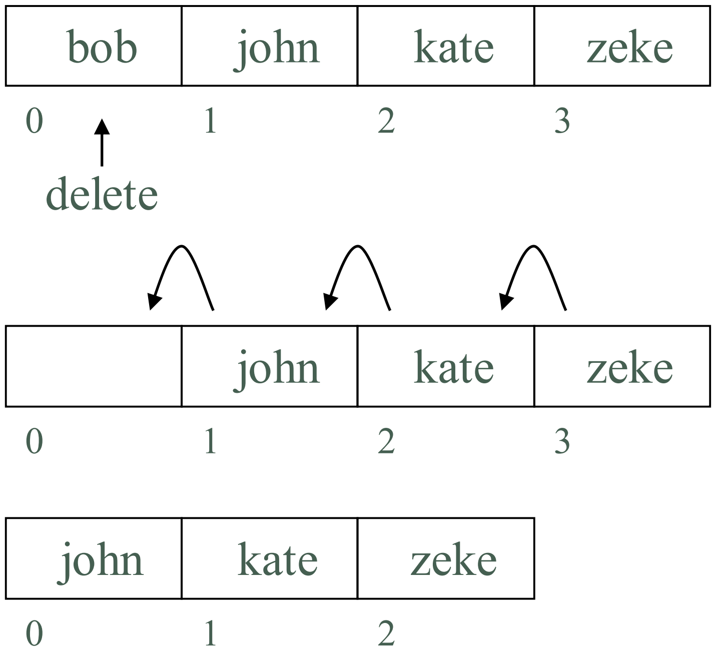
- Pseudocode:

```
insert(newEntry, position)
   for (i = n-1; i >= position; i--)
      array[i+1] = array[i]
   array[position] = newEntry
   n = n + 1
```

- Is *O*(*n*) in the worst case (inserting at position 0)

Arrays (cont'd)

- Deleting an item may require shifting items to fill the gap



bob	john	kate	zeke
0	1	2	3

delete

	john	kate	zeke
0	1	2	3

john	kate	zeke
0	1	2

Arrays (cont'd)

- Pseudocode:

```
delete(position)
    for (i = position; i < n-1; i++)
        array[i] = array[i+1]
    n = n - 1
```

- Is $O(n)$ in the worst case (deleting at position 0)
- Accessing an item by position is $O(1)$
    - i.e. Getting or replacing entries is very quick

Arrays (cont’d)

- Must use sequential search on unordered arrays
- Can use sequential or binary search on ordered arrays
- A *vector* is an array that grows in size when it overflows (e.g. inserting into a full array)
  - A new array of size 2N is allocated
  - All elements from the old array are copied to the new array
  - The original reference is changed to point to the new array

Arrays (cont’d)

- In addition to ordinary arrays, Java provides the classes:
  - java.util.Vector
  - java.util.ArrayList

List Implemented As Arrays

- Advantages
    - Simple to use (often a built-in type)
    - Retrievals are quick if the index is known (*O(n)*)
- Disadvantages
    - Adding/removing elements may be awkward
    - Fixed size arrays either limits the size of the list or wastes space
    - Dynamic sized arrays requires copying

Summary

- Lists are linear data structure.
- Two common ways to implement lists are arrays and linked list.
- Array is a linear, random access data structure that stores in contiguously in memory.
- Arrays are fixed size in length.
- Insertion and deleting from arrays in O(n) in worse case
- Vectors can solve the problem of overflowing in arrays.

Review Questions

- What are two ways to implement lists?
- What is an array?
- What is the problem of Imposing a maximum size for a list?
- How can you insert an element to a specific location in an array and what is its complexity in worse, average, and best case?
- How can you delete an element from a specific location in an array and what is its complexity in worse, average, and best case?
- What is a vector and how does it fix the problem of overflowing?
- What are the advantage and disadvantage of using array to implement a list?