Week 9 Lecture Notes: Searching and Sorting Arrays Objectives

Concepts covered in this lesson:

- Introduction to Search Algorithms
- Linear search
- Binary search
- Problem Solving and Program Design
- Introduction to Sorting Algorithms
- Sorting and Searching vectors

Introduction to Search Algorithms

- Search: locate an item in a list of information

- Two algorithms we will examine:
  - Linear search
  - Binary search

Linear Search

- Also called the sequential search
- Starting at the first element, this algorithm sequentially steps through an array examining each element until it locates the value it is searching for.

Linear Search - Example

- Array `numlist` contains:

17	23	5	11	2	29	3

- Searching for the the value `11`, linear search examines `17, 23, 5,` and `11`
- Searching for the the value `7`, linear search examines `17, 23, 5, 11, 2, 29,` and `3`

Linear Search

- Algorithm:

set found to false; set position to –1; set index to 0
while index < number of elts. and found is false
if list[index] is equal to search value
found = true
position = index
end if
add 1 to index
end while
return position

A Linear Search Function

```
int linearSearch(int arr[], int size, int value)
{
   int index = 0;       // Used as a subscript to search the array
   int position = -1;   // To record the position of search value
   bool found = false; // Flag to indicate if value was found

   while (index < size && !found)
   {
      if (arr[index] == value) // If the value is found
      {
         found = true; // Set the flag
         position = index; // Record the value's subscript
      }
      index++; // Go to the next element
   }
return position; // Return the position, or -1
}
```

Pr8-1.cpp - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
<global>
Management
Projects
Workspace
Start here
Pr8-1.cpp
1 // This program demonstrates the linear search algorithm.
2 #include <iostream>
3 using namespace std;
4
5 // Function prototype
6 int linearSearch(const int[], int, int);
7
8 int main()
9 {
10 const int SIZE = 5;
11 int tests[SIZE] = { 87, 75, 98, 100, 82 };
12 int results;
13
14 // Search the array for 100.
15 results = linearSearch(tests, SIZE, 100);
16
17 // If linearSearch returned -1, then 100 was not found.
18 if (results == -1)
19 cout << "You did not earn 100 points on any test\n";
20 else
21 {
22 // Otherwise results contains the subscript of
23 // the first 100 in the array.
24 cout << "You earned 100 points on test ";
25 cout << (results + 1) << endl;
26 }
27 return 0;
28 }
29
30 //***
31 // The linearSearch function performs a linear search on an *
32 // integer array. The array arr, which has a maximum of size *
33 // elements, is searched for the number stored in value. If the *
34 // number is found, its array subscript is returned. Otherwise, *
35 // -1 is returned indicating the value was not in the array. *
36 //***
37 int linearSearch(const int arr[], int size, int value)
38 {
39 int index = 0; // Used as a subscript to search array
Logs & others
Code::Blocks
Search results
Cccc
Build log
CppCheck/Vera++
CppCheck/Vera++ messages
Cscope
Debugger
DoxyBlocks
Fortran info
Closed files list
Thread search

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global>

Management

Projects

Workspace

Start here | Pr8-1.cpp

```
 9  {
10     const int SIZE = 5;
11     int tests[SIZE] = { 87, 75, 98, 100, 82 };
12     int results;
13
14     // Search the array for 100.
15     results = linearSearch(tests, SIZE, 100);
16
17     // If linearSearch returned -1, then 100 was not found.
18     if (results == -1)
19        cout << "You did not earn 100 points on any test\n";
20     else
21     {
22        // Otherwise results contains the subscript of
23        // the first 100 in the array.
24        cout << "You earned 100 points on test ";
25        cout << (results + 1) << endl;
26     }
27     return 0;
28  }
29
30  //***************************************************************
31  // The linearSearch function performs a linear search on an       *
32  // integer array. The array arr, which has a maximum of size      *
33  // elements, is searched for the number stored in value. If the   *
34  // number is found, its array subscript is returned. Otherwise,   *
35  // -1 is returned indicating the value was not in the array.      *
36  //***************************************************************
37  int linearSearch(const int arr[], int size, int value)
38  {
39     int index = 0;       // Used as a subscript to search array
40     int position = -1;   // To record position of search value
41     bool found = false;  // Flag to indicate if the value was found
42
43     while (index < size && !found)
44     {
45        if (arr[index] == value)  // If the value is found
46        {
47           found = true;          // Set the flag
```

"F:\BlackUSB\evc\2022\Fall\CS75\ProgrammingExamples\Week9ProgrammingExamplesChapter 08...

```
You earned 100 points on test 4

Process returned 0 (0x0)   execution time : 0.739 s
Press any key to continue.
```

Logs & others

Code::Blocks | Search results | Cccc | Build log | CppCheck/Vera++ | CppCheck/Vera++ messages | Cscope | Debugger | DoxyBlocks | Fortran info | Closed files list | Thread search

```
Executing: '"C:\Program Files\CodeBlocks/cb_console_runner.exe" "F:\BlackUSB\evc\2022\Fall\CS75\ProgrammingExamples\Week9ProgrammingExamplesChapter 08\Pr8-1.exe"' (in 'F:\BlackUSB\evc\2022\Fall\CS75\ProgrammingExamples\Week9P
08')
Set variable: PATH=C:\Program Files\CodeBlocks\MinGW\bin;C:\Program Files\CodeBlocks\MinGW;C:\Program Files (x86)\Common Files\Oracle\Java\javapath;C:\ProgramData\Oracle\Java\javapath;c:\Program Files (x86)\Intel\iCLS Client;
Client;C:\WINDOWS\System32;C:\WINDOWS;C:\WINDOWS\System32\wbem;C:\WINDOWS\System32\WindowsPowerShell\v1.0;C:\Program Files\Intel\Intel(R) Management Engine Components\DAL;C:\Program Files\Intel\Intel(R) Management Engine Comp
(x86)\Intel\Intel(R) Management Engine Components\DAL;C:\Program Files (x86)\Intel\Intel(R) Management Engine Components\IPT;C:\Program Files (x86)\Common Files\Acronis\SnapAPI;C:\Program Files (x86)\Acronis\TrueImageHome;C:\
Server\110\Tools\Binn;C:\Program Files\Common Files\NeatReceipts\Drivers\M12;C:\Program Files (x86)\nodejs;C:\usr\bin;C:\Program Files\dotnet;C:\Program Files\Microsoft SQL Server\130\Tools\Binn;C:\Program Files\Common Files\
Files\PuTTY;C:\Program Files\Microsoft SQL Server\120\Tools\Binn;C:\Program Files\Java\jdk1.8.0_171\bin;C:\Users\garry\AppData\Roaming\npm
```

F:\BlackUSB\evc\2022\Fall\CS75\ProgrammingExamples\Week9ProgrammingExamplesChapter 08\Pr8-1.cpp | C/C++ | Windows (CR+LF) | WINDOWS-1252 | Line 1, Col 1, Pos 0 | Insert | Read

Linear Search - Tradeoffs

- Benefits:
  - Easy algorithm to understand
  - Array can be in any order

- Disadvantages:
  - Inefficient (slow): for array of N elements, examines N/2 elements on average for value in array, N elements for value not in array

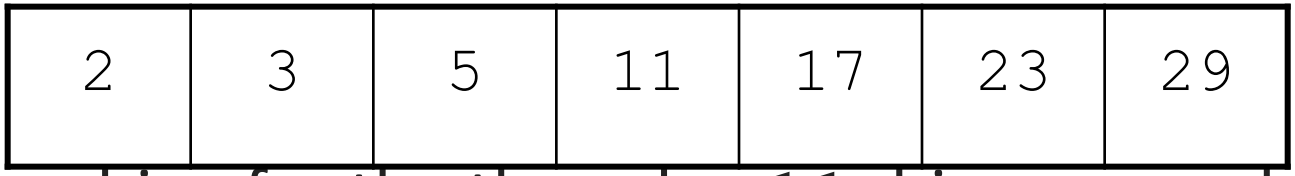
Linear Search

Requires array elements to be in order

1. Divides the array into three sections:
   - middle element
   - elements on one side of the middle element
   - elements on the other side of the middle element
2. If the middle element is the correct value, done. Otherwise, go to step 1. using only the half of the array that may contain the correct value.
3. Continue steps 1. and 2. until either the value is found or there are no more elements to examine

Binary Search - Example

- Array `numlist2` contains:



- Searching for the the value `11`, binary search examines `11` and stops
- Searching for the the value `7`, linear search examines `11, 3, 5,` and stops

Binary Search

Set first to 0
Set last to the last subscript in the array
Set found to false
Set position to -1
While found is not true and first is less than or equal to last
    Set middle to the subscript half-way between array[first] and array[last].
    If array[middle] equals the desired value
        Set found to true
        Set position to middle
    Else If array[middle] is greater than the desired value
        Set last to middle - 1
    Else
        Set first to middle + 1
    End If.
End While.
Return position.

A Binary Search Function

```
int binarySearch(int array[], int size, int value)
{
   int first = 0,              // First array element
       last = size - 1,        // Last array element
       middle,                 // Mid point of search
       position = -1;          // Position of search value
   bool found = false;         // Flag

   while (!found && first <= last)
   {
      middle = (first + last) / 2;     // Calculate mid point
      if (array[middle] == value)      // If value is found at mid
      {
         found = true;
         position = middle;
      }
      else if (array[middle] > value)  // If value is in lower half
         last = middle - 1;
      else
         first = middle + 1;           // If value is in upper half
   }
   return position;
}
```

Pr8-2.cpp - Code::Blocks 20.03

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global> | main() : int

Management

Projects | Workspace

Start here | Pr8-1.cpp | Pr8-5.cpp | Pr8-2.cpp

```
1   // This program demonstrates the binarySearch function, which
2   // performs a binary search on an integer array.
3   #include <iostream>
4   using namespace std;
5
6   // Function prototype
7   int binarySearch(const int [], int, int);
8   const int SIZE = 20;
9
10  int main()
11  {
12     // Array with employee IDs sorted in ascending order.
13     int idNums[SIZE] = {101, 142, 147, 189, 199, 207, 222,
14                         234, 289, 296, 310, 319, 388, 394,
15                         417, 429, 447, 521, 536, 600};
16     int results;   // To hold the search results
17     int empID;     // To hold an employee ID
18
19     // Get an employee ID to search for.
20     cout << "Enter the employee ID you wish to search for: ";
21     cin >> empID;
22
23     // Search for the ID.
24     results = binarySearch(idNums, SIZE, empID);
25
26     // If results contains -1 the ID was not found.
27     if (results == -1)
28        cout << "That number does not exist in the array.\n";
29     else
30     {
31        // Otherwise results contains the subscript of
32        // the specified employee ID in the array.
33        cout << "That ID is found at element " << results;
34        cout << " in the array.\n";
35     }
36     return 0;
37  }
38
39  //***************************************************************
```

"F:\BlackUSB\evc\2022\Fall\CS75\ProgrammingExamples\Week9ProgrammingExamplesChapter 08...

```
Enter the employee ID you wish to search for: 310
That ID is found at element 10 in the array.

Process returned 0 (0x0)   execution time : 14.845 s
Press any key to continue.
```

Logs & others

Code::Blocks | Search results | Cccc | Build log | CppCheck/Vera++ | CppCheck/Vera++ messages | Cscope | Debugger | DoxyBlocks | Fortran info | Closed files list | Thread search

```
Executing: '"C:\Program Files\CodeBlocks/cb_console_runner.exe" "F:\BlackUSB\evc\2022\Fall\CS75\ProgrammingExamples\Week9ProgrammingExamplesChapter 08\Pr8-2.exe"' (in 'F:\BlackUSB\evc\2022\Fall\CS75\ProgrammingExamples\Week9Progr
08')
Set variable: PATH=C:\Program Files\CodeBlocks\MinGW\bin;C:\Program Files\CodeBlocks\MinGW;C:\Program Files (x86)\Common Files\Oracle\Java\javapath;C:\ProgramData\Oracle\Java\javapath;c:\Program Files (x86)\Intel\iCLS Client;c:\P
Client;C:\WINDOWS\System32;C:\WINDOWS;C:\WINDOWS\System32\wbem;C:\WINDOWS\System32\WindowsPowerShell\v1.0;C:\Program Files\Intel\Intel(R) Management Engine Components\DAL;C:\Program Files\Intel\Intel(R) Management Engine Componen
(x86)\Intel\Intel(R) Management Engine Components\DAL;C:\Program Files (x86)\Intel\Intel(R) Management Engine Components\IPT;C:\Program Files (x86)\Common Files\Acronis\SnapAPI;C:\Program Files (x86)\Acronis\TrueImageHome;C:\Prog
Server\110\Tools\Binn;C:\Program Files\Common Files\NeatReceipts\Drivers\M12;C:\Program Files (x86)\nodejs;C:\usr\bin;C:\Program Files\dotnet;C:\Program Files\Microsoft SQL Server\130\Tools\Binn;C:\Program Files\Common Files\Auto
Files\PuTTY;C:\Program Files\Microsoft SQL Server\120\Tools\Binn;C:\Program Files\Java\jdk1.8.0_171\bin;C:\Users\garry\AppData\Roaming\npm
```

F:\BlackUSB\evc\2022\Fall\CS75\ProgrammingExamples\Week9ProgrammingExamplesChapter 08\Pr8-2.cpp | C/C++ | Windows (CR+LF) | WINDOWS-1252 | Line 23, Col 25, Pos 743 | Insert | Read/Write

Binary Search - Tradeoffs

- Benefits:
  - Much more efficient than linear search. For array of N elements, performs at most $log_2 N$ comparisons
- Disadvantages:
  - Requires that array elements be sorted

8.3

Introduction to Sorting Algorithms

Introduction to Sorting Algorithms

- Sort: arrange values into an order:
  - Alphabetical
  - Ascending numeric
  - Descending numeric

- Two algorithms considered here:
  - Bubble sort
  - Selection sort

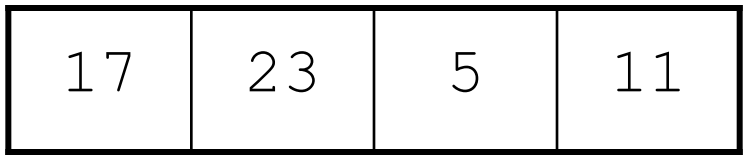
Bubble Sort

Concept:

- Compare 1st two elements
  - If out of order, exchange them to put in order
- Move down one element, compare 2nd and 3rd elements, exchange if necessary. Continue until end of array.
- Pass through array again, exchanging as necessary
- Repeat until pass made with no exchanges

Example – First Pass

Array `numlist3` contains:



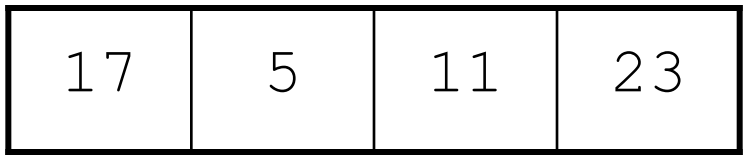
compare values `17` and `23` – in correct order, so no exchange

compare values `23` and `5` – not in correct order, so exchange them

compare values `23` and `11` – not in correct order, so exchange them

Example – Second Pass

After first pass, array `numlist3` contains:



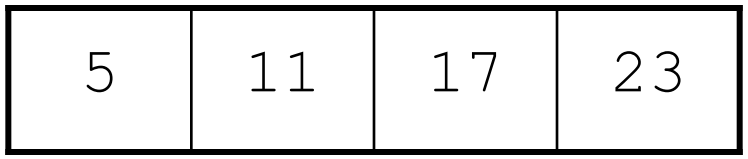
compare values `17` and `5` – not in correct order, so exchange them

compare values `17` and `11` – not in correct order, so exchange them

compare values `17` and `23` – in correct order, so no exchange

Example – Third Pass

After second pass, array `numlist3` contains:



compare values `5` and `11` – in correct order, so no exchange

compare values `11` and `17` – in correct order, so no exchange

compare values `17` and `23` – in correct order, so no exchange

No exchanges, so array is in order

Bubble Sort - Tradeoffs

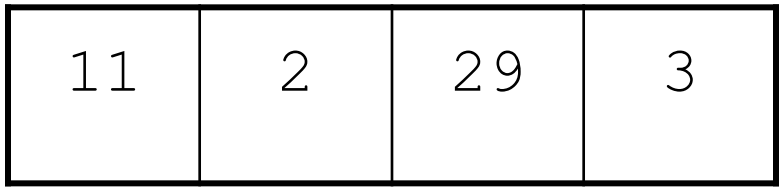
- Benefit:
  - Easy to understand and implement
- Disadvantage:
  - Inefficient: slow for large arrays

Selection Sort

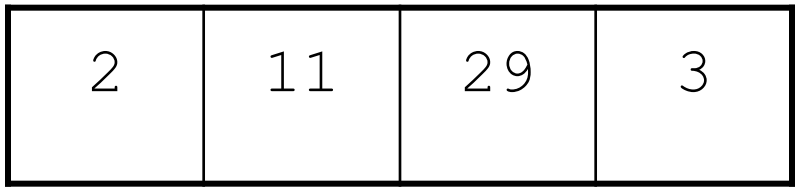
- Concept for sort in ascending order:
  - Locate smallest element in array. Exchange it with element in position 0
  - Locate next smallest element in array. Exchange it with element in position 1.
  - Continue until all elements are arranged in order

Selection Sort - Example

Array `numlist` contains:



1. Smallest element is `2`. Exchange `2` with element in 1st position in array:



Example (Continued)

2. Next smallest element is `3`. Exchange `3` with element in 2nd position in array:

2	3	29	11

3. Next smallest element is `11`. Exchange `11` with element in 3rd position in array:

2	3	11	29

Start here | Pr8-1.cpp | Pr8-5.cpp

```
1  // This program demonstrates the Selection Sort algorithm.
2  #include <iostream>
3  using namespace std;
4
5  // Function prototypes
6  void selectionSort(int[], int);
7  void swap(int &, int &);
8
9  int main()
10 {
11    const int SIZE = 6;
12
13    // Array of unsorted values
14    int values[SIZE] = { 6, 1, 5, 2, 4, 3 };
15
16    // Display the unsorted array.
17    cout << "The unsorted values:\n";
18    for (auto element : values)
19       cout << element << " ";
20    cout << endl;
21
22    // Sort the array.
23    selectionSort(values, SIZE);
24
25    // Display the sorted array.
26    cout << "The sorted values:\n";
27    for (auto element : values)
28       cout << element << " ";
29    cout << endl;
30
31    return 0;
32 }
33
34 //*************************************************************
35 // The selectionSort function sorts an int array in ascending order. *
36 //*************************************************************
37 void selectionSort(int array[], int size)
38 {
39    int minIndex, minValue;
```

Logs & others

Code::Blocks | Search results | Cccc | Build log | CppCheck/Vera++ | CppCheck/Vera++ messages | Cscope | Debugger | DoxyBlocks | Fortran info | Closed files list | Thread search

```
Set variable: PATH=C:\Program Files\CodeBlocks\MinGW\bin;C:\Program Files\CodeBlocks\MinGW;C:\Program Files (x86)\Common Files\Oracle\Java\javapath;C:\ProgramData\Oracle\Java\javapath;c:\Program Files (x86)\Intel\iCLS Client;c:\Program Files\Int
Client;C:\WINDOWS\System32;C:\WINDOWS;C:\WINDOWS\System32\wbem;C:\WINDOWS\System32\WindowsPowerShell\v1.0;C:\Program Files\Intel\Intel(R) Management Engine Components\DAL;C:\Program Files\Intel\Intel(R) Management Engine Components\IPT;C:\Progra
(x86)\Intel\Intel(R) Management Engine Components\DAL;C:\Program Files (x86)\Intel\Intel(R) Management Engine Components\IPT;C:\Program Files (x86)\Common Files\Acronis\SnapAPI;C:\Program Files (x86)\Acronis\TrueImageHome;C:\Program Files\Micros
Server\110\Tools\Binn;C:\Program Files\Common Files\NeatReceipts\Drivers\M12;C:\Program Files (x86)\nodejs;C:\usr\bin;C:\Program Files\dotnet;C:\Program Files\Microsoft SQL Server\130\Tools\Binn;C:\Program Files\Common Files\Autodesk Shared;C:\P
Files\PuTTY;C:\Program Files\Microsoft SQL Server\120\Tools\Binn;C:\Program Files\Java\jdk1.8.0_171\bin;C:\Users\garry\AppData\Roaming\npm
Process terminated with status -1073741510 (3 minute(s), 48 second(s))
```

Pr8-5.cpp - Code::Blocks 20.03

Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

Start here | Pr8-1.cpp | Pr8-5.cpp

```
29      cout << endl;
30
31      return 0;
32  }
33
34  //***************************************************************
35  // The selectionSort function sorts an int array in ascending order. *
36  //***************************************************************
37  void selectionSort(int array[], int size)
38  {
39     int minIndex, minValue;
40
41     for (int start = 0; start < (size - 1); start++)
42     {
43        minIndex = start;
44        minValue = array[start];
45        for (int index = start + 1; index < size; index++)
46        {
47           if (array[index] < minValue)
48           {
49              minValue = array[index];
50              minIndex = index;
51           }
52        }
53        swap(array[minIndex], array[start]);
54     }
55  }
56
57  //*************************************************
58  // The swap function swaps a and b in memory.      *
59  //*************************************************
60  void swap(int &a, int &b)
61  {
62     int temp = a;
63     a = b;
64     b = temp;
65  }
66
67
```

"F:\BlackUSB\evc\2022\Fall\CS75\ProgrammingExamples\Week9ProgrammingExamplesChapter 08...

```
The unsorted values:
6 1 5 2 4 3
The sorted values:
1 2 3 4 5 6

Process returned 0 (0x0)   execution time : 0.119 s
Press any key to continue.
```

Logs & others

Code::Blocks | Search results | Cccc | Build log | CppCheck/Vera++ | CppCheck/Vera++ messages | Cscope | Debugger | DoxyBlocks | Fortran info | Closed files list | Thread search

```
Executing: '"C:\Program Files\CodeBlocks/cb_console_runner.exe" "F:\BlackUSB\evc\2022\Fall\CS75\ProgrammingExamples\Week9ProgrammingExamplesChapter 08\Pr8-5.exe"' (in 'F:\BlackUSB\evc\2022\Fall\CS75\ProgrammingExamples\Week9ProgrammingExamplesChapter 08')
Set variable: PATH=C:\Program Files\CodeBlocks\MinGW\bin;C:\Program Files\CodeBlocks\MinGW;C:\Program Files (x86)\Common Files\Oracle\Java\javapath;C:\ProgramData\Oracle\Java\javapath;c:\Program Files (x86)\Intel\iCLS Client;c:\Program Files\Intel\iCLS Client;C:\WINDOWS\System32;C:\WINDOWS;C:\WINDOWS\System32\wbem;C:\WINDOWS\System32\WindowsPowerShell\v1.0;C:\Program Files\Intel\Intel(R) Management Engine Components\DAL;C:\Program Files\Intel\Intel(R) Management Engine Components\IPT;C:\Program Files (x86)\Intel\Intel(R) Management Engine Components\DAL;C:\Program Files (x86)\Intel\Intel(R) Management Engine Components\IPT;C:\Program Files (x86)\Common Files\Acronis\SnapAPI;C:\Program Files (x86)\Acronis\TrueImageHome;C:\Program Files\Microsoft SQL Server\110\Tools\Binn;C:\Program Files\Common Files\NeatReceipts\Drivers\M12;C:\Program Files (x86)\nodejs;C:\usr\bin;C:\Program Files\dotnet;C:\Program Files\Microsoft SQL Server\130\Tools\Binn;C:\Program Files\Common Files\Autodesk Shared Files\PuTTY;C:\Program Files\Microsoft SQL Server\120\Tools\Binn;C:\Program Files\Java\jdk1.8.0_171\bin;C:\Users\garry\AppData\Roaming\npm
```

Selection Sort - Tradeoffs

- Benefit:
  - More efficient than Bubble Sort, since fewer exchanges
- Disadvantage:
  - May not be as easy as Bubble Sort to understand

Sorting and Searching Vectors

- Sorting and searching algorithms can be applied to vectors as well as arrays
- Need slight modifications to functions to use vector arguments:
  - `vector <type> &` used in prototype
  - No need to indicate vector size – functions can use `size` member function to calculate