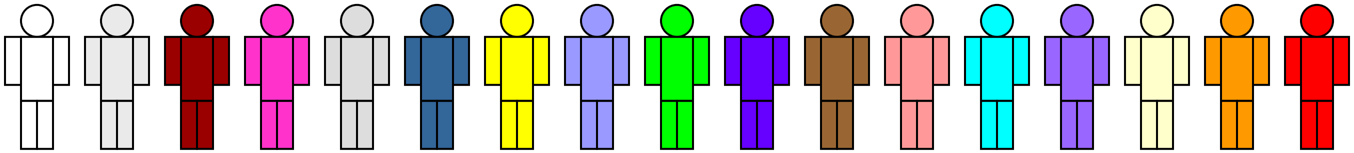
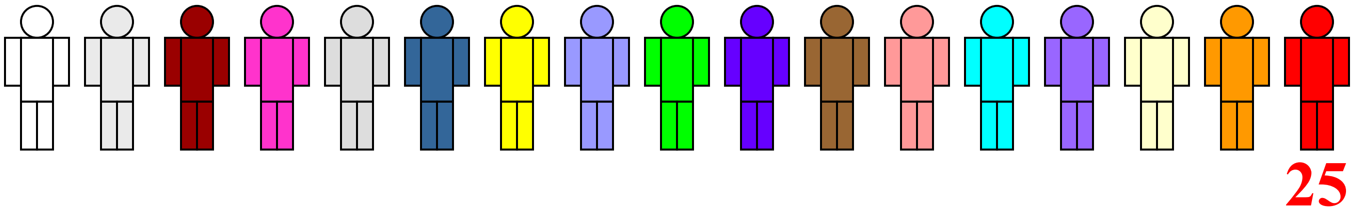
Quicksort

Ira A. Fulton Schools of Engineering
Arizona State University

High-level illustration of Quicksort

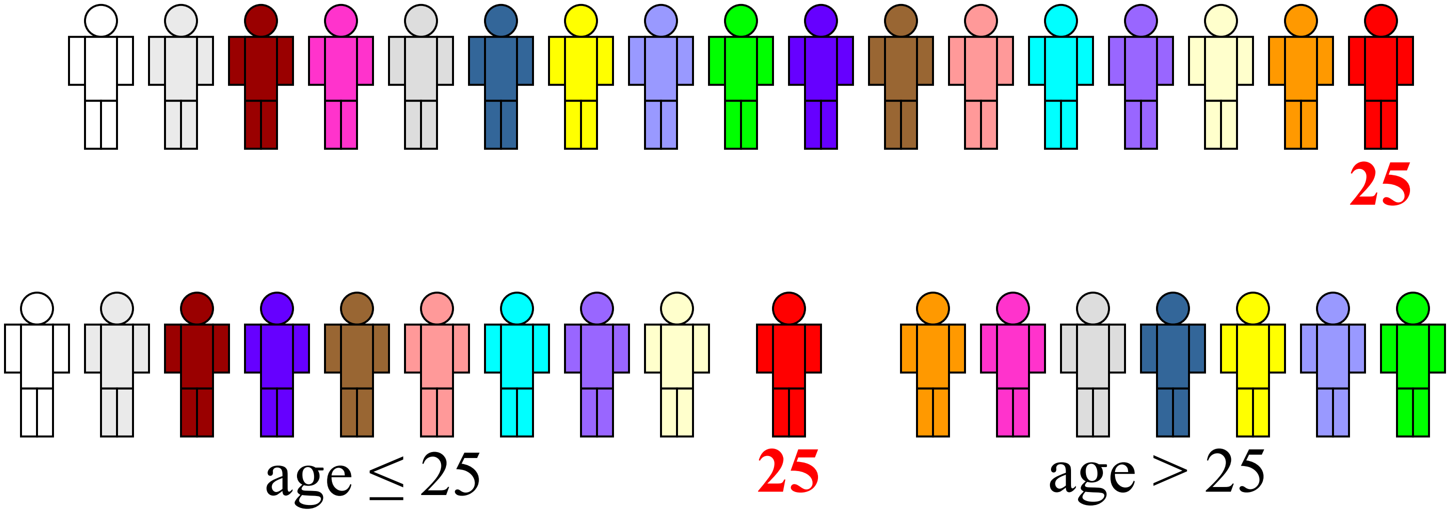


High-level illustration of Quicksort



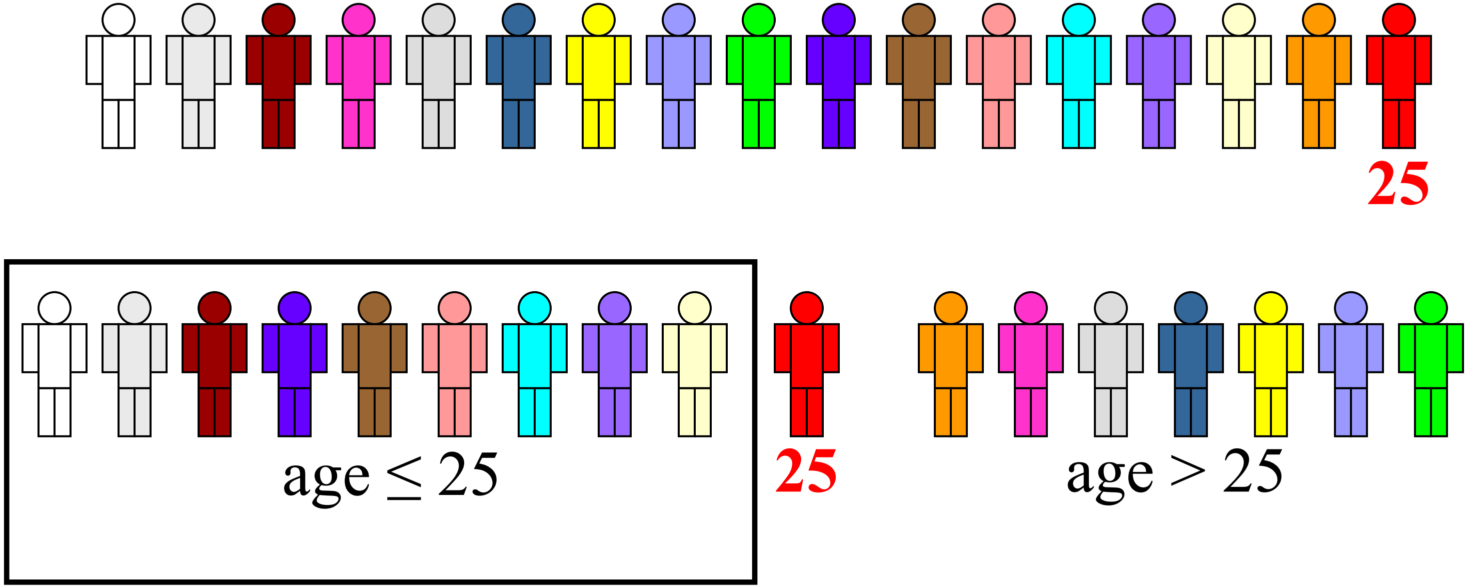
Select pivot

High-level illustration of Quicksort



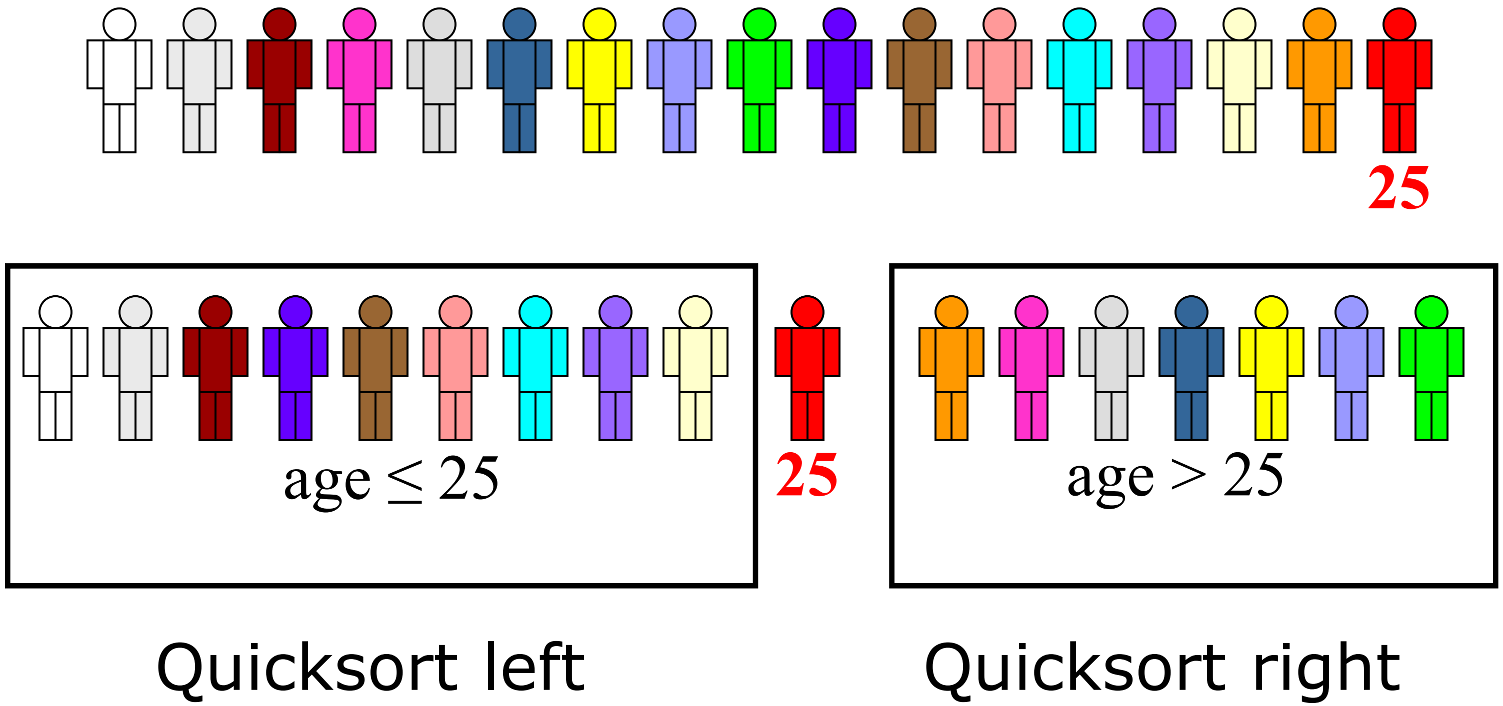
Partition

High-level illustration of Quicksort



Quicksort left

High-level illustration of Quicksort



The Algorithm

```
Quicksort(A, p, r)
if p < r
   q := Partition(A, p, r)
   Quicksort(A, p, q-1)
   Quicksort(A, q+1, r)
```

```
Partition(A, p, r)
x := A[r]
i := p-1
for j:=p to r-1 do
    If A[j] ≤ x then
        i := i+1
        temp := A[i]
        A[i] := A[j]
        A[j] := temp
temp := A[i+1]
A[i+1] := A[r]
A[r] := temp
return i+1
```

How Is *partition* Implemented?

Ideally, we can rearrange the list, so that half elements go to left and half elements go to right of *k*. In practice:

1. Choose $A[r]$ as the *pivot* element to go to the correct place in the list.
2. If p ≤ i ≤ q-1, then A[i] ≤ pivot;
3. If q+1 ≤ j ≤ r, then A[j] > pivot;

4. Scan from left to right, and do O(1) work at each position.

Example

p r

2, 8, 7, 1, 3, 5, 6, 4

x=4

i

j

Example

p r

2, 8, 7, 1, 3, 5, 6, 4

i

j

x=4

Example

p r

2, 8, 7, 1, 3, 5, 6, 4 x=4

i

j

Example

p r

2, 8, 7, 1, 3, 5, 6, 4

i

j

x=4

Example

p r

2, 8, 7, 1, 3, 5, 6, 4 x=4

i

j

Example

p r

2, 8, 7, 1, 3, 5, 6, 4 x=4

i

j

Example

p r

2, 8, 7, 1, 3, 5, 6, 4 x=4

i

j

Example

p r

2, 8, 7, 1, 3, 5, 6, 4

x=4

i

j

Example

p r

2, 8, 7, 1, 3, 5, 6, 4

i

j

x=4

Example

p r

2, 8, 7, 1, 3, 5, 6, 4

i

j

x=4

Example

p r

2, **1**, 7, **8**, 3, 5, 6, 4

i

j

x=4

Example

p r

2, 1, 7, 8, 3, 5, 6, 4

i

j

x=4

Example

p r

2, 1, 7, 8, 3, 5, 6, 4 x=4

i

j

Example

p r

2, 1, **3**, 8, **7**, 5, 6, 4

x=4

i

j

Example

p r

2, 1, 3, 8, 7, 5, 6, 4

x=4

i

j

Example

p r

2, 1, 3, 8, 7, 5, 6, 4

x=4

i

j

Example

p r

2, 1, 3, 8, 7, 5, 6, 4

i

j

x=4

Example

p r

2, 1, 3, 8, 7, 5, 6, 4

i

j

x=4

Example

p r

2, 1, 3, 8, 7, 5, 6, 4

x=4

i

j

Example

p r

2, 1, 3, **4**, 7, 5, 6, **8**

x=4

i

j

Example

2, 8, 7, 1, 3, 5, 6, 4

2, 8, 7, 1, 3, 5, 6, 4

2, 8, 7, 1, 3, 5, 6, 4

2, 8, 7, 1, 3, 5, 6, 4

2, 1, 7, 8, 3, 5, 6, 4

2, 1, 3, 8, 7, 5, 6, 4

2, 1, 3, 8, 7, 5, 6, 4

2, 1, 3, 8, 7, 5, 6, 4

2, 1, 3, 4, 7, 5, 6, 8

x=4

Example

2, 8, 7, 1, 3, 5, 6, 4

x=4

2, 8, 7, 1, 3, 5, 6, 4

2, 8, 7, 1, 3, 5, 6, 4

2, 8, 7, 1, 3, 5, 6, 4

2, 1, 7, 8, 3, 5, 6, 4

2, 1, 3, 8, 7, 5, 6, 4

2, 1, 3, 8, 7, 5, 6, 4

2, 1, 3, 8, 7, 5, 6, 4

1, 2, 3, 4, 7, 5, 6, 8

Example

2, 8, 7, 1, 3, 5, 6, 4 x=4

2, 8, 7, 1, 3, 5, 6, 4

2, 8, 7, 1, 3, 5, 6, 4

2, 8, 7, 1, 3, 5, 6, 4

2, 1, 7, 8, 3, 5, 6, 4

2, 1, 3, 8, 7, 5, 6, 4

2, 1, 3, 8, 7, 5, 6, 4

2, 1, 3, 8, 7, 5, 6, 4

1, 2, 3, 4, 7, 5, 6, 8

Summary

- We have presented the quicksort algorithm
- We have illustrated the steps in partition
- We will study the time complexity of quicksort next

Ira A. Fulton Schools of
Engineering
Arizona State University

Time Complexity of Partition

The (best-case, worst-case) time complexity of partition is $\Theta(n)$, where n is the number of elements in the array to be partitioned.

We are scanning the array from left to right, spending $O(1)$ time at each position.

Time Complexity of Quick-Sort

The complexity depends on the partition.

Best-case partitioning: divide into 2 equal sublists

$$T(n) = \begin{cases} \Theta(1) & if\ n = 1 \\ 2T(n/2) + \Theta(n) & otherwise \end{cases} \quad \Rightarrow \quad T(n) = \Theta(n\lg n)$$

Worst-case partitioning: if the list is already sorted:

The right sublist: 0, and left sublist: *n* - 1 elements

$$T(n) = T(n-1) + \Theta(n) = T(n-2) + \Theta(n-1) + \Theta(n)$$

$$= \cdots = \sum_{k=1}^{n} \Theta(k) = \Theta \sum_{k=1}^{n} (k) = \Theta\left(\frac{n(n+1)}{2}\right) = \Theta(n^2)$$

Average Time Complexity of Quicksort

- The pivot A[r] partitions the array A[1:n] into a left part of q-1 elements and a right part of n-q elements.
- We call this a (*q-1*)-to-(*n*-*q*) split.
- The number q can take any of the integers between 1 and n.
- Since all permutations are equally likely, the probability for q to be equal to k is 1/n, for k=1, 2, ..., n.

Average Time Complexity of Quicksort

- $T(n)=n+1+2(T(0)+T(1)+\ldots+T(n-1))/n$
- $nT(n)=n(n+1)+2(T(0)+T(1)+\ldots+T(n-1))$
- $(n-1)T(n-1)=(n-1)n+2(T(0)+T(1)+\ldots+T(n-2))$
- $nT(n)-(n-1)T(n-1)=2n+2T(n-1)$
- $nT(n)=2n+(n+1)T(n-1)$
- $T(n)/(n+1)=2/(n+1)+T(n-1)/n$
- $T(n)=(n+1)2(1+1/2+1/3+\ldots+1/(n+1)) = \Theta(n \log n)$

Stable Sorting and Non-Stable Sorting

- A sorting algorithm is stable if A[i]=A[j] implies the relative order of the two elements will not change during the sorting process.
- Insertion sort is stable.
- MergeSort is stable
- Quicksort is NOT stable.

Tony Hoare and Quicksort

- Invented by Tony Hoare in 1959
- Divide and Conquer algorithm

- Selecting the pivot
- Using the pivot to partition

- Wiki page

Tony Hoare

ASU®
Ira A. Fulton Schools of
Engineering
Arizona State University