Binary Search

- Strategy to be employed when items in a list are ordered
- Comparable to divide an conquer strategy
  - At each stage of the strategy we divide the list in half based on the median value

- For example:

int []x = {16,19,22,24,27,29,37,40,43,44,47,52,60,64,71}

Let’s search for 27!

Step 1 – Identify the middle value

Step 2 – Determine if the item is less than the middle value

- if it is we can eliminate all the values above the middle value
- if it is not we can eliminate all the values below it

Step 3 – Repeat Steps 1 and 2 until we have one value remaining!

- We start with:
  - 16 19 22 24 27 29 37 40 43 44 47 52 56 60 64 71
  - The bottom value is 16, the middle value is 40, the top value is 71

- Identify the middle value (40) and eliminate half the values
  - 16 19 22 24 27 29 37
  - The bottom value is 16, the middle value is 24, the top value is 37

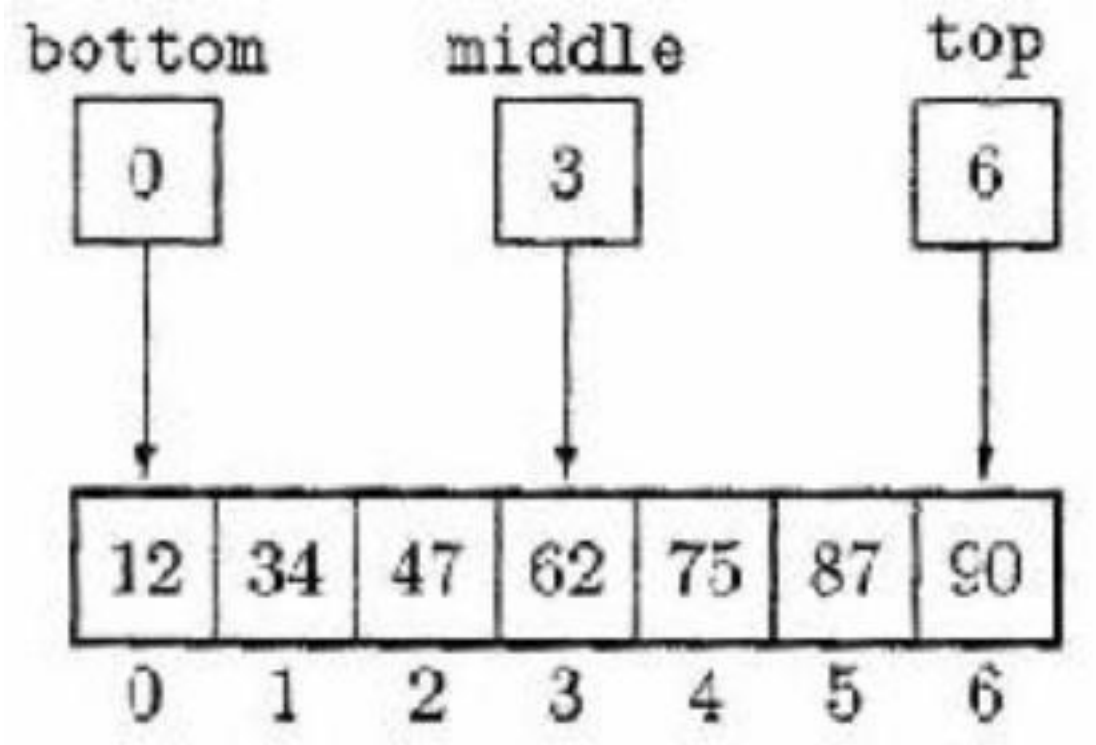
- Identify the middle value (24) and eliminate half the values
  - 27 29 37
  - The bottom value is 27, the middle value is 29, the top value is 37

- Identify the middle value (29) and eliminate half the values
  - 27
  - The remaining value is 27

Example – int []x = {12, 34, 47, 62, 75, 87, 90}

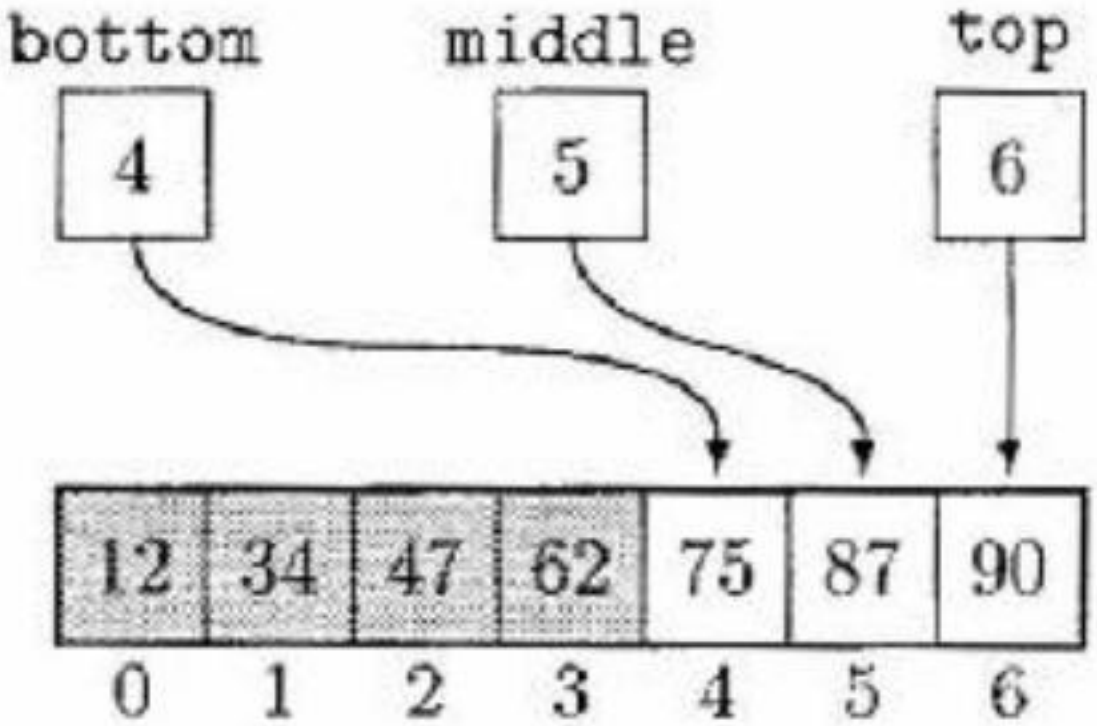
- Create variables that keep track of the following:
- *bottom*
- *middle*
- *top*

- *Let's search for 75*



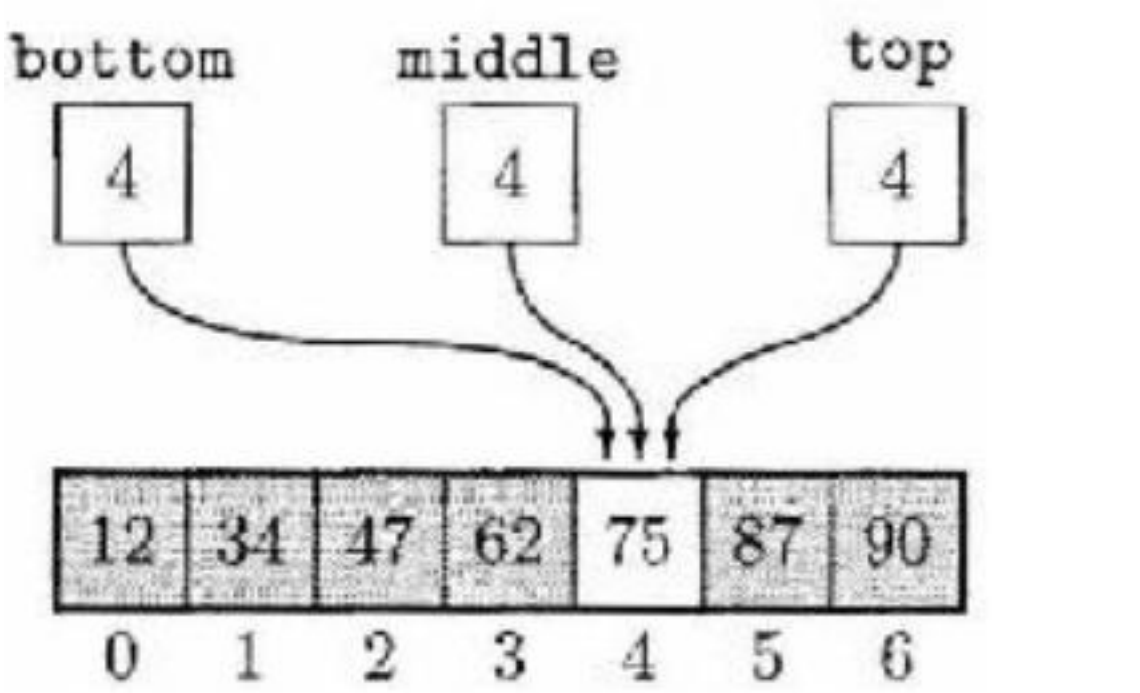
- The middle value is 62 and since 62<75, we know that the value is not in the first half of the array.

- Discard the first half and set the bottom index to the middle index + 1
- The new middle index will be (4+6)/2 = 5
- The top index is unchanged at 6



- The middle value is 87 and since 87>75, we know that the value is not in the second half of the array.

- Discard the second half and set the top index to the middle index -1
- The new middle index will be (4+4)/2 = 4
- The bottom index is unchanged at 4



- Once middle has found the value, the search ends successfully

```
public static int binSearch (double[] list, double item)
{
   int bottom = 0;                    // lower bound of subarray
   int top = list.length - 1;         // upper bound of subarray
   int middle;                        // middle of subarray
   boolean found = false;             // to stop if item found
   int location = -1;                 // index of item in array

   while (bottom <= top && !found)
   {
      middle = (bottom + top)/2;
      if (list[middle] == item)       // success
      {
         found = true;
         location = middle;
      }
      else if (list[middle] < item) // not in bottom half
         bottom = middle + 1;
      else                           // item cannot be in top half
         top = middle - 1;
   }
   return location;
}
```

double [] x = {12,34,47,62,75,87,90};
System.***out.println(binSearch(x,75));***