Quick Sort

- To sort a subarray A[p..r]
  - Choose an element from the subarray as a *pivot*
- Divide:
  - Partition the array A[p..r] into two (possibly empty) subarrays A[p.. q-1] and A[q+1..r] such that each element of A[p..q-1] <= A[q] (pivot) and each element of A[q+1..r] >= A[q] (pivot)
- Conquer
  - Recursively sort the two subarrays A[p..q-1], A[q+1..r]
- Combine: no work needs to be done.

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

```
int partition(A, p, r)
{
   x = A[r]; // use A[r] as the pivot
   i = p-1;
   for (j=p; j<=r-1; j++) {
      if (A[j] <= x) {
         i++;
         exchange(A[i], A[j]);
      }
   }
   exchange(A[i+1], A[r]);
   return i+1;
}
```

Example for Partition

i	j							
	2	8	7	1	3	5	6	4
	i	j						
	2	8	7	1	3	5	6	4
	i		j					
	2	8	7	1	3	5	6	4
	i			j				
	2	8	7	1	3	5	6	4
		i			j			
	2	1	7	8	3	5	6	4

Loop invariant: A[p..i] ≤ x < A[i+1..j-1]
at the beginning of iteration

Example for Partition

		i			j		
2	1	3	8	7	5	6	4

		i				j	
2	1	3	8	7	5	6	4

		i					j
2	1	3	8	7	5	6	4

State when loop finishes

		i					j
2	1	3	4	7	5	6	8

Final swap after loop

Analysis

- Worst case: the array is sorted, $\Omega(n^2)$
- Best case
  - $T(n) <= 2T(n/2) + \Theta(n)$, $T(n) \in O(n \log n)$
- Average case

Average case

- Assume the pivot chosen lies in any position with equal probability
- Let $t(m)$ be average time taken to sort m-elelemt array by quick sort.
- The partition operation takes linear time $g(n) \in \Theta(n)$. We know $\exists d > 0, n_0 \in N, \forall n \geq n_0, g(n) \leq dn$
- We obtain the following recurrence

$$t(n) = \frac{1}{n}\sum_{l=1}^{n}(g(n) + t(l-1) + t(n-l))$$

$$\leq dn + \frac{1}{n}\sum_{l=1}^{n}(t(l-1) + t(n-l)) \qquad for \qquad n \geq n_0$$

Constructive induction

- We prove that $t(n) \in O(n \log n)$
  - We need to show $\exists c > 0, n_1 \in N, \forall n \geq n_1, t(n) \leq cn \log n$
  - We choose $n_1 = 2$
  - We have chosen $n_0 >= 2$ and d such that

$$t(n) \leq dn + \frac{1}{n}\sum_{l=1}^{n}(t(l-1) + t(n-l)) \qquad for \qquad n \geq n_0$$

$$= dn + \frac{2}{n}\sum_{k=0}^{n-1} t(k)$$

Induction step

$$t(n) \le dn + \frac{2}{n}\sum_{k=0}^{n-1} t(k)$$

$$= dn + \frac{2}{n}(t(0)+t(1)) + \frac{2}{n}\sum_{k=2}^{n-1} t(k)$$

$$\le dn + \frac{2a}{n} + \frac{2}{n}\sum_{k=2}^{n-1} ck\log k$$

$$\le dn + \frac{2a}{n} + \frac{2c}{n}\int_2^n x\log x\,dx$$

$$= dn + \frac{2a}{n} + \frac{2c}{n}\left[\frac{x^2\log x}{2} - \frac{x^2}{4}\right]_{x=2}^{n}$$

$$< dn + \frac{2a}{n} + \frac{2c}{n}\left(\frac{n^2\log n}{2} - \frac{n^2}{4}\right)$$

$$= cn\log n - \left(\frac{c}{2} - d - \frac{2a}{n^2}\right)n$$

We need

$$\frac{c}{2} - d - \frac{2a}{n^2} > 0$$

$$c \ge 2d + \frac{4a}{n^2}$$

We know if $n > n_0$

$$c \ge 2d + \frac{4a}{(n_0+1)^2}$$

Induction basis and Induction hypothesis

- Basis: consider any integer n such that $2 \le n \le n_0$. We can choose c such that
  - $c >= t(n)/(n\log n)$ for all n such that $2 \le n \le n_0$
  - Note that we can do this only because is n_0 fixed, i.e., it's not a function of n
- Hypothesis: Assume the induction hypothesis that $t(k) <= c\, k \log k$ for all k such that $2 <= k < n$.