Average Case Analysis

Beyond Worst Case Analysis

Worst-case analysis.
- Analyze running time as function of worst input of a given size.

Average case analysis.
- Analyze average running time over some distribution of inputs.
- Ex: quicksort.

Amortized analysis.
- Worst-case bound on sequence of operations.
- Ex: splay trees, union-find.

Competitive analysis.
- Make quantitative statements about online algorithms.
- Ex: paging, load balancing.

Average Case Analysis

Average case analysis.
- Analyze average running time over some distribution of inputs.
- Ex: quicksort.
  - O(N log N) if input is assumed to be in random order.
  - leads to randomized algorithm with O(N log N) expected running time, independent of input
- Major disadvantage: hard to quantify what input distributions will look like in practice.

Quicksort

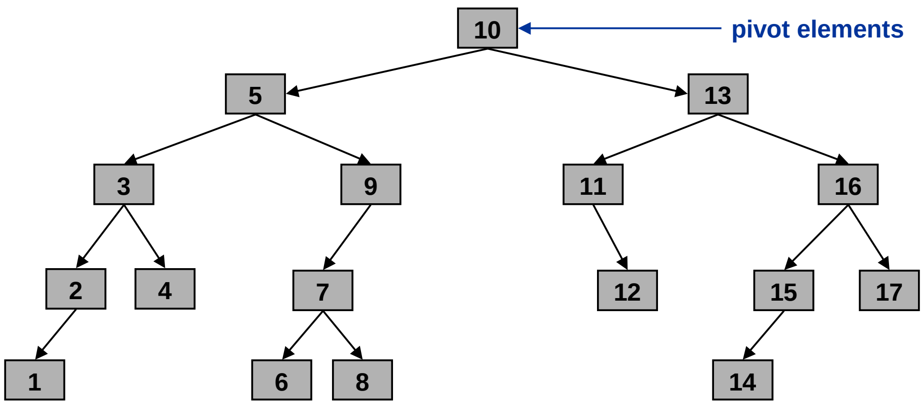
Quicksort.
- Assume all elements are unique.
- Assume input is a random permutation of inputs.
- Denote i^{th} largest element by i.

quicksort.c

```
void quicksort(Item a[], int left, int right) {
  int m;
  if (right > left) {
    m = partition(a, left, right);
    quicksort(a, left, m - 1);
    quicksort(a, m + 1, right);
  }
}
```

Quicksort: BST Representation of Pivots

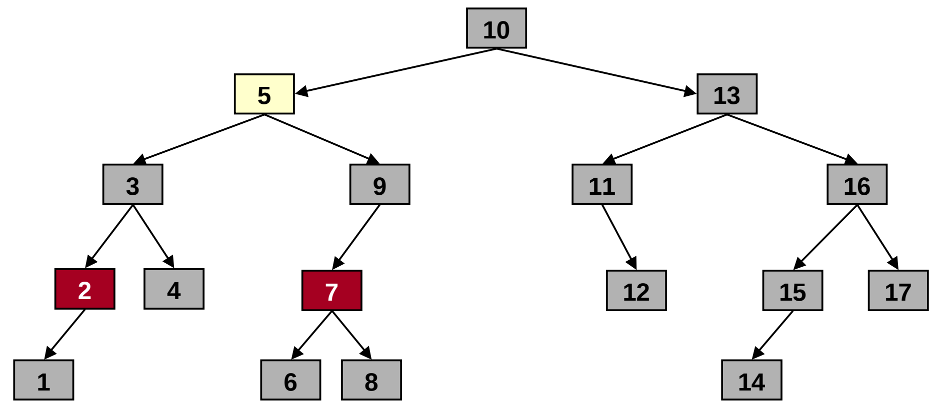
7	6	12	3	11	8	7	1	15	13	17	5	16	14	9	4	10
7	6	4	3	9	8	2	1	5	10	17	15	16	14	11	12	13
1	2	4	3	5	8	6	7	9	10	12	11	13	14	15	17	16
1	2	3	4	5	8	6	7	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17



Quicksort: Average Case Analysis

Probability that i = 2 and j = 7 get compared.

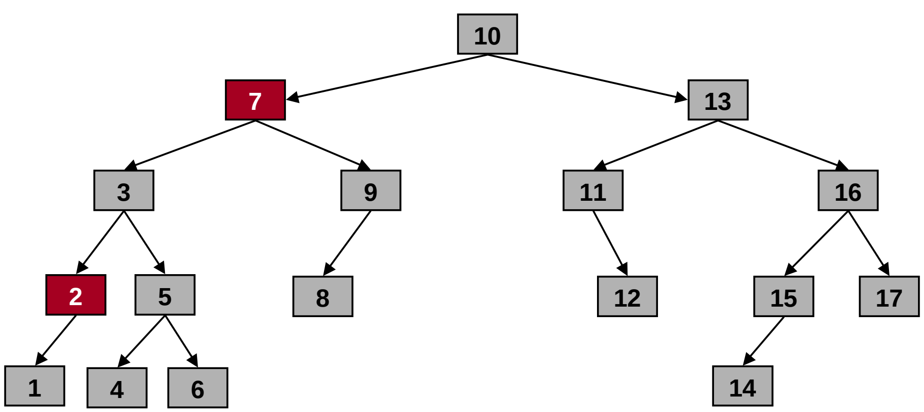
- Let x be pivot element that separates i and j.
- Case 1: $x \in \{3, 4, 5, 6\}$ $\Rightarrow$ i and j not compared.



Quicksort: Average Case Analysis

Probability that i = 2 and j = 7 get compared.

- Let x be pivot element that separates i and j.
- Case 1: $x \in \{3, 4, 5, 6\}$ $\Rightarrow$ i and j not compared.
- Case 2: $x \in \{2, 7\}$ $\Rightarrow$ i and j are compared.



Quicksort: Average Case Analysis

Probability that i = 2 and j = 7 get compared.

- Let x be pivot element that separates i and j.
- Case 1: $x \in \{3, 4, 5, 6\}$ $\Rightarrow$ i and j not compared.
- Case 2: $x \in \{2, 7\}$ $\Rightarrow$ i and j are compared.

$$\Pr[\, i \text{ and } j \text{ compared } \ (i < j)\,] = \frac{2}{j - i + 1}$$

Quicksort: Average Case Analysis

Quicksort: average case analysis.

$$\Pr[\,i \text{ and } j \text{ compared } (i<j)\,] = \frac{2}{j-i+1}$$

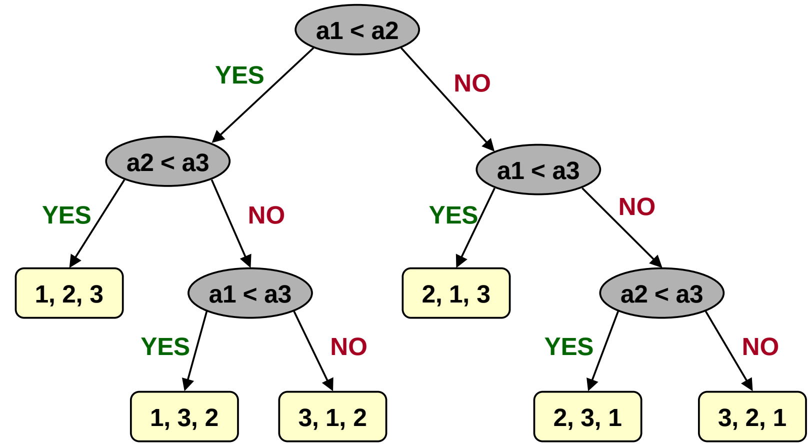
- Expected # of comparisons $= \displaystyle\sum_{1 \le i < j \le N} \frac{2}{j-i+1}$

$$= 2\sum_{i=1}^{N}\sum_{j=2}^{i}\frac{1}{j}$$
$$\le 2N\sum_{j=1}^{N}\frac{1}{j}$$
$$\approx 2N\int \frac{1}{j}$$
$$= 2N \ln N$$

Remark: same analysis if we choose partition element uniformly at random.

- Running time independent of input.

Comparison Based Sorting Lower Bound



Decision Tree of Program

Comparison Based Sorting Lower Bound

Lower bound = $\Omega(N \log_2 N)$: applies to comparison-based algorithms.

- Tree height h determines worst case running time.
- N! different input orderings.
- One (or more) leaves corresponds to each ordering.
- Binary tree with N! leaves has height:

$$\begin{aligned} h &\ge \log_2 (N!) \\ &\ge \log_2 (N/e)^N \\ &= N \log_2 N - N \log_2 e \\ &= \Omega(N \log_2 N) \end{aligned}$$

Stirling's formula