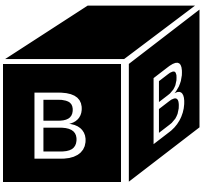


Chapter *16*

Iterative Searching and Sorting

(i < numItm) & (v[i] # srchNum)

```
(i < numItm) & (v[i] # srchNum)
```

- Evaluate the first part.
- Evaluate the second part.
- Perform the AND operation.

Full evaluation of AND expressions

```
(i < numItm) & (v[i] # srchNum)
```

Full evaluation of AND expressions

- Evaluate the first part.
- Evaluate the second part.
- Perform the AND operation.

Short-circuit evaluation of AND expressions

- Evaluate the first part.
- If it is false, skip the second part.
- Otherwise, evaluate the second part.

```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
        OUT i: INTEGER; OUT fnd: BOOLEAN);
BEGIN
    ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
    i := 0;
    WHILE (i < numItm) & (v[i] # srchNum) DO
        INC(i)
    END;
    fnd := i < numItm
END Search;
```

Figure 16.1
The first version of the sequential search algorithm. This is not the most efficient version.

- Best case:
- Worst case:

```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
        OUT i: INTEGER; OUT fnd: BOOLEAN);
BEGIN
    ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
    i := 0;
    WHILE (i < numItm) & (v[i] # srchNum) DO
        INC(i)
    END;
    fnd := i < numItm
END Search;
```

- Best case: 2 comparisons
- Worst case:

```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
        OUT i: INTEGER; OUT fnd: BOOLEAN);
BEGIN
    ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
    i := 0;
    WHILE (i < numItm) & (v[i] # srchNum) DO
        INC(i)
    END;
    fnd := i < numItm
END Search;
```

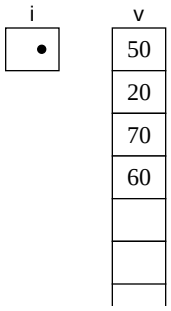
- Best case: 2 comparisons
- Worst case: $2n + 1$ comparisons

```
PROCEDURE Search (VAR v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
        OUT i: INTEGER; OUT fnd: BOOLEAN);
BEGIN
    ASSERT((0 <= numItm) & (numItm < LEN(v)), 20);
    v[numItm] := srchNum;
    i := 0;
    WHILE v[i] # srchNum DO
        INC(i)
    END;
    fnd := i < numItm
END Search;
```

Figure 16.2
An efficient version of the sequential search algorithm.

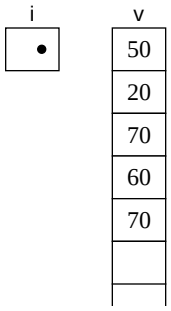
```
PROCEDURE Search (VAR v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
BEGIN
   ASSERT((0 <= numItm) & (numItm < LEN(v)), 20);
   v[numItm] := srchNum;
   i := 0;
   WHILE v[i] # srchNum DO
      INC(i)
   END;
   fnd := i < numItm
END Search;
```

numItm	**srchNum**	**fnd**
4	70	



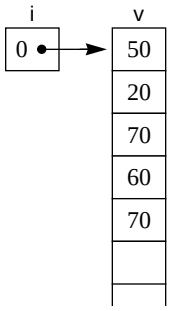
```
PROCEDURE Search (VAR v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
BEGIN
   ASSERT((0 <= numItm) & (numItm < LEN(v)), 20);
   v[numItm] := srchNum;
   i := 0;
   WHILE v[i] # srchNum DO
      INC(i)
   END;
   fnd := i < numItm
END Search;
```

numItm	**srchNum**	**fnd**
4	70	



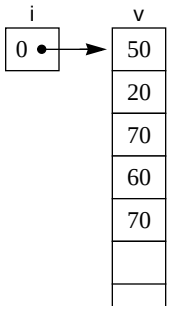
```
PROCEDURE Search (VAR v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
BEGIN
   ASSERT((0 <= numItm) & (numItm < LEN(v)), 20);
   v[numItm] := srchNum;
   i := 0;
   WHILE v[i] # srchNum DO
      INC(i)
   END;
   fnd := i < numItm
END Search;
```

numItm	**srchNum**	**fnd**
4	70	



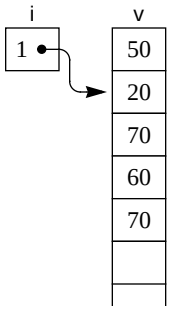
```
PROCEDURE Search (VAR v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
BEGIN
   ASSERT((0 <= numItm) & (numItm < LEN(v)), 20);
   v[numItm] := srchNum;
   i := 0;
   WHILE v[i] # srchNum DO
      INC(i)
   END;
   fnd := i < numItm
END Search;
```

numItm	**srchNum**	**fnd**
4	70	



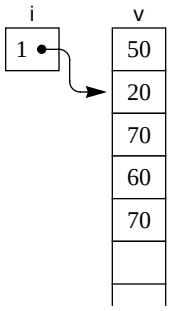
```
PROCEDURE Search (VAR v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
BEGIN
   ASSERT((0 <= numItm) & (numItm < LEN(v)), 20);
   v[numItm] := srchNum;
   i := 0;
   WHILE v[i] # srchNum DO
      INC(i)
   END;
   fnd := i < numItm
END Search;
```

numItm	**srchNum**	**fnd**
4	70	



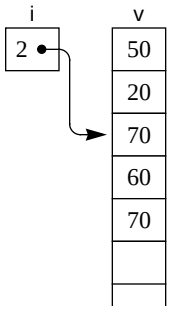
```
PROCEDURE Search (VAR v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
BEGIN
   ASSERT((0 <= numItm) & (numItm < LEN(v)), 20);
   v[numItm] := srchNum;
   i := 0;
   WHILE v[i] # srchNum DO
      INC(i)
   END;
   fnd := i < numItm
END Search;
```

numItm	**srchNum**	**fnd**
4	70	



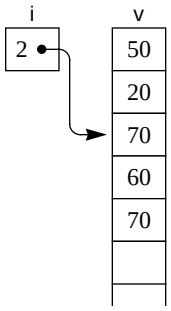
```
PROCEDURE Search (VAR v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
BEGIN
   ASSERT((0 <= numItm) & (numItm < LEN(v)), 20);
   v[numItm] := srchNum;
   i := 0;
   WHILE v[i] # srchNum DO
      INC(i)
   END;
   fnd := i < numItm
END Search;
```

numItm	**srchNum**	**fnd**
4	70	



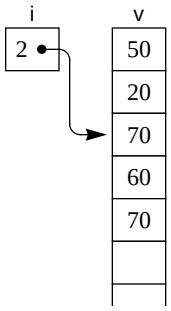
```
PROCEDURE Search (VAR v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
BEGIN
   ASSERT((0 <= numItm) & (numItm < LEN(v)), 20);
   v[numItm] := srchNum;
   i := 0;
   WHILE v[i] # srchNum DO
      INC(i)
   END;
   fnd := i < numItm
END Search;
```

numItm	**srchNum**	**fnd**
4	70	



```
PROCEDURE Search (VAR v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
BEGIN
   ASSERT((0 <= numItm) & (numItm < LEN(v)), 20);
   v[numItm] := srchNum;
   i := 0;
   WHILE v[i] # srchNum DO
      INC(i)
   END;
   fnd := i < numItm
END Search;
```

numItm	**srchNum**	**fnd**
4	70	TRUE



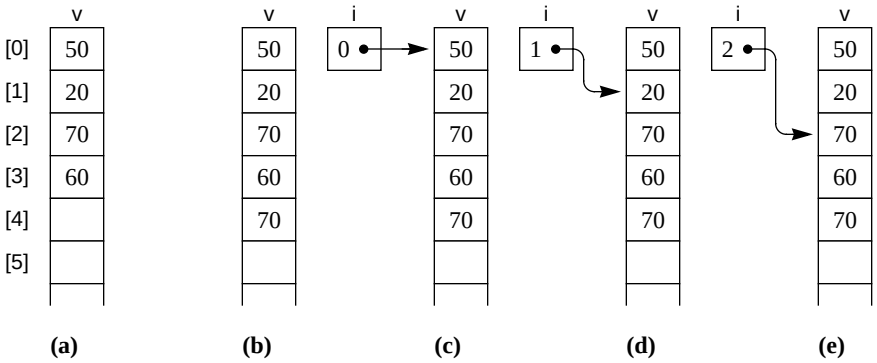
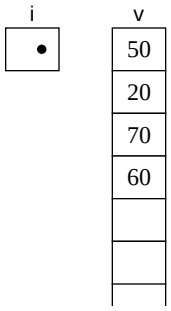


Figure 16.3
A trace of the sequential search algorithm when the value of srchNum, 70, is in the list.

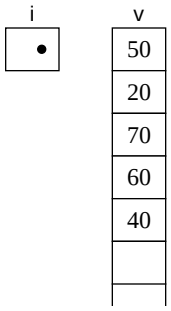
```
PROCEDURE Search (VAR v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
BEGIN
   ASSERT((0 <= numItm) & (numItm < LEN(v)), 20);
   v[numItm] := srchNum;
   i := 0;
   WHILE v[i] # srchNum DO
      INC(i)
   END;
   fnd := i < numItm
END Search;
```

numItm	**srchNum**	**fnd**
4	40	



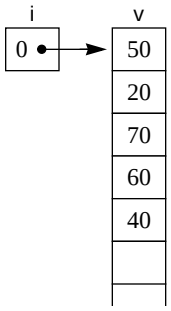
```
PROCEDURE Search (VAR v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
BEGIN
   ASSERT((0 <= numItm) & (numItm < LEN(v)), 20);
   v[numItm] := srchNum;
   i := 0;
   WHILE v[i] # srchNum DO
      INC(i)
   END;
   fnd := i < numItm
END Search;
```

numItm	**srchNum**	**fnd**
4	40	



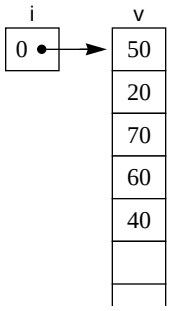
```
PROCEDURE Search (VAR v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
BEGIN
   ASSERT((0 <= numItm) & (numItm < LEN(v)), 20);
   v[numItm] := srchNum;
   i := 0;
   WHILE v[i] # srchNum DO
      INC(i)
   END;
   fnd := i < numItm
END Search;
```

numItm	**srchNum**	**fnd**
4	40	



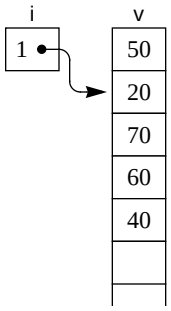
```
PROCEDURE Search (VAR v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
BEGIN
   ASSERT((0 <= numItm) & (numItm < LEN(v)), 20);
   v[numItm] := srchNum;
   i := 0;
   WHILE v[i] # srchNum DO
      INC(i)
   END;
   fnd := i < numItm
END Search;
```

numItm	**srchNum**	**fnd**
4	40	



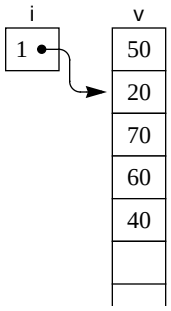
```
PROCEDURE Search (VAR v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
BEGIN
   ASSERT((0 <= numItm) & (numItm < LEN(v)), 20);
   v[numItm] := srchNum;
   i := 0;
   WHILE v[i] # srchNum DO
      INC(i)
   END;
   fnd := i < numItm
END Search;
```

numItm	**srchNum**	**fnd**
4	40	



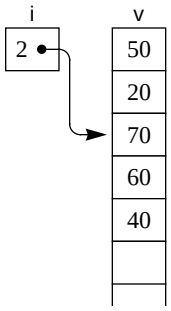
```
PROCEDURE Search (VAR v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
BEGIN
   ASSERT((0 <= numItm) & (numItm < LEN(v)), 20);
   v[numItm] := srchNum;
   i := 0;
   WHILE v[i] # srchNum DO
      INC(i)
   END;
   fnd := i < numItm
END Search;
```

numItm	**srchNum**	**fnd**
4	40	



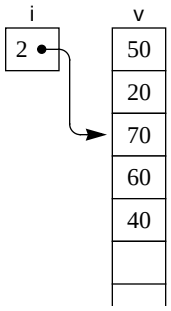
```
PROCEDURE Search (VAR v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
BEGIN
   ASSERT((0 <= numItm) & (numItm < LEN(v)), 20);
   v[numItm] := srchNum;
   i := 0;
   WHILE v[i] # srchNum DO
      INC(i)
   END;
   fnd := i < numItm
END Search;
```

numItm	**srchNum**	**fnd**
4	40	



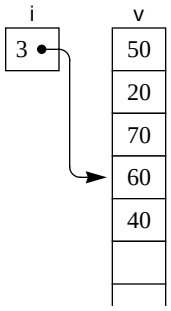
```
PROCEDURE Search (VAR v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
BEGIN
   ASSERT((0 <= numItm) & (numItm < LEN(v)), 20);
   v[numItm] := srchNum;
   i := 0;
   WHILE v[i] # srchNum DO
      INC(i)
   END;
   fnd := i < numItm
END Search;
```

numItm	**srchNum**	**fnd**
4	40	



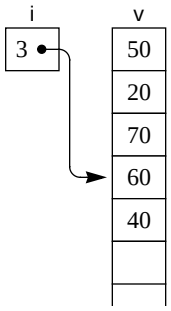
```
PROCEDURE Search (VAR v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
BEGIN
   ASSERT((0 <= numItm) & (numItm < LEN(v)), 20);
   v[numItm] := srchNum;
   i := 0;
   WHILE v[i] # srchNum DO
      INC(i)
   END;
   fnd := i < numItm
END Search;
```

numItm	**srchNum**	**fnd**
4	40	



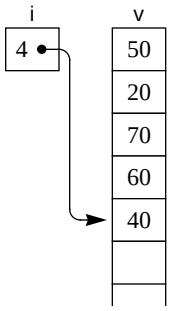
```
PROCEDURE Search (VAR v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
BEGIN
   ASSERT((0 <= numItm) & (numItm < LEN(v)), 20);
   v[numItm] := srchNum;
   i := 0;
   WHILE v[i] # srchNum DO
      INC(i)
   END;
   fnd := i < numItm
END Search;
```

numItm	**srchNum**	**fnd**
4	40	



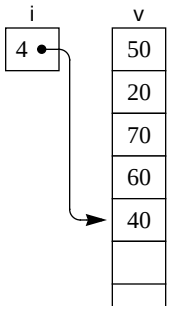
```
PROCEDURE Search (VAR v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
BEGIN
   ASSERT((0 <= numItm) & (numItm < LEN(v)), 20);
   v[numItm] := srchNum;
   i := 0;
   WHILE v[i] # srchNum DO
      INC(i)
   END;
   fnd := i < numItm
END Search;
```

numItm	**srchNum**	**fnd**
4	40	



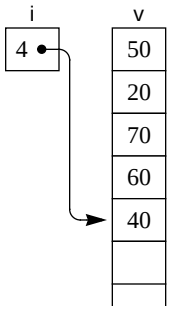
```
PROCEDURE Search (VAR v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
BEGIN
   ASSERT((0 <= numItm) & (numItm < LEN(v)), 20);
   v[numItm] := srchNum;
   i := 0;
   WHILE v[i] # srchNum DO
      INC(i)
   END;
   fnd := i < numItm
END Search;
```

numItm	**srchNum**	**fnd**
4	40	



```
PROCEDURE Search (VAR v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
BEGIN
   ASSERT((0 <= numItm) & (numItm < LEN(v)), 20);
   v[numItm] := srchNum;
   i := 0;
   WHILE v[i] # srchNum DO
      INC(i)
   END;
   fnd := i < numItm
END Search;
```

numItm	**srchNum**	**fnd**
4	40	FALSE



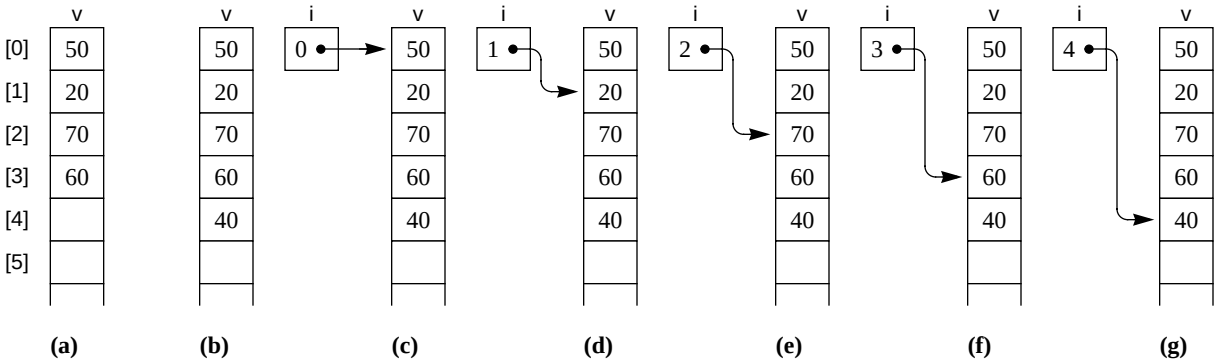


Figure 16.4
A trace of the sequential search algorithm when the value of srchNum, 40, is not in the list.

- Best case: 1 comparison
- Worst case: $n + 1$ comparisons

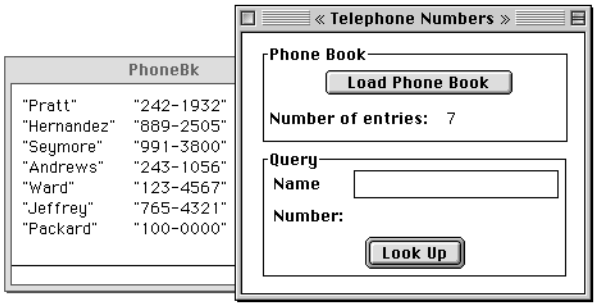
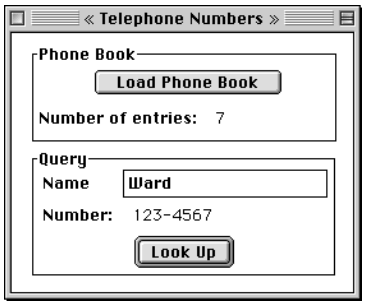


Figure 16.5
A tool dialog box.

```
MENU   "Pbox17"
    "A..."    ""    "StdCmds.OpenToolDialog('Pbox17/Rsrc/DlgA', 'Telephone Numbers')"    ""
END
```



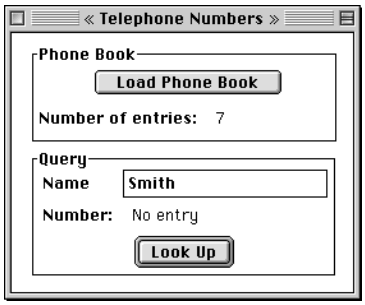


Figure 16.6
Performing a query.

nameList

[0]	Pratt
[1]	Hernandez
[2]	Seymore

numberList

[0]	242-1932
[1]	889-2505
[2]	991-3800

Figure 16.7
The parallel arrays for the phone book in Figure 16.8.

```
MODULE Pbox16A;
    IMPORT Dialog, TextModels, TextControllers, PboxMappers;
    TYPE
        Name = ARRAY 32 OF CHAR;
        Number = ARRAY 16 OF CHAR;
    VAR
        d*: RECORD
            numEntries-: INTEGER;
            name*: Name;
            number-: Number
        END;
    CONST
        maxItems = 1024;
    VAR
        nameList: ARRAY maxItems OF Name;
        numberList: ARRAY maxItems OF Number;
```

Figure 16.8
Using the sequential search to look up a phone number.

```
PROCEDURE LoadBook*;
    VAR
        md: TextModels.Model;
        cn: TextControllers.Controller;
        sc: PboxMappers.Scanner;
        i: INTEGER;
BEGIN
    cn := TextControllers.Focus();
    IF cn # NIL THEN
        md := cn.text; sc.ConnectTo(md);
        i := 0;
        sc.ScanString(nameList[i]); sc.ScanString(numberList[i]);
        WHILE ~sc.eot DO
            INC(i);
            sc.ScanString(nameList[i]); sc.ScanString(numberList[i])
        END;
        d.numEntries := i
    END;
    d.number := "";
    Dialog.Update(d)
END LoadBook;
```

```
PROCEDURE Search (VAR v: ARRAY OF Name; numItm: INTEGER; IN srchName: Name;
        OUT i: INTEGER; OUT fnd: BOOLEAN);
BEGIN
    ASSERT((0 <= numItm) & (numItm < LEN(v)), 20);
    v[numItm] := srchName$;
    i := 0;
    WHILE v[i] # srchName DO
        INC(i)
    END;
    fnd := i < numItm
END Search;
```

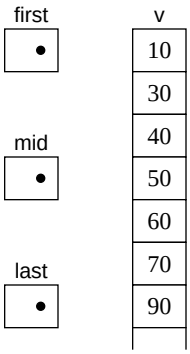
```
    PROCEDURE LookUp*;
        VAR
            j: INTEGER;
            found: BOOLEAN;
    BEGIN
        Search(nameList, d.numEntries, d.name, j, found);
        IF found THEN
            d.number := numberList[j]$
        ELSE
            d.number := "No entry"
        END;
        Dialog.Update(d)
    END LookUp;

BEGIN
    d.numEntries := 0; d.name := ""; d.number := ""
END Pbox16A.
```

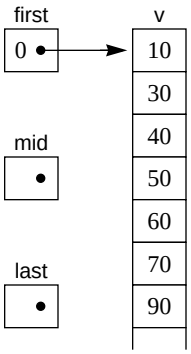
```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
        OUT i: INTEGER; OUT fnd: BOOLEAN);
    VAR
        first, mid, last: INTEGER;
BEGIN
    ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
    first := 0;
    last := numItm - 1;
    WHILE first <= last DO
        mid := (first + last) DIV 2;
        IF srchNum < v[mid] THEN
            last := mid - 1
        ELSIF srchNum > v[mid] THEN
            first := mid + 1
        ELSE
            fnd := TRUE; i := mid;
            RETURN
        END
    END;
    fnd := FALSE
END Search;
```

Figure 16.9
The binary search algorithm.

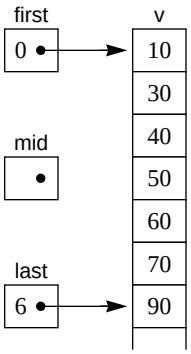
```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
   VAR                                          numItm    srchNum   i        fnd
      first, mid, last: INTEGER;                7         40
BEGIN
   ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
   first := 0;
   last := numItm - 1;
   WHILE first <= last DO
      mid := (first + last) DIV 2;
      IF srchNum < v[mid] THEN
         last := mid - 1
      ELSIF srchNum > v[mid] THEN
         first := mid + 1
      ELSE
         fnd := TRUE; i := mid;
         RETURN
      END
   END;
   fnd := FALSE
END Search;
```



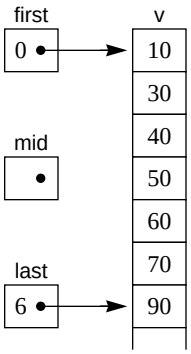
```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
   VAR                                          numItm    srchNum   i        fnd
      first, mid, last: INTEGER;                7         40
BEGIN
   ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
   first := 0;
   last := numItm - 1;
   WHILE first <= last DO
      mid := (first + last) DIV 2;
      IF srchNum < v[mid] THEN
         last := mid - 1
      ELSIF srchNum > v[mid] THEN
         first := mid + 1
      ELSE
         fnd := TRUE; i := mid;
         RETURN
      END
   END;
   fnd := FALSE
END Search;
```



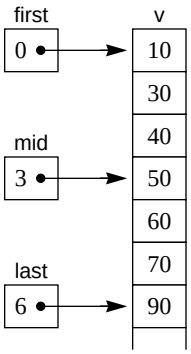
```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
   VAR                                          numItm    srchNum   i        fnd
      first, mid, last: INTEGER;                7         40
BEGIN
   ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
   first := 0;
   last := numItm - 1;
   WHILE first <= last DO
      mid := (first + last) DIV 2;
      IF srchNum < v[mid] THEN
         last := mid - 1
      ELSIF srchNum > v[mid] THEN
         first := mid + 1
      ELSE
         fnd := TRUE; i := mid;
         RETURN
      END
   END;
   fnd := FALSE
END Search;
```



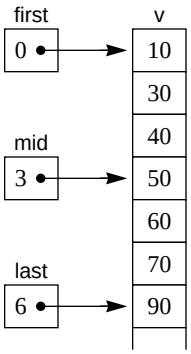
```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
   VAR                                          numItm    srchNum   i        fnd
      first, mid, last: INTEGER;                7         40
BEGIN
   ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
   first := 0;
   last := numItm - 1;
   WHILE first <= last DO
      mid := (first + last) DIV 2;
      IF srchNum < v[mid] THEN
         last := mid - 1
      ELSIF srchNum > v[mid] THEN
         first := mid + 1
      ELSE
         fnd := TRUE; i := mid;
         RETURN
      END
   END;
   fnd := FALSE
END Search;
```



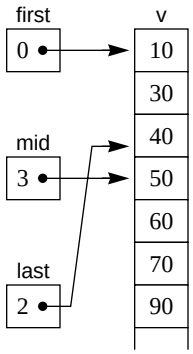
```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
   VAR                                          numItm    srchNum   i        fnd
      first, mid, last: INTEGER;                7         40
BEGIN
   ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
   first := 0;
   last := numItm - 1;
   WHILE first <= last DO
      mid := (first + last) DIV 2;
      IF srchNum < v[mid] THEN
         last := mid - 1
      ELSIF srchNum > v[mid] THEN
         first := mid + 1
      ELSE
         fnd := TRUE; i := mid;
         RETURN
      END
   END;
   fnd := FALSE
END Search;
```



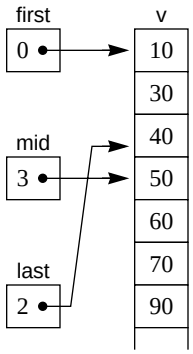
```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
   VAR                                          numItm    srchNum   i        fnd
      first, mid, last: INTEGER;                7         40
BEGIN
   ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
   first := 0;
   last := numItm - 1;
   WHILE first <= last DO
      mid := (first + last) DIV 2;
      IF srchNum < v[mid] THEN
         last := mid - 1
      ELSIF srchNum > v[mid] THEN
         first := mid + 1
      ELSE
         fnd := TRUE; i := mid;
         RETURN
      END
   END;
   fnd := FALSE
END Search;
```



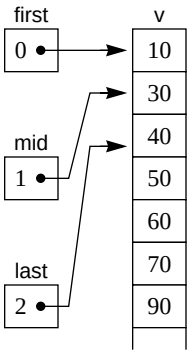
```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
   VAR                                          numItm    srchNum   i        fnd
      first, mid, last: INTEGER;                7         40
BEGIN
   ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
   first := 0;
   last := numItm - 1;
   WHILE first <= last DO
      mid := (first + last) DIV 2;
      IF srchNum < v[mid] THEN
         last := mid - 1
      ELSIF srchNum > v[mid] THEN
         first := mid + 1
      ELSE
         fnd := TRUE; i := mid;
         RETURN
      END
   END;
   fnd := FALSE
END Search;
```



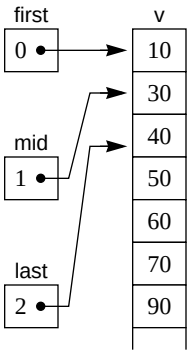
```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
   VAR                                          numItm    srchNum   i        fnd
      first, mid, last: INTEGER;                7         40
BEGIN
   ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
   first := 0;
   last := numItm - 1;
   WHILE first <= last DO
      mid := (first + last) DIV 2;
      IF srchNum < v[mid] THEN
         last := mid - 1
      ELSIF srchNum > v[mid] THEN
         first := mid + 1
      ELSE
         fnd := TRUE; i := mid;
         RETURN
      END
   END;
   fnd := FALSE
END Search;
```



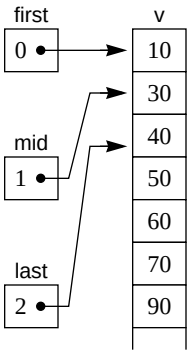
```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
   VAR                                          numItm   srchNum   i      fnd
      first, mid, last: INTEGER;                7        40
BEGIN
   ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
   first := 0;
   last := numItm - 1;
   WHILE first <= last DO
      mid := (first + last) DIV 2;
      IF srchNum < v[mid] THEN
         last := mid - 1
      ELSIF srchNum > v[mid] THEN
         first := mid + 1
      ELSE
         fnd := TRUE; i := mid;
         RETURN
      END
   END;
   fnd := FALSE
END Search;
```



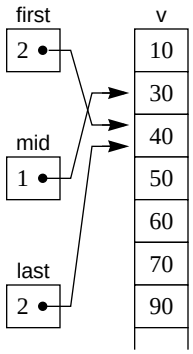
```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
   VAR                                          numItm   srchNum   i      fnd
      first, mid, last: INTEGER;                7        40
BEGIN
   ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
   first := 0;
   last := numItm - 1;
   WHILE first <= last DO
      mid := (first + last) DIV 2;
      IF srchNum < v[mid] THEN
         last := mid - 1
      ELSIF srchNum > v[mid] THEN
         first := mid + 1
      ELSE
         fnd := TRUE; i := mid;
         RETURN
      END
   END;
   fnd := FALSE
END Search;
```



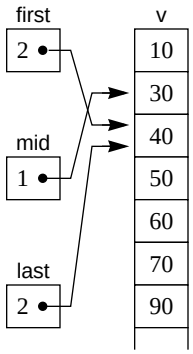
```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
   VAR                                          numItm   srchNum   i      fnd
      first, mid, last: INTEGER;                7        40
BEGIN
   ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
   first := 0;
   last := numItm - 1;
   WHILE first <= last DO
      mid := (first + last) DIV 2;
      IF srchNum < v[mid] THEN
         last := mid - 1
      ELSIF srchNum > v[mid] THEN
         first := mid + 1
      ELSE
         fnd := TRUE; i := mid;
         RETURN
      END
   END;
   fnd := FALSE
END Search;
```



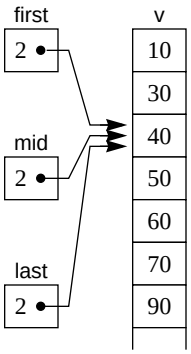
```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
   VAR                                          numItm   srchNum   i      fnd
      first, mid, last: INTEGER;                7        40
BEGIN
   ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
   first := 0;
   last := numItm - 1;
   WHILE first <= last DO
      mid := (first + last) DIV 2;
      IF srchNum < v[mid] THEN
         last := mid - 1
      ELSIF srchNum > v[mid] THEN
         first := mid + 1
      ELSE
         fnd := TRUE; i := mid;
         RETURN
      END
   END;
   fnd := FALSE
END Search;
```



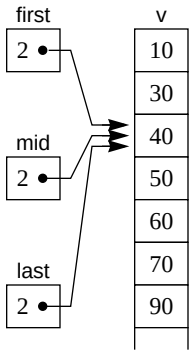
```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
   VAR                                          numItm    srchNum   i        fnd
      first, mid, last: INTEGER;                7         40
BEGIN
   ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
   first := 0;
   last := numItm - 1;
   WHILE first <= last DO
      mid := (first + last) DIV 2;
      IF srchNum < v[mid] THEN
         last := mid - 1
      ELSIF srchNum > v[mid] THEN
         first := mid + 1
      ELSE
         fnd := TRUE; i := mid;
         RETURN
      END
   END;
   fnd := FALSE
END Search;
```



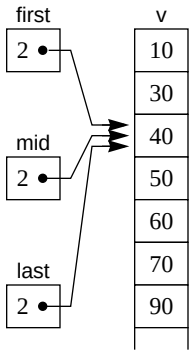
```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
   VAR                                          numItm    srchNum   i        fnd
      first, mid, last: INTEGER;                7         40
BEGIN
   ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
   first := 0;
   last := numItm - 1;
   WHILE first <= last DO
      mid := (first + last) DIV 2;
      IF srchNum < v[mid] THEN
         last := mid - 1
      ELSIF srchNum > v[mid] THEN
         first := mid + 1
      ELSE
         fnd := TRUE; i := mid;
         RETURN
      END
   END;
   fnd := FALSE
END Search;
```



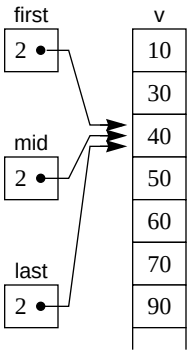
```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
   VAR                                          numItm    srchNum   i        fnd
      first, mid, last: INTEGER;                7         40
BEGIN
   ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
   first := 0;
   last := numItm - 1;
   WHILE first <= last DO
      mid := (first + last) DIV 2;
      IF srchNum < v[mid] THEN
         last := mid - 1
      ELSIF srchNum > v[mid] THEN
         first := mid + 1
      ELSE
         fnd := TRUE; i := mid;
         RETURN
      END
   END;
   fnd := FALSE
END Search;
```



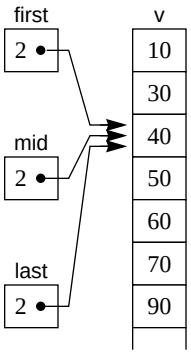
```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
   VAR                                          numItm   srchNum   i      fnd
      first, mid, last: INTEGER;                7        40
BEGIN
   ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
   first := 0;
   last := numItm - 1;
   WHILE first <= last DO
      mid := (first + last) DIV 2;
      IF srchNum < v[mid] THEN
         last := mid - 1
      ELSIF srchNum > v[mid] THEN
         first := mid + 1
      ELSE
         fnd := TRUE; i := mid;
         RETURN
      END
   END;
   fnd := FALSE
END Search;
```



```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
   VAR                                          numItm   srchNum   i      fnd
      first, mid, last: INTEGER;                7        40        2      TRUE
BEGIN
   ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
   first := 0;
   last := numItm - 1;
   WHILE first <= last DO
      mid := (first + last) DIV 2;
      IF srchNum < v[mid] THEN
         last := mid - 1
      ELSIF srchNum > v[mid] THEN
         first := mid + 1
      ELSE
         fnd := TRUE; i := mid;
         RETURN
      END
   END;
   fnd := FALSE
END Search;
```



```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
   VAR                                          numItm    srchNum   i      fnd
      first, mid, last: INTEGER;                7         40        2      TRUE
BEGIN
   ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
   first := 0;
   last := numItm - 1;
   WHILE first <= last DO
      mid := (first + last) DIV 2;
      IF srchNum < v[mid] THEN
         last := mid - 1
      ELSIF srchNum > v[mid] THEN
         first := mid + 1
      ELSE
         fnd := TRUE; i := mid;
         RETURN
      END
   END;
   fnd := FALSE
END Search;
```



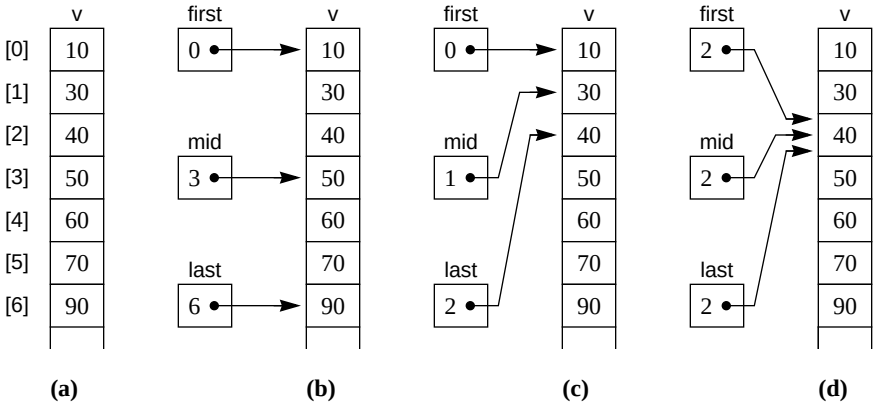
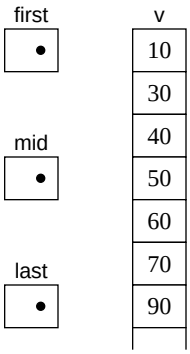
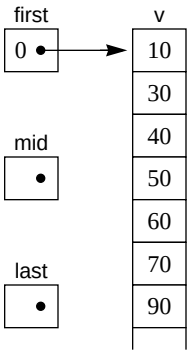


Figure 16.10
A trace of the binary search algorithm when the value of srchNum, 40, is in the list.

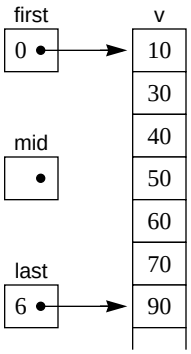
```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
   VAR                                          numItm    srchNum   i        fnd
      first, mid, last: INTEGER;                7         80
BEGIN
   ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
   first := 0;
   last := numItm - 1;
   WHILE first <= last DO
      mid := (first + last) DIV 2;
      IF srchNum < v[mid] THEN
         last := mid - 1
      ELSIF srchNum > v[mid] THEN
         first := mid + 1
      ELSE
         fnd := TRUE; i := mid;
         RETURN
      END
   END;
   fnd := FALSE
END Search;
```



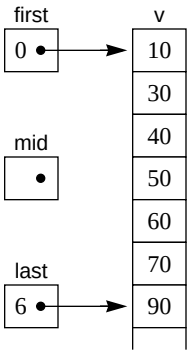
```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
   VAR                                          numItm    srchNum   i        fnd
      first, mid, last: INTEGER;                7         80
BEGIN
   ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
   first := 0;
   last := numItm - 1;
   WHILE first <= last DO
      mid := (first + last) DIV 2;
      IF srchNum < v[mid] THEN
         last := mid - 1
      ELSIF srchNum > v[mid] THEN
         first := mid + 1
      ELSE
         fnd := TRUE; i := mid;
         RETURN
      END
   END;
   fnd := FALSE
END Search;
```



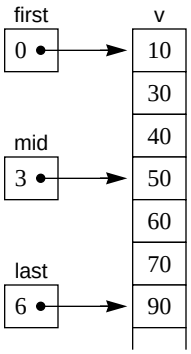
```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
   VAR                                          numItm    srchNum   i        fnd
      first, mid, last: INTEGER;                7         80
BEGIN
   ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
   first := 0;
   last := numItm - 1;
   WHILE first <= last DO
      mid := (first + last) DIV 2;
      IF srchNum < v[mid] THEN
         last := mid - 1
      ELSIF srchNum > v[mid] THEN
         first := mid + 1
      ELSE
         fnd := TRUE; i := mid;
         RETURN
      END
   END;
   fnd := FALSE
END Search;
```



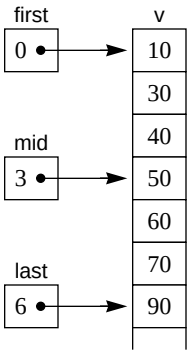
```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
   VAR                                              numItm    srchNum   i        fnd
      first, mid, last: INTEGER;                    7         80
BEGIN
   ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
   first := 0;
   last := numItm - 1;
   WHILE first <= last DO
      mid := (first + last) DIV 2;
      IF srchNum < v[mid] THEN
         last := mid - 1
      ELSIF srchNum > v[mid] THEN
         first := mid + 1
      ELSE
         fnd := TRUE; i := mid;
         RETURN
      END
   END;
   fnd := FALSE
END Search;
```



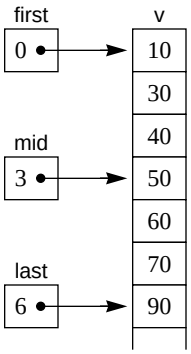
```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
   VAR                                          numItm    srchNum   i        fnd
      first, mid, last: INTEGER;                7         80
BEGIN
   ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
   first := 0;
   last := numItm - 1;
   WHILE first <= last DO
      mid := (first + last) DIV 2;
      IF srchNum < v[mid] THEN
         last := mid - 1
      ELSIF srchNum > v[mid] THEN
         first := mid + 1
      ELSE
         fnd := TRUE; i := mid;
         RETURN
      END
   END;
   fnd := FALSE
END Search;
```



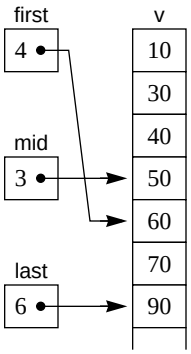
```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
   VAR                                          numItm    srchNum   i        fnd
      first, mid, last: INTEGER;                7         80
BEGIN
   ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
   first := 0;
   last := numItm - 1;
   WHILE first <= last DO
      mid := (first + last) DIV 2;
      IF srchNum < v[mid] THEN
         last := mid - 1
      ELSIF srchNum > v[mid] THEN
         first := mid + 1
      ELSE
         fnd := TRUE; i := mid;
         RETURN
      END
   END;
   fnd := FALSE
END Search;
```



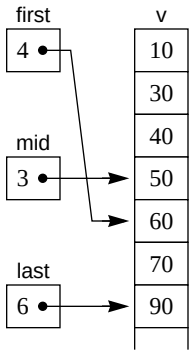
```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
   VAR                                              numItm    srchNum   i        fnd
      first, mid, last: INTEGER;                    7         80
BEGIN
   ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
   first := 0;
   last := numItm - 1;
   WHILE first <= last DO
      mid := (first + last) DIV 2;
      IF srchNum < v[mid] THEN
         last := mid - 1
      ELSIF srchNum > v[mid] THEN
         first := mid + 1
      ELSE
         fnd := TRUE; i := mid;
         RETURN
      END
   END;
   fnd := FALSE
END Search;
```



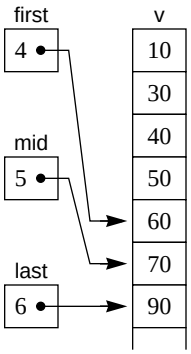
```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
   VAR                                          numItm    srchNum   i        fnd
      first, mid, last: INTEGER;                7         80
BEGIN
   ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
   first := 0;
   last := numItm - 1;
   WHILE first <= last DO
      mid := (first + last) DIV 2;
      IF srchNum < v[mid] THEN
         last := mid - 1
      ELSIF srchNum > v[mid] THEN
         first := mid + 1
      ELSE
         fnd := TRUE; i := mid;
         RETURN
      END
   END;
   fnd := FALSE
END Search;
```



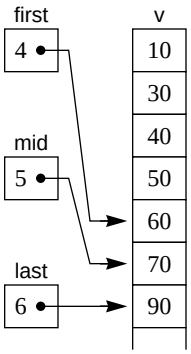
```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
   VAR                                                  numItm   srchNum   i      fnd
      first, mid, last: INTEGER;                        7        80
BEGIN
   ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
   first := 0;
   last := numItm - 1;
   WHILE first <= last DO
      mid := (first + last) DIV 2;
      IF srchNum < v[mid] THEN
         last := mid - 1
      ELSIF srchNum > v[mid] THEN
         first := mid + 1
      ELSE
         fnd := TRUE; i := mid;
         RETURN
      END
   END;
   fnd := FALSE
END Search;
```



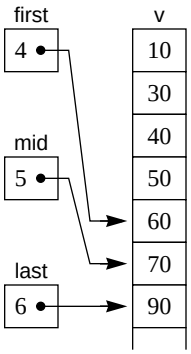
```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
   VAR                                          numItm   srchNum   i      fnd
      first, mid, last: INTEGER;                7        80
BEGIN
   ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
   first := 0;
   last := numItm - 1;
   WHILE first <= last DO
      mid := (first + last) DIV 2;
      IF srchNum < v[mid] THEN
         last := mid - 1
      ELSIF srchNum > v[mid] THEN
         first := mid + 1
      ELSE
         fnd := TRUE; i := mid;
         RETURN
      END
   END;
   fnd := FALSE
END Search;
```



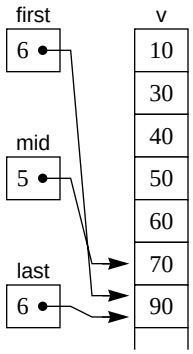
```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
   VAR                                          numItm    srchNum   i        fnd
      first, mid, last: INTEGER;                7         80
BEGIN
   ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
   first := 0;
   last := numItm - 1;
   WHILE first <= last DO
      mid := (first + last) DIV 2;
      IF srchNum < v[mid] THEN
         last := mid - 1
      ELSIF srchNum > v[mid] THEN
         first := mid + 1
      ELSE
         fnd := TRUE; i := mid;
         RETURN
      END
   END;
   fnd := FALSE
END Search;
```



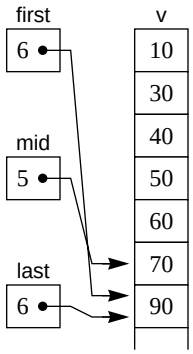
```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
   VAR                                              numItm    srchNum   i        fnd
      first, mid, last: INTEGER;                    7         80
BEGIN
   ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
   first := 0;
   last := numItm - 1;
   WHILE first <= last DO
      mid := (first + last) DIV 2;
      IF srchNum < v[mid] THEN
         last := mid - 1
      ELSIF srchNum > v[mid] THEN
         first := mid + 1
      ELSE
         fnd := TRUE; i := mid;
         RETURN
      END
   END;
   fnd := FALSE
END Search;
```



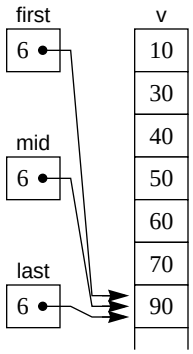
```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
   VAR                                              numItm    srchNum   i        fnd
      first, mid, last: INTEGER;                    7         80
BEGIN
   ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
   first := 0;
   last := numItm - 1;
   WHILE first <= last DO
      mid := (first + last) DIV 2;
      IF srchNum < v[mid] THEN
         last := mid - 1
      ELSIF srchNum > v[mid] THEN
         first := mid + 1
      ELSE
         fnd := TRUE; i := mid;
         RETURN
      END
   END;
   fnd := FALSE
END Search;
```



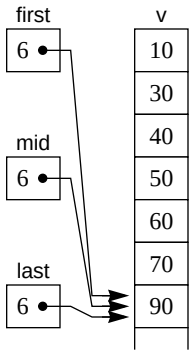
```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
   VAR                                              numItm    srchNum   i        fnd
      first, mid, last: INTEGER;                    7         80
BEGIN
   ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
   first := 0;
   last := numItm - 1;
   WHILE first <= last DO
      mid := (first + last) DIV 2;
      IF srchNum < v[mid] THEN
         last := mid - 1
      ELSIF srchNum > v[mid] THEN
         first := mid + 1
      ELSE
         fnd := TRUE; i := mid;
         RETURN
      END
   END;
   fnd := FALSE
END Search;
```



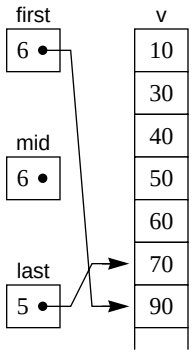
```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
   VAR                                          numItm    srchNum   i        fnd
      first, mid, last: INTEGER;                7         80
BEGIN
   ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
   first := 0;
   last := numItm - 1;
   WHILE first <= last DO
      mid := (first + last) DIV 2;
      IF srchNum < v[mid] THEN
         last := mid - 1
      ELSIF srchNum > v[mid] THEN
         first := mid + 1
      ELSE
         fnd := TRUE; i := mid;
         RETURN
      END
   END;
   fnd := FALSE
END Search;
```



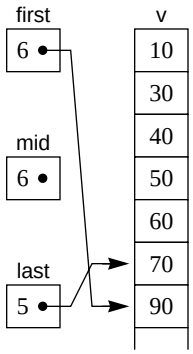
```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
   VAR                                          numItm   srchNum   i        fnd
      first, mid, last: INTEGER;                7        80
BEGIN
   ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
   first := 0;
   last := numItm - 1;
   WHILE first <= last DO
      mid := (first + last) DIV 2;
      IF srchNum < v[mid] THEN
         last := mid - 1
      ELSIF srchNum > v[mid] THEN
         first := mid + 1
      ELSE
         fnd := TRUE; i := mid;
         RETURN
      END
   END;
   fnd := FALSE
END Search;
```



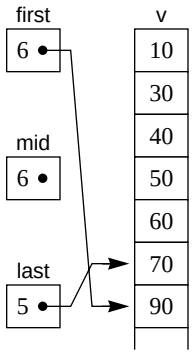
```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
   VAR                                              numItm    srchNum   i        fnd
      first, mid, last: INTEGER;                    7         80
BEGIN
   ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
   first := 0;
   last := numItm - 1;
   WHILE first <= last DO
      mid := (first + last) DIV 2;
      IF srchNum < v[mid] THEN
         last := mid - 1
      ELSIF srchNum > v[mid] THEN
         first := mid + 1
      ELSE
         fnd := TRUE; i := mid;
         RETURN
      END
   END;
   fnd := FALSE
END Search;
```



```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
   VAR                                              numItm    srchNum   i        fnd
      first, mid, last: INTEGER;                    7         80
BEGIN
   ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
   first := 0;
   last := numItm - 1;
   WHILE first <= last DO
      mid := (first + last) DIV 2;
      IF srchNum < v[mid] THEN
         last := mid - 1
      ELSIF srchNum > v[mid] THEN
         first := mid + 1
      ELSE
         fnd := TRUE; i := mid;
         RETURN
      END
   END;
   fnd := FALSE
END Search;
```



```
PROCEDURE Search (IN v: ARRAY OF INTEGER; numItm, srchNum: INTEGER;
      OUT i: INTEGER; OUT fnd: BOOLEAN);
   VAR                                          numItm    srchNum   i     fnd
      first, mid, last: INTEGER;                7         80              FALSE
BEGIN
   ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
   first := 0;
   last := numItm - 1;
   WHILE first <= last DO
      mid := (first + last) DIV 2;
      IF srchNum < v[mid] THEN
         last := mid - 1
      ELSIF srchNum > v[mid] THEN
         first := mid + 1
      ELSE
         fnd := TRUE; i := mid;
         RETURN
      END
   END;
   fnd := FALSE
END Search;
```



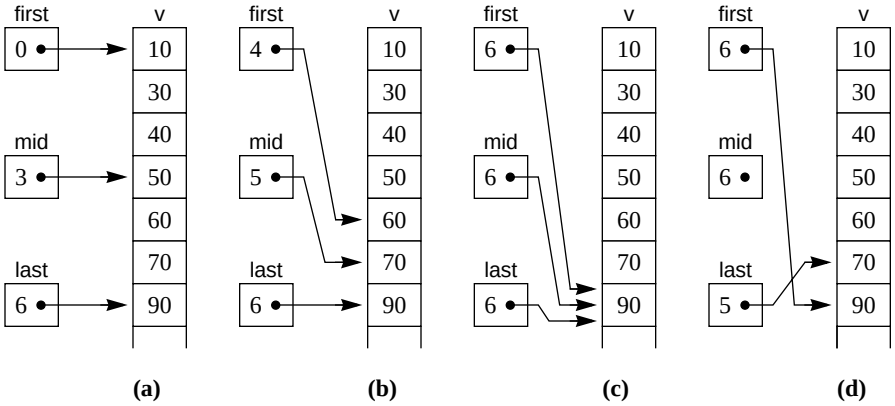


Figure 16.11
A trace of the binary search algorithm when the value of srchNum, 80, is not in the list.

If numItm has the value *n*, how many times will the loop execute?

Let *t* equal the number of times. Here are some specific values of *n* and *t*:

n = 16, *t* = ?

If numItm has the value *n*, how many times will the loop execute?

Let *t* equal the number of times. Here are some specific values of *n* and *t*:

n = 16, *t* = 4 times: 16 8 4 2 1
n = 32, *t* = ?

If numItm has the value *n*, how many times will the loop execute?

Let *t* equal the number of times. Here are some specific values of *n* and *t*:

n = 16, *t* = 4 times: 16 8 4 2 1
n = 32, *t* = 5 times: 32 16 8 4 2 1
n = 64, *t* = ?

If numItm has the value *n*, how many times will the loop execute?

Let *t* equal the number of times. Here are some specific values of *n* and *t*:

n = 16, *t* = 4 times: 16 8 4 2 1
n = 32, *t* = 5 times: 32 16 8 4 2 1
n = 64, *t* = 6 times: 64 32 16 8 4 2 1

The general relationship between *n* and *t* is ?

If numItm has the value *n*, how many times will the loop execute?

Let *t* equal the number of times. Here are some specific values of *n* and *t*:

n = 16, *t* = 4 times: 16 8 4 2 1
n = 32, *t* = 5 times: 32 16 8 4 2 1
n = 64, *t* = 6 times: 64 32 16 8 4 2 1

The general relationship between *n* and *t* is $n = 2^t$.

$\log_2 n = \lg n$

lg is the logarithm to the base 2.

$t = \lg n$

Each time the loop executes, it makes three comparisons. So the number of comparisons is $3\lg n$.

- Best case: 3 comparisons
- Worst case: $3\lg n$ comparisons

Sort algorithms put lists of values in order. For example, if numItm is 9 and the first 9 values of v are

7 3 8 2 1 4 9 5 6

then after the sort, the nine values should be

1 2 3 4 5 6 7 8 9

```
FOR k := numItm - 1 TO 1 BY -1 DO
    Select the largest element from the top part of the list between v[0] and v[k].
    Exchange it with v[k].
END
```

```
PROCEDURE Sort (VAR v: ARRAY OF INTEGER; numItm: INTEGER);
    VAR
        i, k: INTEGER;
        maxIndex: INTEGER;
        temp: INTEGER;
BEGIN
    ASSERT((0 <= numItm) & (numItm <= LEN(v)), 20);
    FOR k := numItm - 1 TO 1 BY -1 DO
        maxIndex := 0;
        FOR i := 1 TO k DO
            IF v[i] > v[maxIndex] THEN
                maxIndex := i
            END
        END;
        temp := v[k];
        v[k] := v[maxIndex];
        v[maxIndex] := temp
    END
END Sort;
```

Figure 16.12
The selection sort.

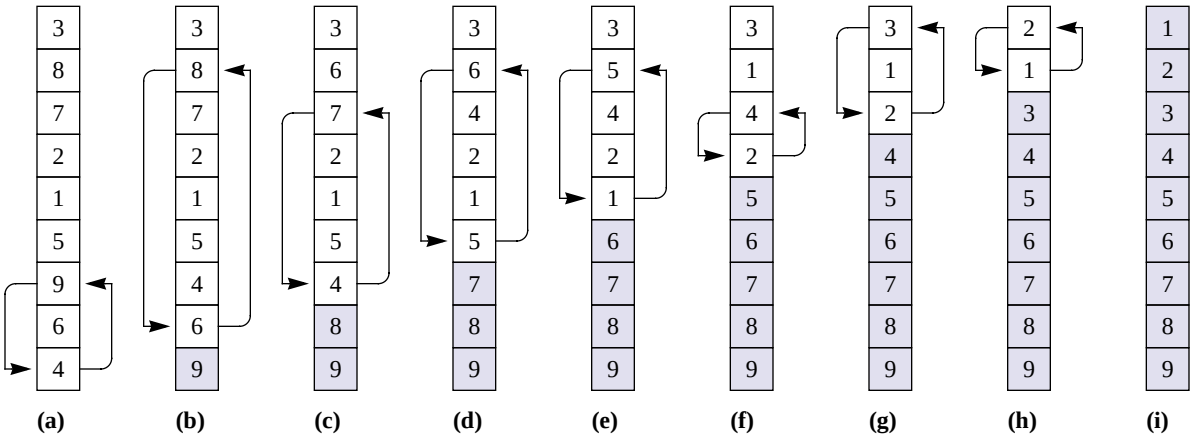


Figure 16.13
Eight executions of the outer loop of the selection sort. The shaded cells contain the values of the array that are in order.

The inner loop makes one comparison each time it executes. So the number of comparisons is the number of times the inner loop executes. Let *n* be the value of numItm.

The number of comparisons is

$(n-1) + (n-2) + (n-3) + \ldots + 3 + 2 + 1$

$n(n-1)/2$