Quicksort

In this lecture:

- quicksort algorithm
- worst, best and average performance

Quicksort algorithm

- Designed by C. A. R. Hoare in 1962.
- Recursive algorithm:
- choose an element in the array (*pivot element*)
- partition the array in two parts:

numbers less than pivot+pivot+numbers greater than pivot

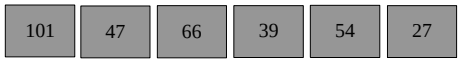
- return the division index.
- sort each part (using quicksort)

Divide and Conquer Algorithm

Divide and conquer algorithms just as merge sort.

- split the problem into parts (“divide”). Merge sort just divides the array in halves. Quick sort partitions the array using a pivot.
- solve the subproblems (call sort again on them).
- combine them to produce the solution. Merge sort uses merge procedure to do this. Quick sort does all the work in the partition and does not have to do anything special to combine results.

Partition



Pivot: 66

Desired result:

Return 4 (the index of 66)

Recursion

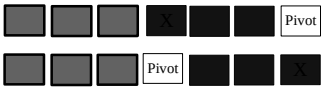
```
public void recQuickSort(int[]arr,int l,int r){
   if (r - l <= 0) return;
  // base case
   else {
      int border = partition(arr, l, r);
      recQuickSort(arr, l, border-1);
      recQuickSort(arr, border+1, r);
   }
}
```

Partition algorithm

- The algorithm is a bit fiddly, especially if you want to make it efficient
- All we want at the moment is to make it correct and working in linear time.
- Some times it is referred to as “Dutch flag” algorithm. The idea is that the array contains “red” items, “blue” items and “white” items and they need to be efficiently collected in separate parts of the array.

Partition algorithm

- Pick a pivot
- put the pivot out of the way (e.g. swap it to the end of the array)
- do the Dutch flag routine on the rest of the array:
- swap the pivot with the leftmost blue element:



Adaptation of the Dutch flag routine

- Red - all numbers less than the pivot.
- Blue - all numbers greater or equal to the pivot.
- just split in two parts (could have separated numbers which are equal to the pivot - would correspond to white).
- *Loop invariant*: everything to the left of the **`red`** counter is Red, everything starting from the **`blue`** counter is Blue. Loop invariant is a property which holds before the loop is executed and after each iteration through the loop.

Partition algorithm

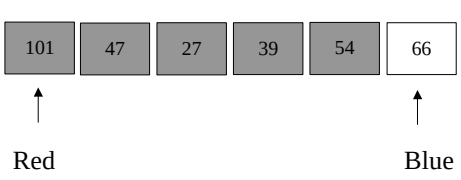
```
red = l;  // set at the left border of the
          // range
blue = r; //set at the right border where the
          // pivot sits

while(red < blue) {
   if (arr[red] < pivot) red++;
   else {
        blue--;
        swap(arr, red, blue);
    }
}
swap(arr, blue, r); // put the pivot on the
                    // border
return blue;
```

Illustration

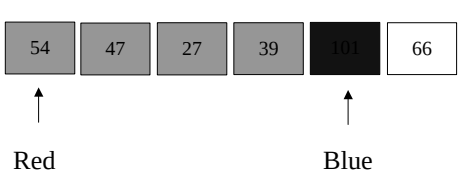
Select the pivot (66)

Swap the pivot

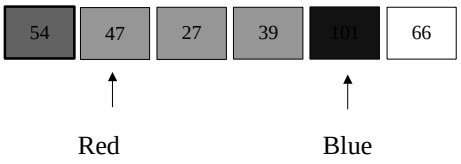
Illustration



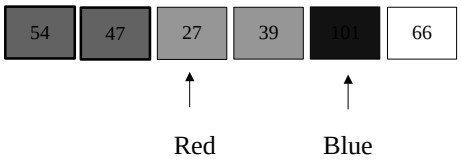
Illustration



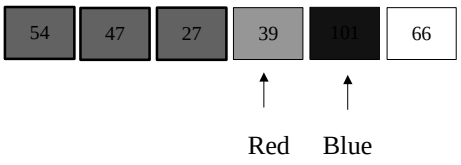
Illustration



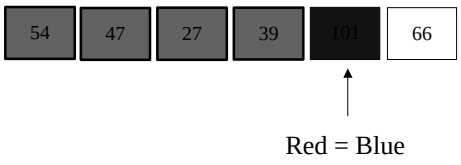
Illustration



Illustration



Illustration



Illustration

Swap the pivot

Return the border (4)

Choosing the pivot

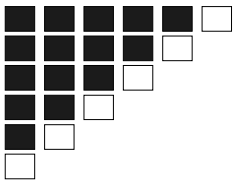
- If we happen to pick the pivot which is always the largest or the smallest element, quicksort works just like selection sort
- If we always pick the element with the median value, it splits the array in half at every recursion level
- Hence: best case is O(N log N), worst case quadratic.

Choosing the pivot

- Pick first, last, or middle element: works fine with random arrays
- Generate a random index
- Pick a median of three values
- ...

Complexity of quicksort

- Worst case: $O(N^2)$
  - the pivot is always the greatest (the least) element
  - at each recursive call the array is split into a part where all elements smaller than the pivot are, the pivot, and an empty part.



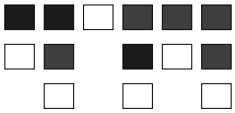
red - numbers smaller than the pivot,

white - the pivot,

blue - numbers greater than the pivot

Complexity of quicksort

- Best case: O(N log N)
  - the pivot is always the median element
  - at each recursive call the array is split into two equal parts, elements smaller than the pivot and elements larger than the pivot.



Complexity of quicksort

- Average case: ?

Complexity of quicksort

- Average case:
  - Complicated proof using recurrences: Shaffer Chapter 14
  - Hand-waving proof: if the pivot is chosen at random, what is its average expected value?
  - about the median of all values in that part of the array
  - hence on average the array is split in about equal parts
  - hence quicksort in the average case behaves as in the best case: O(N log N).

Comparison to merge sort

- Quick sort does not use extra space unlike merge sort
- Quick sort is not guaranteed to have O(n log n) performance in the worst case, unlike merge sort
- Question: can we do better than O(n log n)?
- We'll see later in the course a proof that comparison sorting cannot be done with fewer than O(n log n) comparisons.

Reading

- Shaffer, chapter 8 on quick sort.