ECE 217:
Data Structure and Algorithm

Lecture 3: Searching, Sorting, and the "vector" Type

Instructor: Shayan (Sean) Taheri, Ph.D.

Assistant Professor

The Department of Electrical and Cyber Engineering (ECE)

The Institute for Health and Cyber Knowledge (I-HACK)

The Gannon University (GU)

Personal Information

- Name: Shayan (Sean) Taheri.
- Date of Birth: July/28/1991.
- Past Position: Postdoctoral Fellow at University of Florida.
- Ph.D. Degree: Electrical Engineering from the University of Central Florida.
- M.S. Degree: Computer Engineering from the Utah State University.
- University Profile: https://www.gannon.edu/FacultyProfiles.aspx?profile=taheri001

List Processing

- List: a collection of values of the same type
- Array is a convenient place to store a list
- Basic list operations:
  - Search the list for a given item
  - Sort the list
  - Insert an item in the list
  - Delete an item from the list
  - Print the list

Searching

- Sequential search algorithm:
  - Not very efficient for large lists
  - On average, number of key comparisons is equal to half the size of the list
  - Does not assume that the list is sorted
- If the list is sorted, the search algorithm can be improved

Example – Sequential Search

Write a program that will:

1. initialize an array of 20 integers with random values between 10 and 99 inclusive,
2. display the values of all array elements,
3. prompt the user to enter a value to search for, and then
4. display the location of the element in the array if found or an error message otherwise.

Write 3 functions to support your program:

5. one to load the array with random numbers between 10 & 99,
6. another to display the array elements, and
7. a third to return the position of the search value in the array if found or -1 otherwise.

Sequential Search (Cont.)

```
#include <iostream>
#include <cstdlib>
using namespace std;

const int ARRAY_SIZE = 20;

void loadList(int list[], int listSize);
void printList(int list[], int listSize);
int seqSearch(int list[], int listSize, int searchValue);

int main()
{
   int list[ARRAY_SIZE];
   int searchValue;
```

Sequential Search (Cont.)

```
    loadList(list, ARRAY_SIZE);
    printList(list, ARRAY_SIZE);

    cout << "Value to search for?: ";
    cin >> searchValue;

    int pos = seqSearch(list, ARRAY_SIZE, searchValue);

    if (pos == -1)
       cout << "Element " << searchValue
            << " is not in the array." << endl;
    else
       cout << "Found element " << searchValue
            << " at position " << pos << endl;

       return 0;
}
```

Sequential Search (Cont.)

```
void loadList(int list[], int listSize)
{
   for (int i=0; i<listSize; i++)
      list[i] = 10 + rand()%90;
}


void printList(int list[], int listSize)
{
   for (int i=0; i<listSize; i++)
      cout << list[i] << " ";
   cout << endl;
}
```

Sequential Search (Cont.)

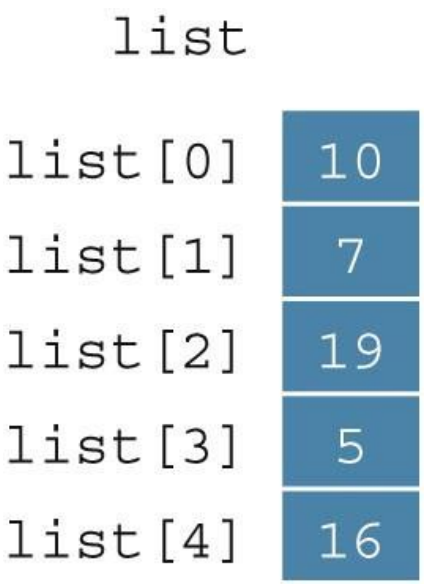
```
int seqSearch(int list[], int listSize, int searchValue)
{
   int pos = 0;
   bool found = false;

   while (pos<listSize && !found) {
      if (list[pos] == searchValue)
         found = true;
      else
         pos++;
   }

   if (found)
      return pos;
   else
      return -1;
}
```

Bubble Sort

- `list[0]...list[n - 1]`
  - List of *n* elements, indexed 0 to *n* – 1
  - Example: a list of five elements (Figure 16-1)



Bubble Sort (cont'd.)

- Series of `n-1` iterations
  - Successive elements `list[index]` and `list[index+1]` are compared
  - If (`list[index] > list[index+1]`)

    `list[index]` and `list[index+1]` are swapped
  - Smaller elements move toward the top (beginning of the list)
  - Larger elements move toward the bottom (end of the list)

Bubble Sort (cont'd.)

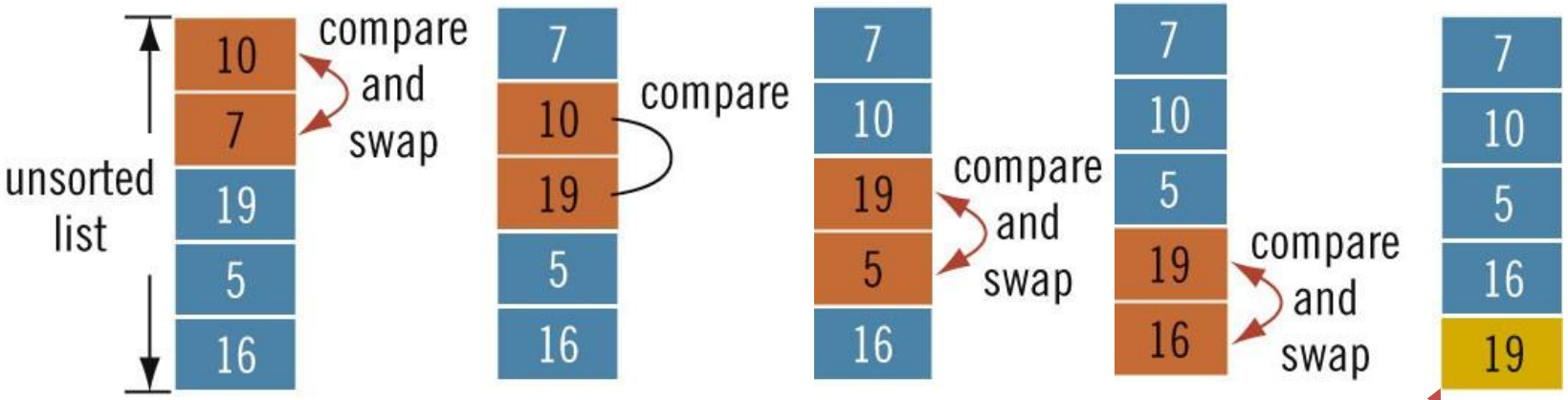


FIGURE 16-2 Elements of `list` during the first iteration

Largest Element Sink to the bottom

Bubble Sort (cont'd.)

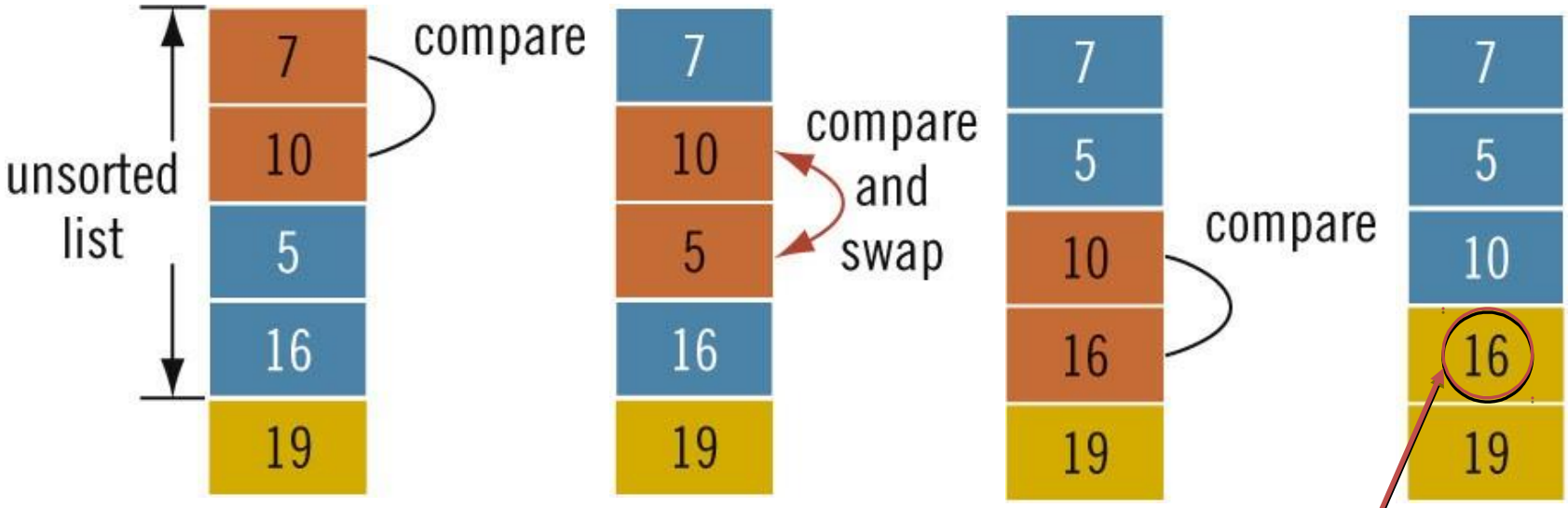


FIGURE 16-3 Elements of `list` during the second iteration

Second Largest Element Sink to the bottom

Bubble Sort (cont'd.)

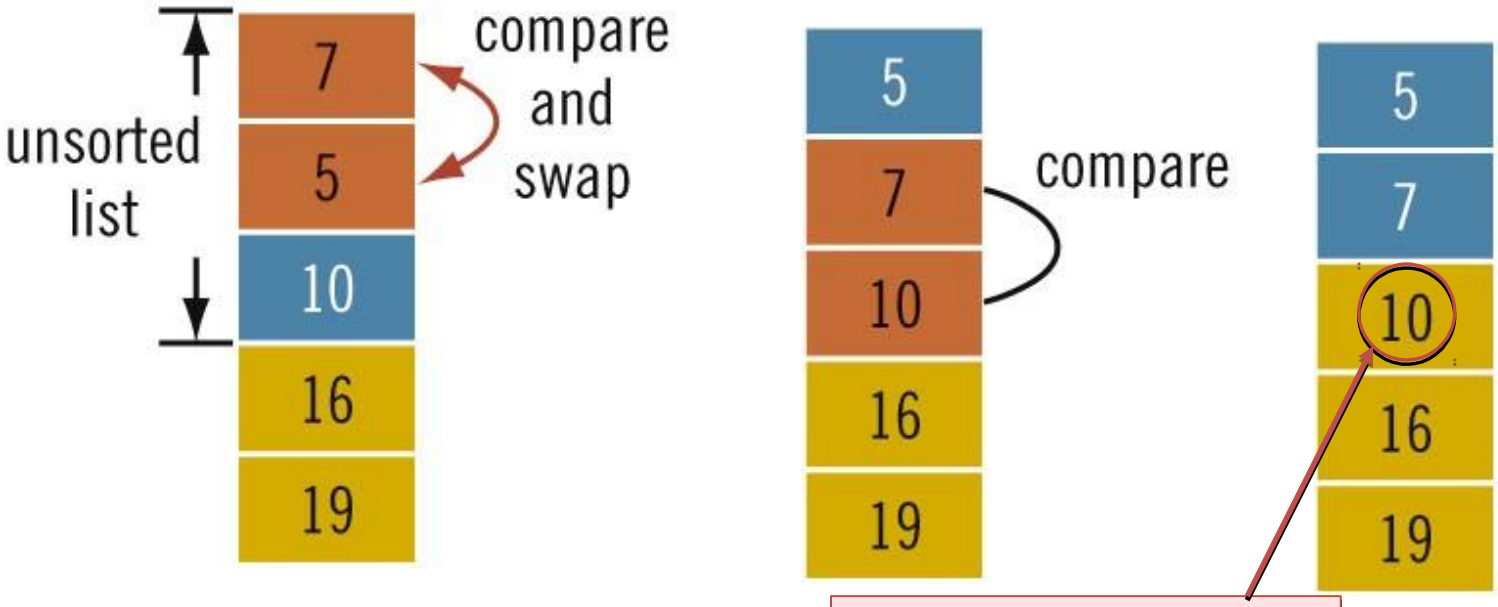


FIGURE 16-4 Elements of `list` during the third

Third Largest Element
Sink to the bottom

Bubble Sort (cont'd.)

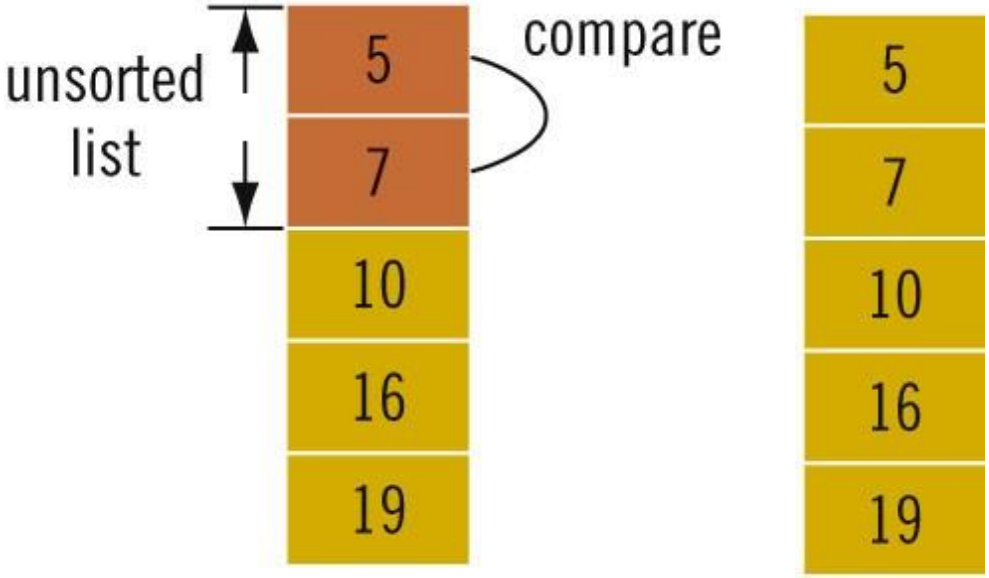


FIGURE 16-5 Elements of `list` during the fourth iteration

Example

Write a function and a driver program to test the bubble sort.

Bubble Sort

```
void bubbleSort(int list[], int listSize) {
    int temp;

    for (int iteration=1; iteration<listSize; iteration++)
        {
         for (int index=0; index<listSize-iteration; index++)
             if (list[index] > list[index+1])
          {
             temp = list[index];
             list[index] = list[index+1];
             list[index+1] = temp;
             }
    }
}
```

How can you improve the code?

Bubble Sort (cont'd.)

- List of length *n*
  - Exactly $n(n - 1) / 2$ key comparisons
  - On average $n(n - 1) / 4$ item assignments
- If $n = 1000$
  - 500,000 key comparisons and 250,000 item assignments
- Can improve performance if we stop the sort when no swapping occurs in an iteration

Insertion Sort

- Sorts the list by moving each element to its proper place

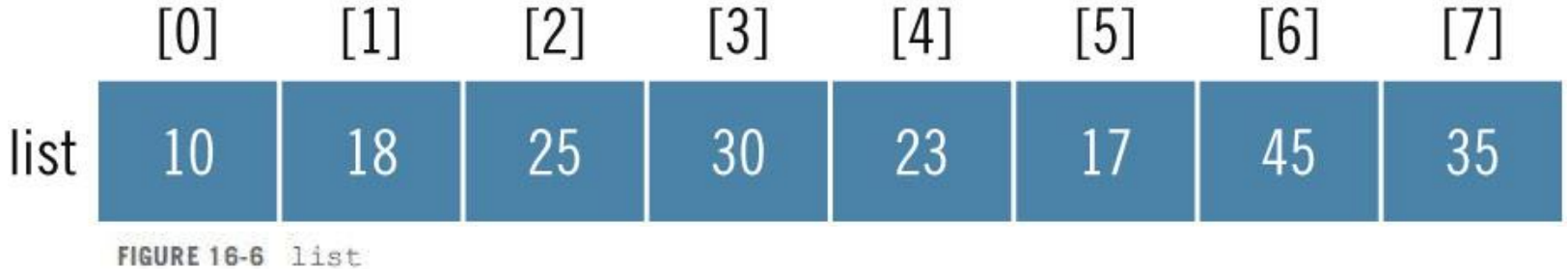


FIGURE 16-6 list

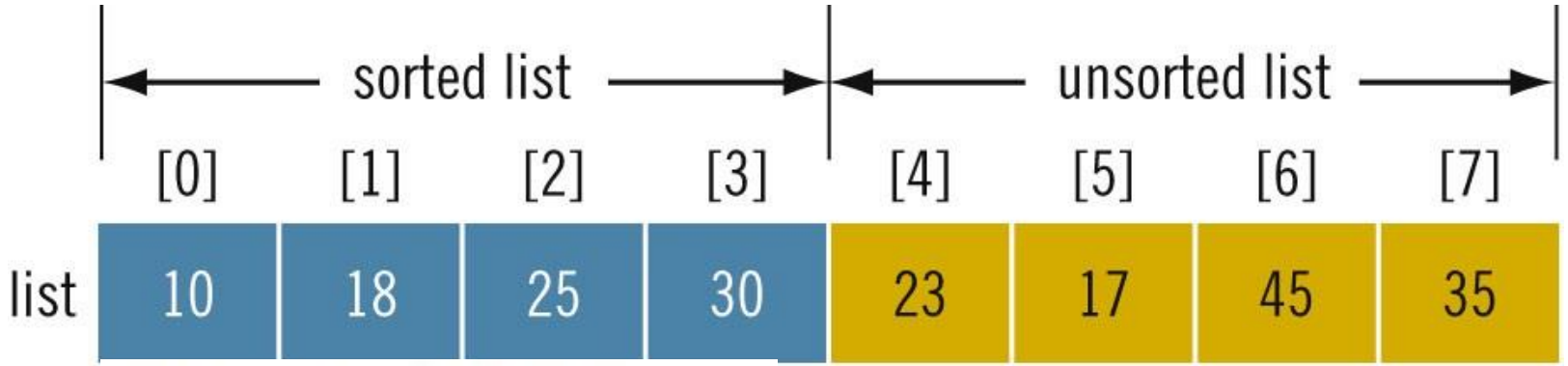
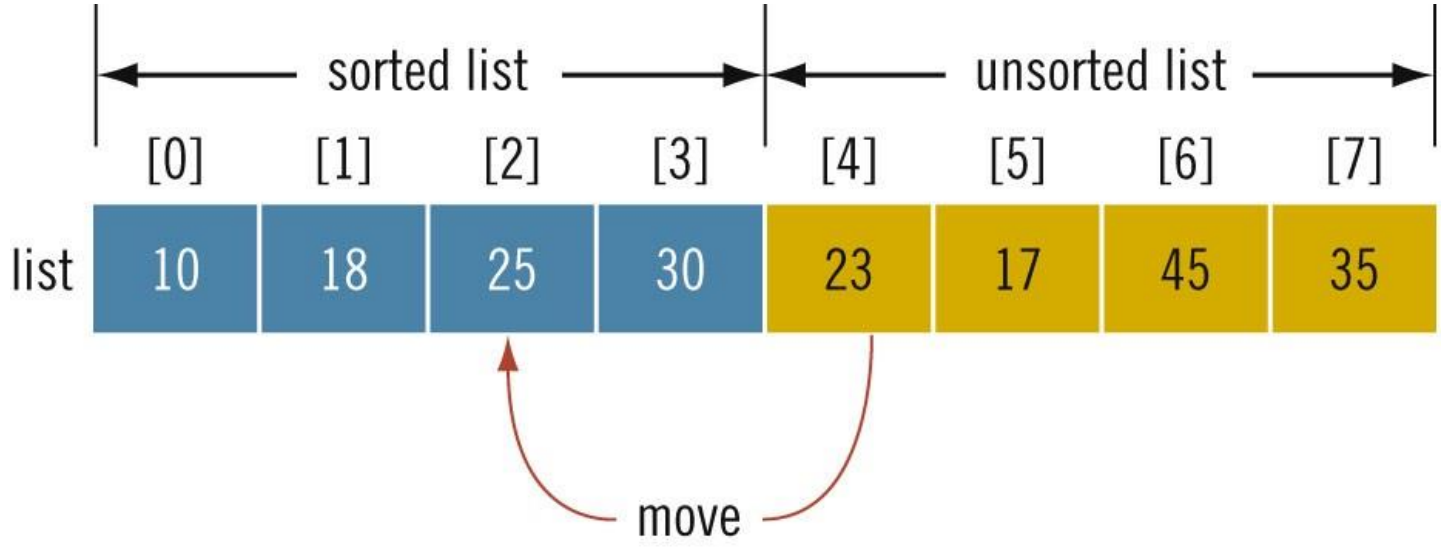


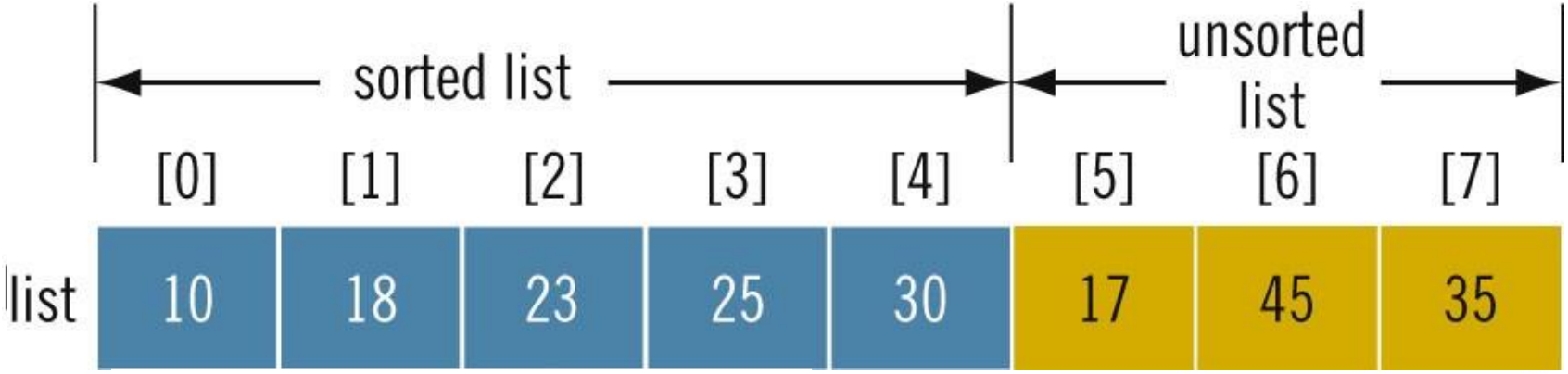
FIGURE 16-7 Sorted and unsorted portion of list

Insertion Sort (cont'd.)

- Consider the element `list[4]`
  - First element of unsorted list
  - `list[4] < list[3]`
    - Move `list[4]` to proper location at `list[2]`



Insertion Sort (cont'd.)



Insertion Sort (cont'd.)

- During the sorting phase, the array is divided into two sublists: sorted and unsorted
  - Sorted sublist elements are sorted
  - Elements in the unsorted sublist are to be moved into their proper places in the sorted sublist, one at a time

Example

Write a function and a driver program to implement and test insertion sort.

Insertion Sort

```
void insertionSort(int list[], int listSize)
{
   int firstOutOfOrder, location;
   int temp;

   for (firstOutOfOrder = 1; firstOutOfOrder < listSize;
                             firstOutOfOrder++)
      if (list[firstOutOfOrder] < list[firstOutOfOrder-1]) {
         temp = list[firstOutOfOrder];
         location = firstOutOfOrder;

         do {
            list[location] = list[location-1];
            location--;
         } while (location > 0 && list[location-1]  > temp);

         list[location] = temp;
      }
}
```

Insertion Sort (cont'd.)

- List of length n
  - About $(n^2 + 3n - 4) / 4$ key comparisons
  - About $n(n - 1) / 4$ item assignments
- If $n = 1000$
  - 250,000 key comparisons
  - 250,000 item assignments

Binary Search

- Much faster than a sequential search
- List must be sorted
- “Divide and conquer”
- Compare search item with middle element
  - If less than middle: search only upper half of list
  - If more than middle: search only lower half of list

Binary Search (cont'd.)

To find element 75 in the array:

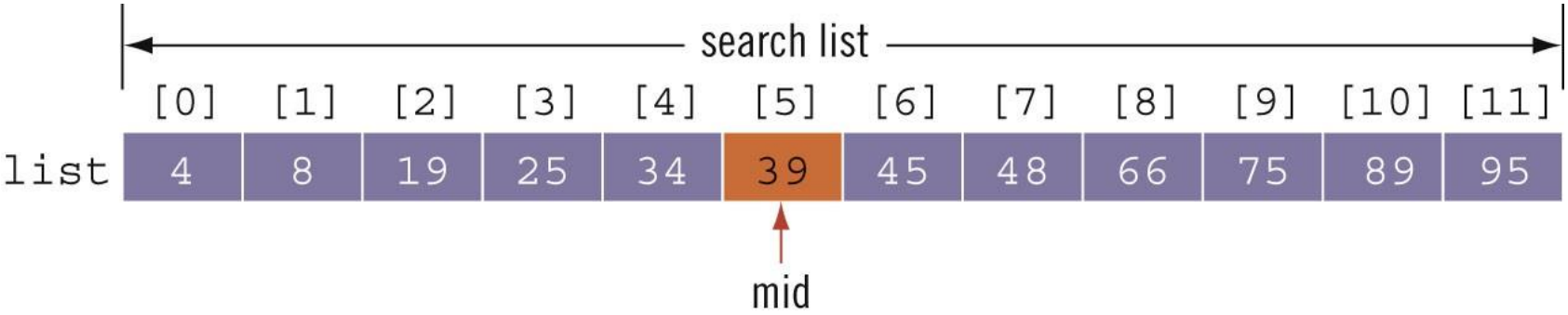


FIGURE 16-14 Search list, `list[0]...list[11]`

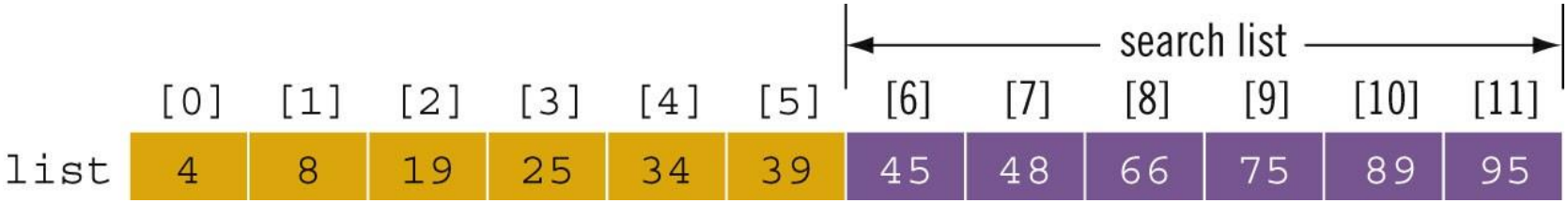


FIGURE 16-15 Search list, `list[6]...list[11]`

Example

Write a function and a driver program to test the binary search algorithm.

Binary Search

```
int binarySearch(int list[], int listSize, int searchValue) {
      bool found = false;
      int first=0, last=listSize-1, mid;

      while (first <= last && !found) {
         mid = (first + last) / 2;
         if (searchValue == list[mid])
            found = true;
         else
            if (searchValue > list[mid])
               first = mid+1;
            else
               last = mid -1;
      }
      if (found)
          return mid;
      else
          return -1;
}
```

Performance of Binary Search

- If *L* is a sorted list of size 1024
  - Every iteration of the `while` loop cuts the size of the search list by half
  - At most, 11 iterations to determine whether *x* is in *L*
  - Binary search will make 22 comparisons at most

- If L has 2^{20} = 1,048,576 elements
  - Binary search makes 42 item comparisons at most

Performance of Binary Search (cont'd.)

- For a sorted list of length *n*:
  - Maximum number comparisons is $2\log_2 n + 2$

`vector` type (class)

- Only a fixed number of elements can be stored in an array
- Inserting and removing elements causes shifting of remaining elements
- `vector` type implements a list
- A Vector variable is also called
  - `vector` container
  - `vector`
  - `vector object`
  - `object`

vector type (class) (cont'd.)

TABLE 16-1 Various Ways to Declare and Initialize a vector Object

Statement	Effect
`vector<elemType> vecList;`	Creates the empty **vector** object **vecList** without any elements.
`vector<elemType> vecList(otherVecList);`	Creates the **vector** object **vecList** and initializes **vecList** to the elements of the vector **otherVecList**. **vecList** and **otherVecList** are of the same type.
`vector<elemType> vecList(size);`	Creates the **vector** object **vecList** of size **size**. **vecList** is initialized using the default values.
`vector<elemType> vecList(n, elem);`	Creates the **vector** object **vecList** of size **n**. **vecList** is initialized using **n** copies of the element **elem**.

vector type (class) (cont'd.)

TABLE 16-2 Operations on a vector Object

Expression	Effect
vecList.**at**(index)	Returns the element at the position specified by **index**.
vecList[index]	Returns the element at the position specified by **index**.
vecList.**front**()	Returns the first element. (Does not check whether the object is empty.)
vecList.**back**()	Returns the last element. (Does not check whether the object is empty.)
vecList.**clear**()	Deletes all elements from the object.
vecList.**push_back**(elem)	A copy of **elem** is inserted into **vecList** at the end.
vecList.**pop_back**()	Delete the last element of **vecList**.

vector type (class) (cont'd.)

TABLE 16-2 Operations on a vector Object (continued)

Expression	Effect
vecList.**empty**()	Returns true if the object vecList is empty and false otherwise.
vecList.**size**()	Returns the number of elements currently in the object vecList. The value returned is an unsigned int value.
vecList.**max_size**()	Returns the maximum number of elements that can be inserted into the object vecList.

Example

Redo the sequential search example using a vector instead of an array.

Example - Vectors

```
#include <iostream>
#include <cstdlib>
#include <vector>
using namespace std;

const int LIST_SIZE = 20;

void loadList(vector<int> &list);
void printList(vector<int>);
int seqSearch(vector<int> list, int searchValue);

int main()
{
   vector<int> list(LIST_SIZE);
   int searchValue;
```

Must include the <vector> header

Must include & for reference parameters

Must specify the data type of the vector's elements

Example – Vectors (Cont.)

```
    loadList(list);
    printList(list);

    cout << "Value to search for?: ";
    cin >> searchValue;

    int pos = seqSearch(list, searchValue);

    if (pos == -1)
       cout << "Element " << searchValue
            << " is not in the array." << endl;
    else
       cout << "Found element " << searchValue
            << " at position " << pos << endl;

     return 0;
}
```

Example – Vectors (Cont.)

```
void loadList(vector<int> &list)
{
   for (int i=0; i<list.size(); i++)
      list[i] = 10 + rand()%90;
}


void printList(vector<int> list)
{
   for (int i=0; i<list.size(); i++)
      cout << list[i] << " ";
   cout << endl;
}
```

Use the size() method to get the size of the vector

Example – Vectors (Cont.)

```
int seqSearch(vector<int> list, int searchValue) {
   int pos = 0;
   bool found = false;

   while (pos<list.size() && !found) {

      if (list[pos] == searchValue)
         found = true;
      else
         pos++;
   }

   if (found)
      return pos;
   else
      return -1;
}
```

Example – Election Results

Write a program to:

1. load candidates names into a string candidatesName array from a candData.txt file,
2. sort the candidatesName array,
3. load the votes data into a parallel votesByRegion integer array from a votesData.txt file,
4. accumulate the total votes for each candidate into a third parallel totalVotes integer array, and
5. display the elections results as shown on Page 1034.

Example – Election Results (P. 1034)

```
#include <iostream>
#include <fstream>
#include <string>
#include <iomanip>

using namespace std;

const int NO_OF_CANDIDATES = 6;
const int NO_OF_REGIONS = 4;

void printHeading();

void initialize(int vbRegion[][NO_OF_REGIONS], int tVotes[],
                int noOfRows);

void getCandidatesName(ifstream& inp, string cNames[],
                       int noOfRows);
```

Example – Election Results (Cont.)

```
void sortCandidatesName(string cNames[], int noOfRows);

int binSearch(string cNames[], int noOfRows, string name);

void processVotes(ifstream& inp, string cNames[],
                  int vbRegion[][NO_OF_REGIONS],
                  int noOfRows);

void addRegionsVote(int vbRegion[][NO_OF_REGIONS],
                    int tVotes[], int noOfRows);

void printResults(string cNames[],
                  int vbRegion[][NO_OF_REGIONS],
                  int tVotes[], int noOfRows);
```

Example – Election Results (Cont.)

```
int main() {
   string candidatesName[NO_OF_CANDIDATES];
   int votesByRegion[NO_OF_CANDIDATES][NO_OF_REGIONS];
   int totalVotes[NO_OF_CANDIDATES];
   ifstream infile;

   infile.open("candData.txt");
   if (!infile) {
       cout << "Input file (candData.txt) does "
            << "not exit." << endl;
       return 1;
   }

   getCandidatesName(infile, candidatesName, NO_OF_CANDIDATES);
   sortCandidatesName(candidatesName, NO_OF_CANDIDATES);

   infile.close();
   infile.clear();
```

Example – Election Results (Cont.)

```
    infile.open("voteData.txt");
    if (!infile) {
        cout << "Input file (voteData.txt) does "
             << "not exist." << endl;
        return 1;
    }

    initialize(votesByRegion, totalVotes, NO_OF_CANDIDATES);
    processVotes(infile, candidatesName,
                 votesByRegion, NO_OF_CANDIDATES);
    addRegionsVote(votesByRegion, totalVotes, NO_OF_CANDIDATES);

    printHeading();
    printResults(candidatesName,votesByRegion,
                 totalVotes, NO_OF_CANDIDATES);

    return 0;
}
```

Example – Election Results (Cont.)

```
void initialize(int vbRegion[][NO_OF_REGIONS], int tVotes[],
                int noOfRows)
{
    int i, j;

    for (i = 0; i < noOfRows; i++)
        for (j = 0; j < NO_OF_REGIONS; j++)
            vbRegion[i][j] = 0;

    for (i = 0; i < noOfRows; i++)
        tVotes[i] = 0;
}
```

Example – Election Results (Cont.)

```
void getCandidatesName(ifstream& inp, string cNames[],
                       int noOfRows) {

    for (int i = 0; i < noOfRows; i++)
        inp >> cNames[i];

}
```

Example – Election Results (Cont.)

```
void sortCandidatesName(string cNames[], int noOfRows)
{
    int firstOutOfOrder, location;
    string temp;

    for (firstOutOfOrder = 1; firstOutOfOrder < noOfRows;
                              firstOutOfOrder++)
      if (cNames[firstOutOfOrder] < cNames[firstOutOfOrder - 1])
      {
           temp = cNames[firstOutOfOrder];
           location = firstOutOfOrder;

           do {
                cNames[location] = cNames[location - 1];
                location--;
           } while (location > 0 && cNames[location - 1] > temp);

           cNames[location] = temp;
      }
}
```

Example – Election Results (Cont.)

```
int binSearch(string cNames[], int noOfRows, string name) {
    int first, last, mid;
    bool found;

    first = 0;
    last = noOfRows - 1;
    found = false;

    while (!found && first <= last) {

        mid = (first + last) / 2;

        if (cNames[mid] == name)
            found = true;
        else if (cNames[mid] > name)
            last = mid - 1;
        else
            first = mid + 1;
    }
    return ((found)?mid:-1);
}
```

Example – Election Results (Cont.)

```
void processVotes(ifstream& inp, string cNames[],
                  int vbRegion[][NO_OF_REGIONS],
                  int noOfRows)
{
    string candName;
    int region, noOfVotes,loc;

    inp >> candName >> region >> noOfVotes;

    while (inp)
    {
        loc =  binSearch(cNames, noOfRows, candName);

        if (loc != -1)
            vbRegion[loc][region - 1] += noOfVotes;

        inp >> candName >> region >> noOfVotes;
    }
}
```

Example – Election Results (Cont.)

```
void addRegionsVote(int vbRegion[][NO_OF_REGIONS],
                    int tVotes[], int noOfRows){
    int i, j;

    for (i = 0; i < noOfRows; i ++)
        for (j = 0; j < NO_OF_REGIONS; j++)
            tVotes[i] = tVotes[i] + vbRegion[i][j];
}

void printHeading(){
    cout << "      ---------------Election Results-----------"
         << "----" << endl << endl;
    cout << "Candidate                    Votes" << endl;
    cout << "Name        Region1   Region2   Region3   "
         << "Region4    Total" << endl;
    cout << "---------   -------   -------   -------   "
         << "-------   ------" << endl;
}
```

Example – Election Results (Cont.)

```
void printResults(string cNames[],
                  int vbRegion[][NO_OF_REGIONS],
                  int tVotes[], int noOfRows)
{
    int i, j;
    int largestVotes = 0;
    int winLoc = 0;
    int sumVotes = 0;

    for (i = 0; i < noOfRows; i++)
    {
        if (largestVotes < tVotes[i])
        {
            largestVotes = tVotes[i];
            winLoc = i;
        }
```

Example – Election Results (Cont.)

```
        sumVotes = sumVotes + tVotes[i];

        cout << left;
        cout << setw(9) << cNames[i] << "   ";
        cout << right;
        for (j = 0; j < NO_OF_REGIONS; j++)
            cout << setw(8) << vbRegion[i][j] << "   ";
        cout << setw(6) << tVotes[i] << endl;
    }

    cout << endl << endl << "Winner: " << cNames[winLoc]
         << ", Votes Received: " << tVotes[winLoc]
         << endl << endl;
    cout << "Total votes polled: " << sumVotes << endl;
}
```

Summary

- List
  - Set of elements of the same type
- Sequential search
  - Searches each element until item is found
- Sorting algorithms
  - Bubble sort
  - Insertion sort

Summary (cont'd.)

- Binary search
  - Much faster than sequential search
  - Requires that the list is sorted
- `vector` type
  - Implements a list
  - Can increase/decrease in size during program execution
  - Must specify the type of object the vector stores

Questions?