CSC 212
Data Structures and Abstractions
Fall 2015

Lecture 13: QuickSort
Quicksort
Developed by Tony Hoare in 1959, with his work published in 1961

http://www.bl.uk/voices-of-science/interviewees/tony-hoare/audio/tony-hoare-inventing-quicksort
QuickSort

Shuffle entire array

Pick a Pivot and Partition array in two parts

Sort recursively each part
Partition

Pick a **Pivot** $A[j]$ and organize items such that:

- all items to the left of $j$ are $\leq$ **Pivot**
- all items to the right of $j$ are $\geq$ **Pivot**
void r_quicksort(ul_int *A, ul_int lo, ul_int hi) {
    // base case
    if (hi <= lo) return;
    // partition
    ul_int p = partition(A, lo, hi);
    // recursively sort parts (avoid negative indices)
    if (p > 0) r_quicksort(A, lo, p-1);
    r_quicksort(A, p+1, hi);
}
How to partition?

Shuffle
How to partition?
How to partition?

Pick a pivot
How to partition?

while $A[i] < \text{pivot}$, increase $i$

while $A[j] > \text{pivot}$, decrease $j$
How to partition?

while $A[i] < \text{pivot}$, increase $i$
while $A[j] > \text{pivot}$, decrease $j$
How to partition?

| 13 |  1 | 31 | 20 | 10 |  4 | 22 | 15 |

while $A[i] < \text{pivot}$, increase $i$
while $A[j] > \text{pivot}$, decrease $j$
How to partition?

while A[i] < pivot, increase i
while A[j] > pivot, decrease j
How to partition?

```
13  1  31  20  10  4  22  15
```

swap A[i] and A[j]

i    j
How to partition?

| 13 | 1  | 4  | 20 | 10 | 31 | 22 | 15 |

while $A[i] < \text{pivot}$, increase $i$

while $A[j] > \text{pivot}$, decrease $j$
How to partition?

while $A[i] < \text{pivot}$, increase $i$
while $A[j] > \text{pivot}$, decrease $j$
How to partition?

```
while A[i] < pivot, increase i
while A[j] > pivot, decrease j
```
How to partition?

\[ \text{swap } A[i] \text{ and } A[j] \]
How to partition?

while $A[i] < \text{pivot}$, increase $i$
while $A[j] > \text{pivot}$, decrease $j$
How to partition?

while $A[i] < \text{pivot}$, increase $i$

while $A[j] > \text{pivot}$, decrease $j$
How to partition?

while $A[i] < \text{pivot}$, increase $i$
while $A[j] > \text{pivot}$, decrease $j$
How to partition?

when i and j cross, swap A[j] and pivot
How to partition?

\[
\begin{array}{cccccccc}
10 & 1 & 4 & 13 & 20 & 31 & 22 & 15 \\
\end{array}
\]

\[
\begin{array}{c}
j \\
i \\
\end{array}
\]

when i and j cross, swap A[j] and pivot
How to partition?

array partitioned!
ul_int partition(ul_int *A, ul_int lo, ul_int hi) {
    ul_int temp, i = lo, j = hi + 1;
    while (1) {
        // while A[i] < pivot, increase i
        while (A[++i] < A[lo]) if (i == hi) break;
        // while A[i] > pivot, decrease j
        while (A[lo] < A[--j]) if (j == lo) break;
        // if i and j cross exit the loop
        if (i >= j) break;
        // swap A[i] and A[j]
    }
    // swap the pivot with A[j]
    // return pivot's position
    return j;
}
Analysis

One *partition* operation + Two *recursive* calls
Analysis

One \textit{partition} operation + Two \textit{recursive} calls

Best case

\[ T(n) = O(n \log n) \] — why?
Analysis

One **partition** operation + Two **recursive** calls

**Best case**
\[ T(n) = O(n \log n) \] — why?

**Worst case**
\[ T(n) = O(n^2) \] — why?
Analysis

One **partition** operation + Two **recursive** calls

**Best case**

\[ T(n) = O(n \log n) \] — why?

**Worst case**

\[ T(n) = O(n^2) \] — why?

**Average case**

\[ T(n) = O(n \log n) \]
Properties

QuickSort does not require additional array aux
Properties

QuickSort does not require additional array aux

QuickSort is in-place
Properties

QuickSort does not require additional array aux
QuickSort is in-place
QuickSort is not stable
# Cost of Sorting

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Best-Case</th>
<th>Average-Case</th>
<th>Worst-Case</th>
<th>In-place</th>
<th>Stable</th>
</tr>
</thead>
<tbody>
<tr>
<td>BubbleSort</td>
<td>$O(n^2)$</td>
<td>$O(n^2)$</td>
<td>$O(n^2)$</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>InsertionSort</td>
<td>$O(n)$</td>
<td>$O(n^2)$</td>
<td>$O(n^2)$</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>SelectionSort</td>
<td>$O(n^2)$</td>
<td>$O(n^2)$</td>
<td>$O(n^2)$</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>MergeSort</td>
<td>$O(n \log n)$</td>
<td>$O(n \log n)$</td>
<td>$O(n \log n)$</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>QuickSort</td>
<td>$O(n \log n)$</td>
<td>$O(n \log n)$</td>
<td>$O(n^2)$</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>