Quick sort

CHRONOLOGICAL LISTING OF A.M. TURING AWARD WINNERS

* person is deceased

(2019)
Catmull, Edwin E.
Hanrahan, Patrick M.

(2018)
Bengio, Yoshua
Hinton, Geoffrey E
LeCun, Yann

(2017)
Hennessy, John L
Patterson, David

(2016)
Berners-Lee, Tim

(2015)
Diffie, Whitfield
Hellman, Martin

(2014)
Stonebraker, Michael

(2013)
Lamport, Leslie

(2012)
Goldwasser, Shafi
Micali, Silvio

(2011)
Pearl, Judea

(2010)

(2000)
Yao, Andrew Chi-Chih

(1999)
Brooks, Frederick ("Fred")

(1998)
Gray, James ("Jim") Nicholas *

(1997)
Engelbart, Douglas *

(1996)
Pnueli, Amir *

(1995)
Blum, Manuel

(1994)
Feigenbaum, Edward A ("Ed")
Reddy, Dabbala Rajagopal ("Raj")

(1993)
Hartmanis, Juris
Stearns, Richard ("Dick") Edwin

(1992)
Lampson, Butler W

(1991)
Milner, Arthur John Robin Gorell ("Robin") *

(1990)
Corbato, Fernando J ("Corby") *

(1981)
Codd, Edgar F. ("Ted") *

(1980)
Hoare, C. Antony ("Tony") R.

(1979)
Iverson, Kenneth E. ("Ken") *

(1978)
Floyd, Robert (Bob) W *

(1977)
Backus, John *

(1976)
Rabin, Michael O.
Scott, Dana Stewart

(1975)
Newell, Allen *
Simon, Herbert ("Herb") Alexander *

(1974)
Knuth, Donald ("Don") Ervin

(1973)
Bachman, Charles William *

(1972)
Dijkstra, Edsger Wybe *

(1971)
McCarthy, John *

PHOTOGRAPHS

C. ANTONY ("TONY") R. HOARE

United Kingdom – 1980

CITATION

For his fundamental contributions to the definition and design of programming languages.

BIRTH:

in Sri Lanka in 1934

EDUCATION:

Dragon School in Oxford and King's School in Canterbury. His university undergraduate education was also in Oxford and he did post graduate there as well as Moscow State University.

EXPERIENCE:

Royal Navy (1956-57), Elliott Brothers (1960-1968), Queens's University, Belfast (1968 - 1977); Oxford University (1977); currently Emeritus Professor at Oxford University and a Senior Researcher at Microsoft Research in Cambridge, UK.

Tony Hoare, 1980 ACM Turing Award Recipient - YouTube

https://www.youtube.com › watch ▾

Oct 25, 2016 - Uploaded by Association for Computing Machinery (ACM)
Speaks about his life, work, and the factors that influenced him during his long career. More information: ...

E B D H A F C G

E B D H A F C G

E B D H A F C G


E B D H A F C G
C B D H A F E G


E B D H A F C G
C B D H A F E G
C B D E A F H G

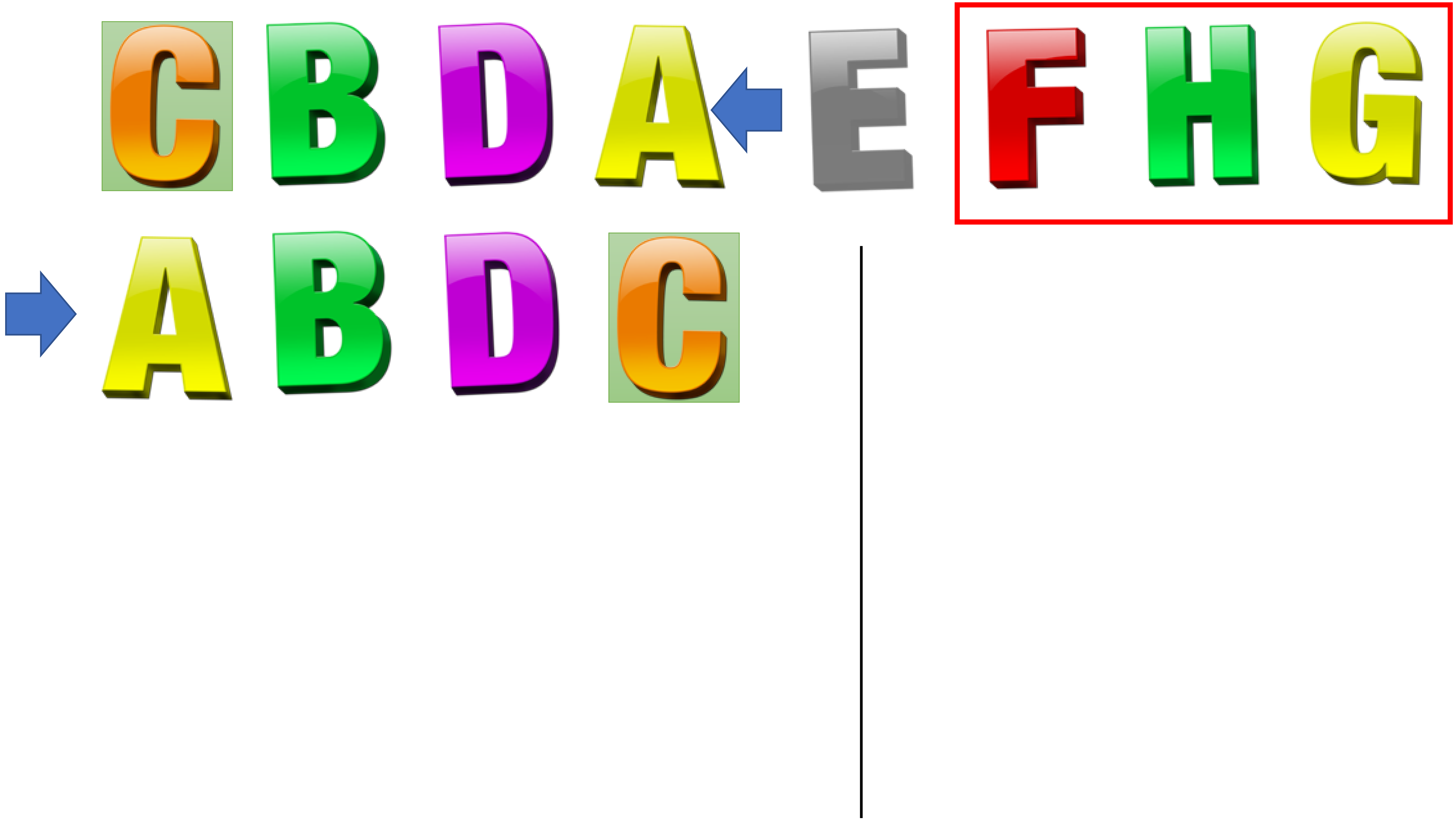

E B D H A F C G
C B D H A F E G
C B D E A F H G
C B D A E F H G

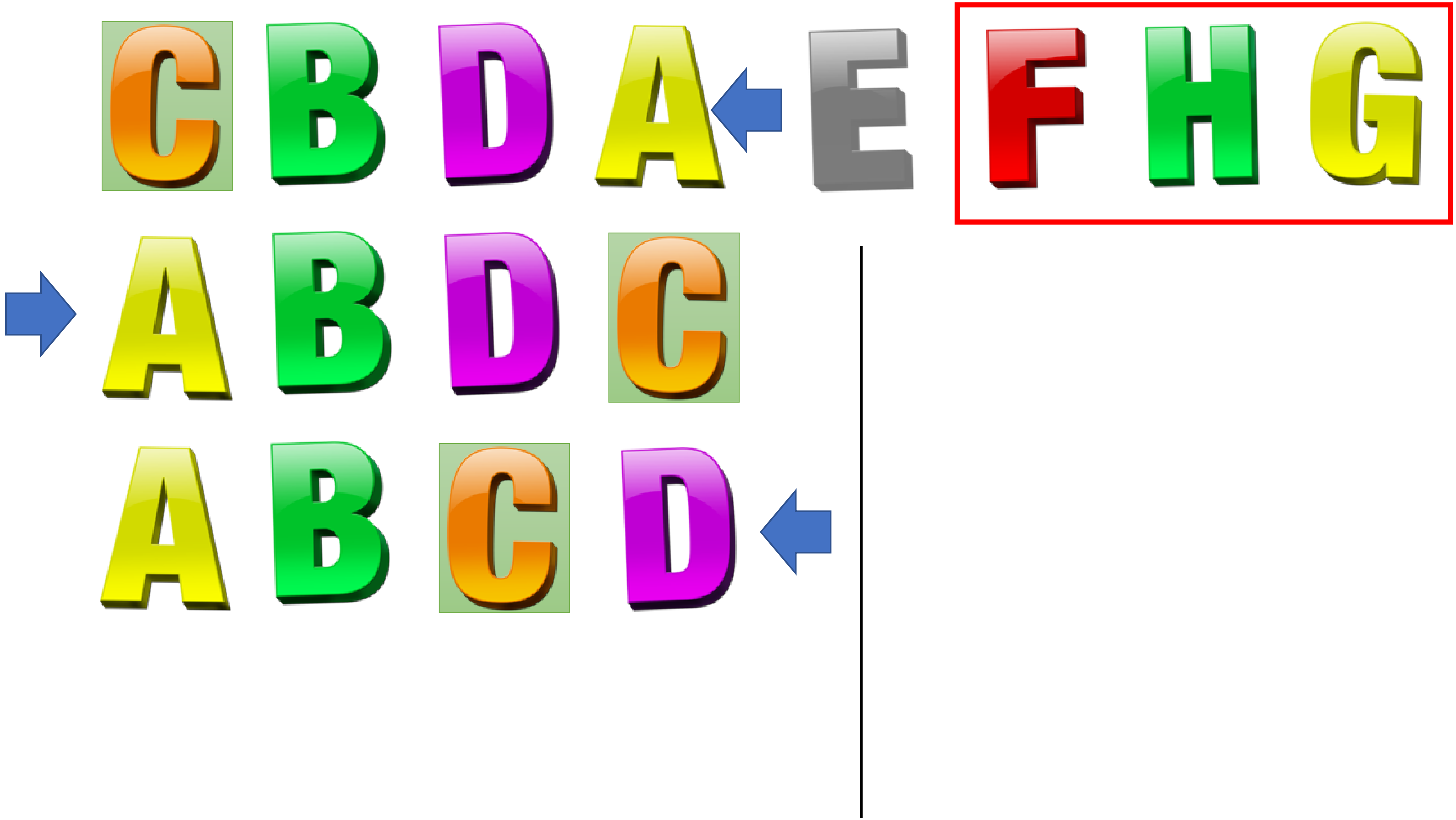

E B D H A F C G
C B D H A F E G
C B D E A F H G
C B D A E F H G

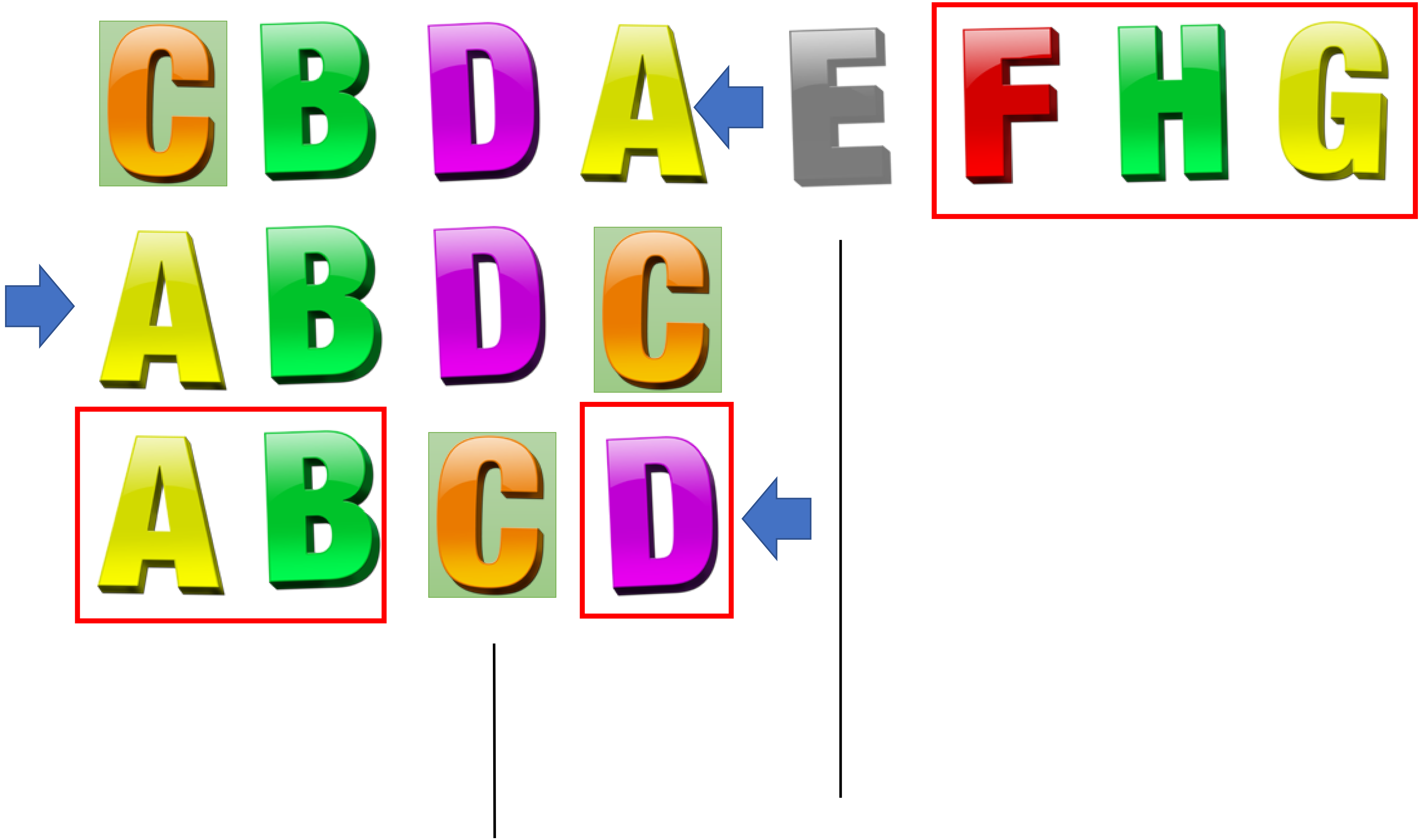
C B D A E F H G

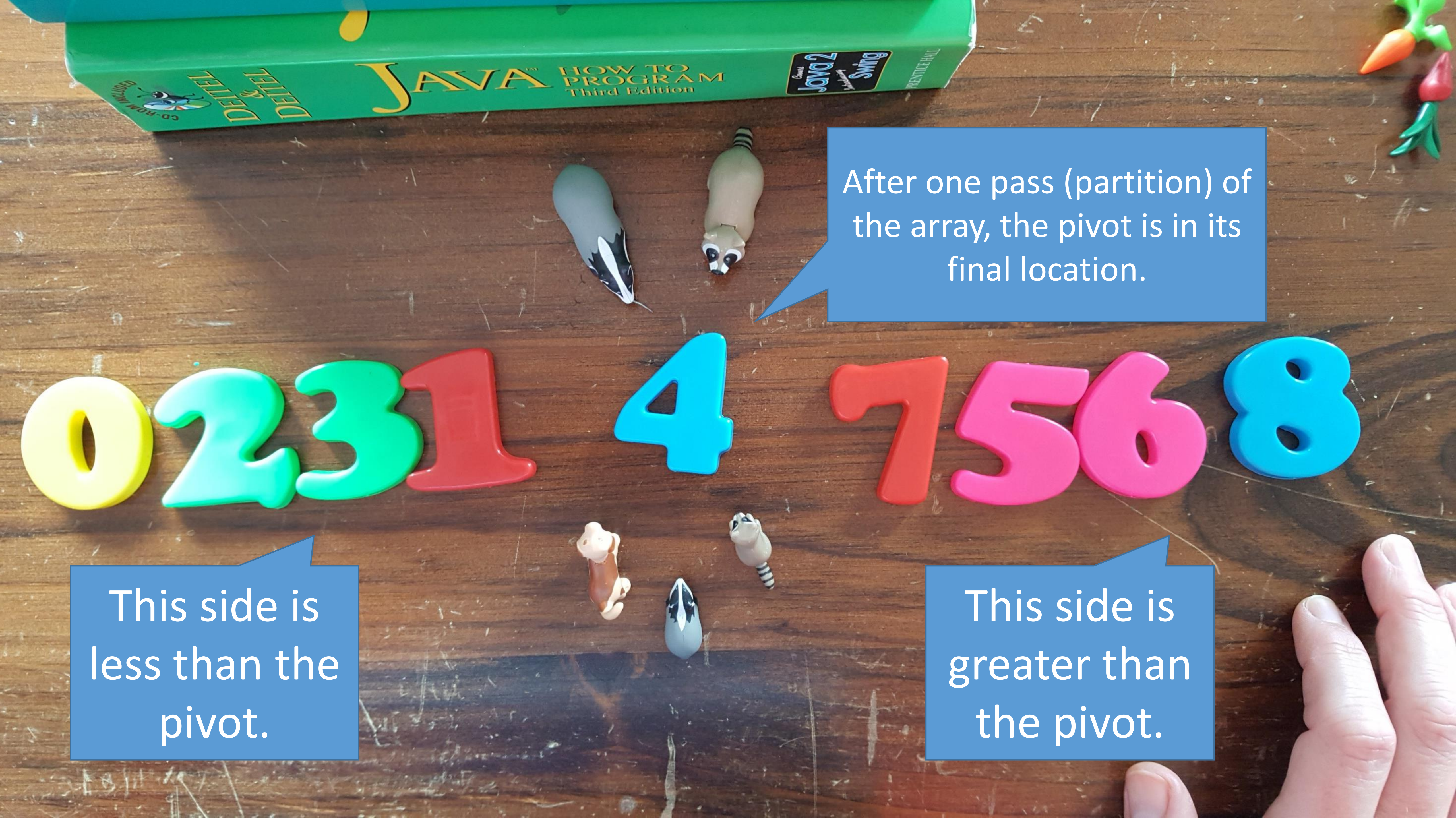

C B D A E F H G


C B D A E F H G


C B D A E F H G
A B D C


C B D A E F H G
A B D C
A B C D


C B D A E F H G
A B D C
A B C D



DEITEL & DEITEL
JAVA HOW TO PROGRAM
Third Edition
PRENTICE HALL
0231 4 7568
After one pass (partition) of the array, the pivot is in its final location.
This side is less than the pivot.
This side is greater than the pivot.

Quicksort

- Algorithm: use a pivot to partition the array into two halves: less than the pivot and greater than the pivot. Then, quicksort each half.
- Recursive.
- The fastest in-place algorithm in the general case.
- It uses swaps, so no extra memory is needed.
- It doesn't work well for non-random data (say, reverse order). In that case, use merge.
- Speed: O(n log n).
- Inventor: Tony Hoare

The Code

```
void quicksort(int[] array, int startIndex, int endIndex)
{
 if (startIndex >= endIndex) {
 return;

 } else {
 int pivotIndex = partition(array, startIndex, endIndex);
 quicksort(array, startIndex, pivotIndex - 1);
 quicksort(array, pivotIndex + 1, endIndex);
 }
}
```

The Code

```
void quicksort(int[] array, int startIndex, int endIndex)
{
 if (startIndex >= endIndex) {
 return;

 } else {
 int pivotIndex = partition(array, startIndex, endIndex);
 quicksort(array, startIndex, pivotIndex - 1);
 quicksort(array, pivotIndex + 1, endIndex);
 }
}
```

The Code

```
void quicksort(int[] array, int startIndex, int endIndex)
{
 if (startIndex >= endIndex) {
 return;

 } else {
 int pivotIndex = partition(array, startIndex, endIndex);
 quicksort(array, startIndex, pivotIndex - 1);
 quicksort(array, pivotIndex + 1, endIndex);
 }
}
```

Partition

Quicksort

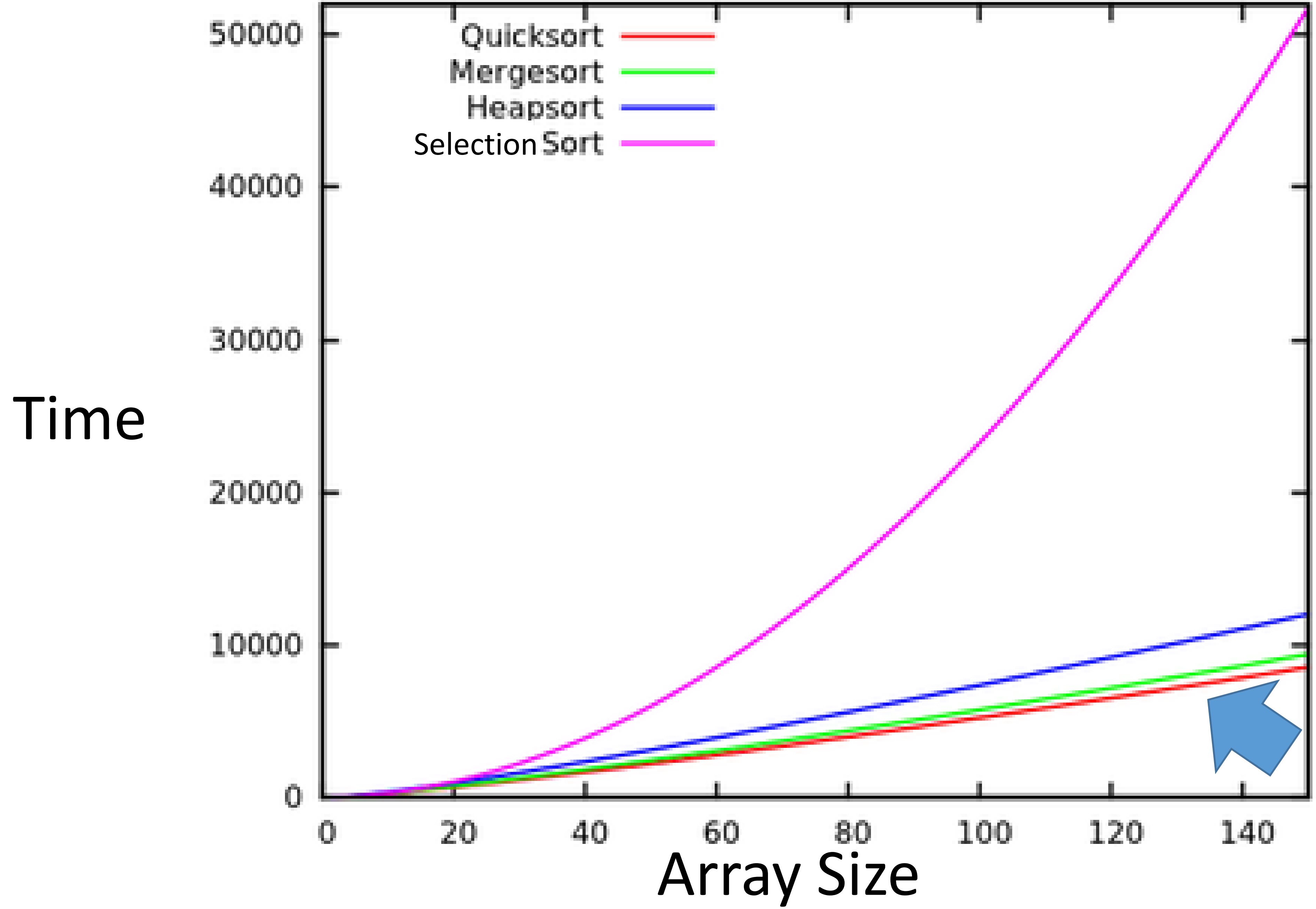
Quicksort’s Speed

O (n log n)

Order

n = length of array

Speed of Various Sorting Algorithms

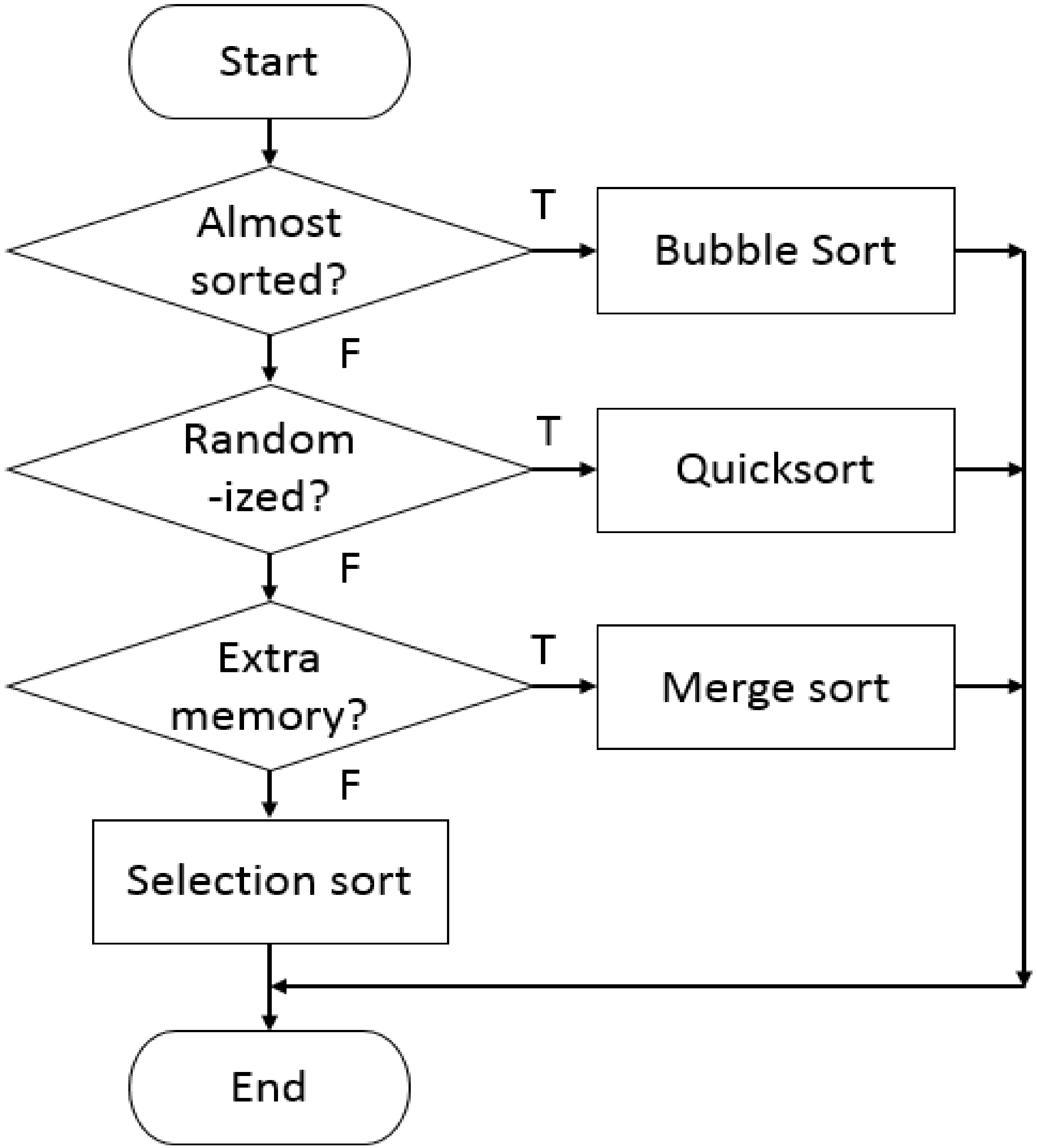


Why are quicksort swaps so effective?

- Bubblesort’s swaps aren’t very purposeful. They only swap with their neighbour. They will move to their correct location eventually.
- However, quicksort’s swaps move the element to the correct half of the array. Each swap is a big move towards its correct place.

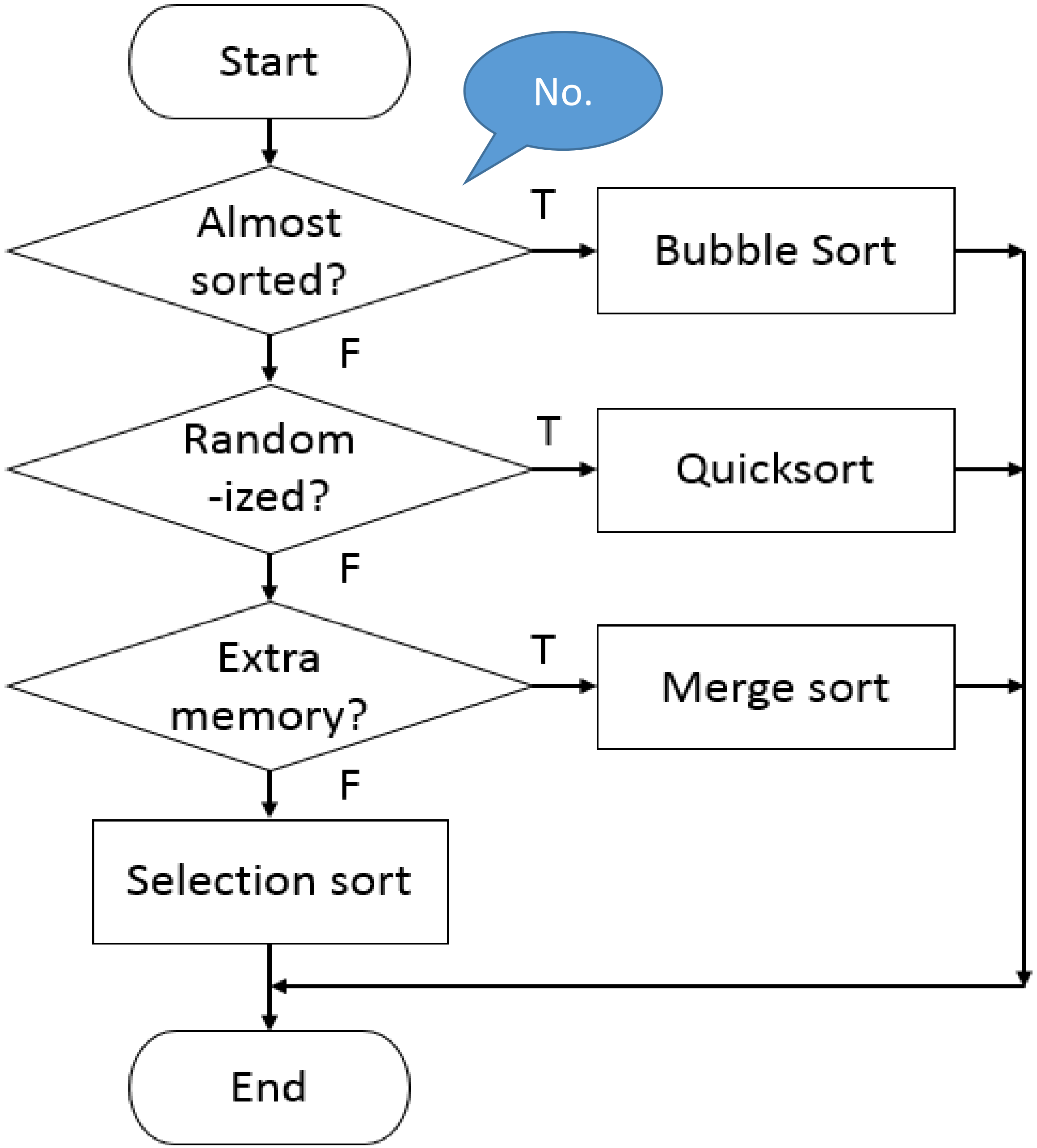
Which sort algorithm should you use?

You have an array that is 5 billion elements long. It is in random order. You have extra memory.



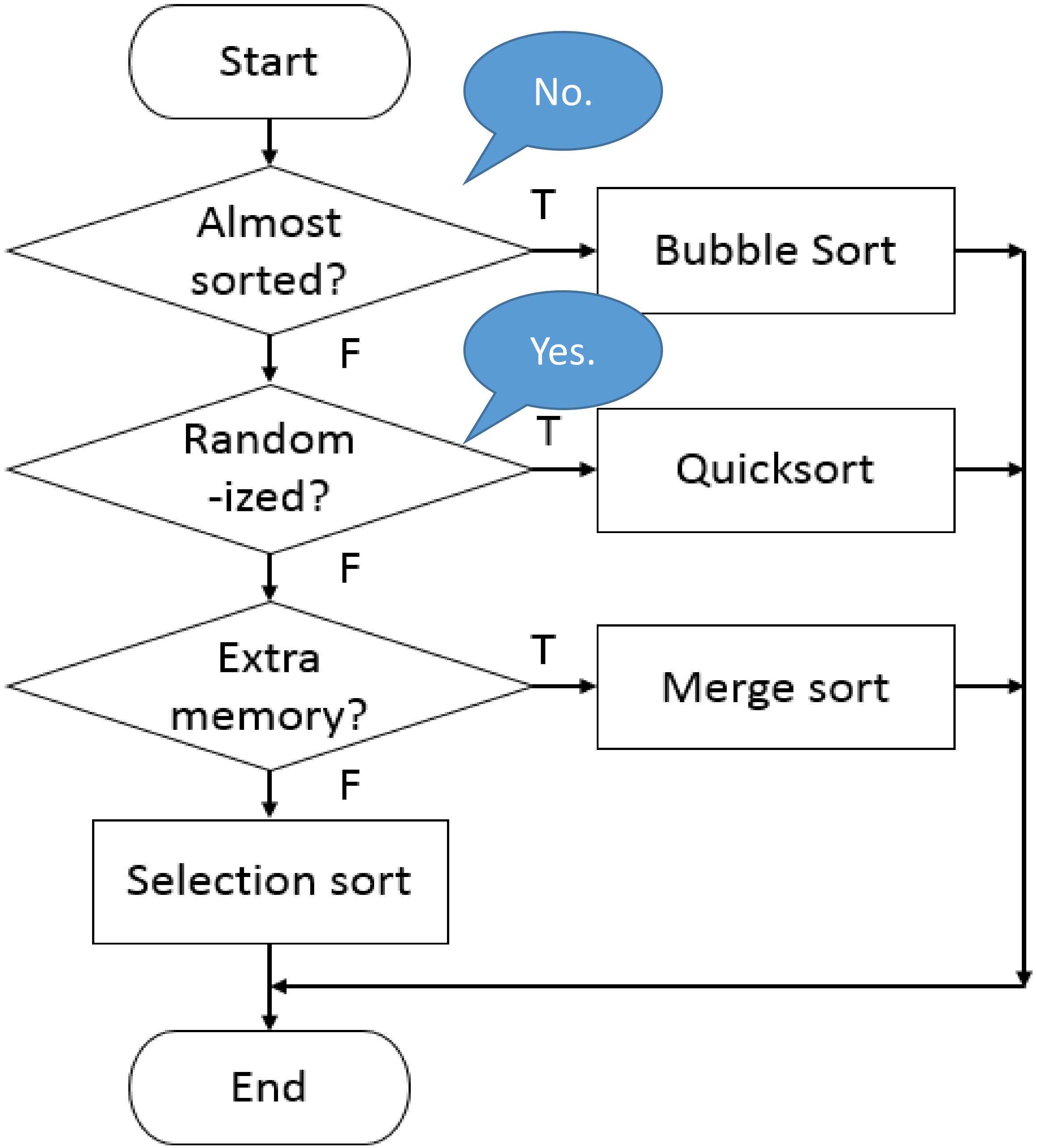
Which sort algorithm should you use?

You have an array that is 5 billion elements long. It is in random order. You have extra memory.



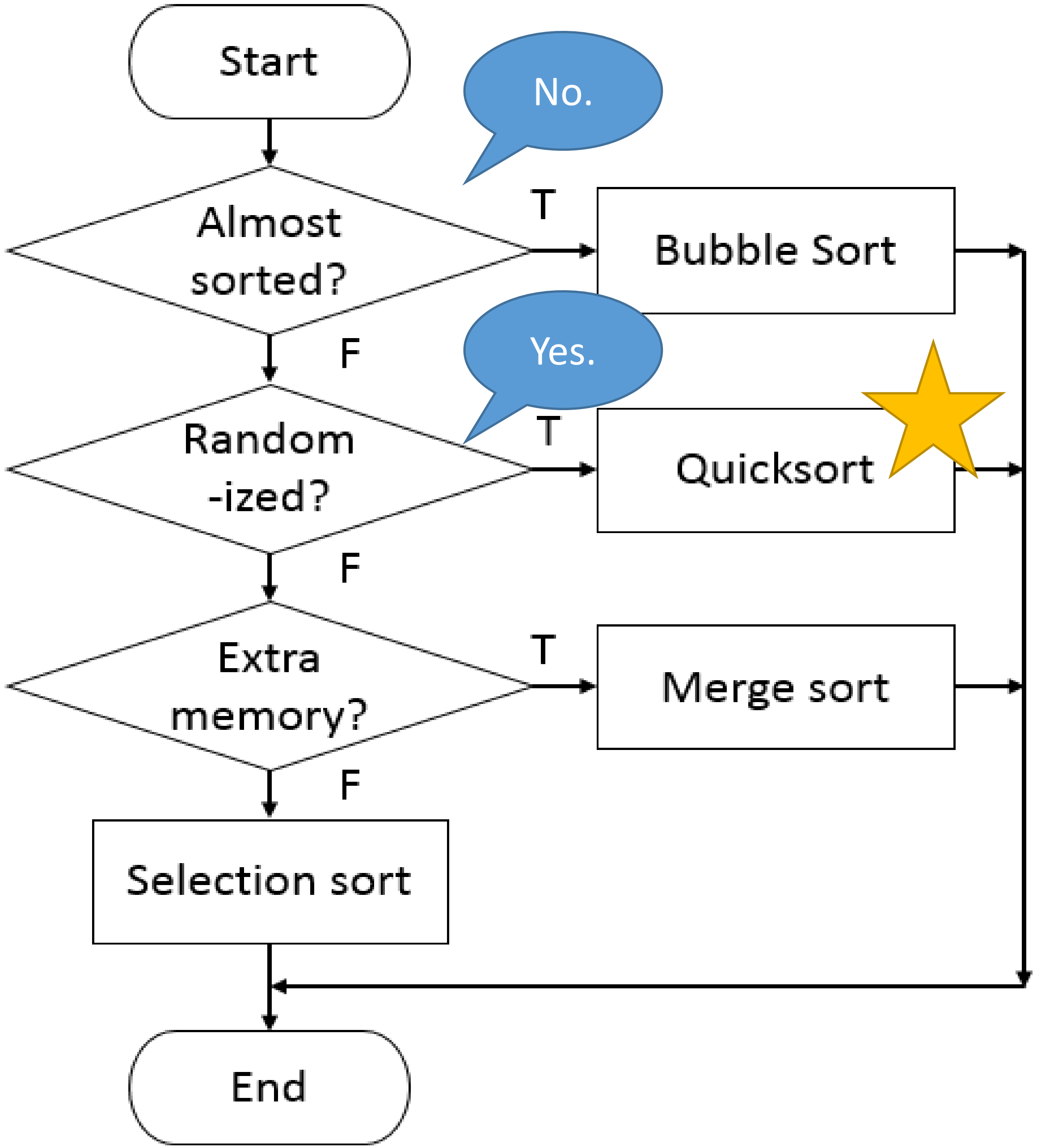
Which sort algorithm should you use?

You have an array that is 5 billion elements long. It is in random order. You have extra memory.



Which sort algorithm should you use?

You have an array that is 5 billion elements long. It is in random order. You have extra memory.



Which sort algorithm should you use?

You have an array that is 5 billion elements long. It is in random order. You have extra memory.