Types of Case Analysis

Average case

- Number of steps required for “typical” case
- Most useful metric in practice
- Different approaches
  - Average case
  - Expected case
Approaches to Average Case

- **Average case**
  - Average over all possible inputs
    - Assumes all inputs have the same probability
  - **Example**
    - Case 1 = 10 steps, Case 2 = 20 steps
    - Average = 15 steps

- **Expected case**
  - Weighted average over all possible inputs
    - Based on probability of each input
  - **Example**
    - Case 1 (90%) = 10 steps, Case 2 (10%) = 20 steps
    - Average = 11 steps
Average Case Example

Example problem

Average # of comparisons needed to find a number in the (sorted) array $A[\ ] = \{1, 4, 8, 12, 15\}$ using

1. Linear search
   - Start from beginning, compare elements one at a time

2. Binary search
   - Start from middle of array at index $k$, compare element
   - If not element, repeat for top or bottom half of remaining array depending on whether element is smaller or greater than $A[k]$
Average Case: Linear Search

Algorithm

1. Find # of comparisons needed for each case
   - 1 → 1 comparison (1)
   - 4 → 2 comparisons (1, 4)
   - 8 → 3 comparisons (1, 4, 8)
   - 12 → 4 comparisons (1, 4, 8, 12)
   - 15 → 5 comparisons (1, 4, 8, 12, 15)

2. Calc average = total # of comparisons / # cases
   - Total # comparisons = 1 + 2 + 3 + 4 + 5 = 15
   - # cases = 5
   - Average = 3 comparisons / number
Average Case: Binary Search

Algorithm

1. Find # of comparisons needed for each case
   - 1  → 3 comparisons (8, 4, 1)
   - 4  → 2 comparisons (8, 4)
   - 8  → 1 comparisons (8)
   - 12 → 2 comparisons (8, 12)
   - 15 → 3 comparisons (8, 12, 15)

2. Calc average = total # of comparisons / # cases
   - Total # comparisons = 3 + 2 + 1 + 2 + 3 = 11
   - # cases = 5
   - Average = 2.2 comparisons / number
Average Case Example

Example problem 2

Average # of comparisons needed to find a number in a sorted array A[ n ] of size n using

1. Linear search
2. Binary search

For simplicity, we assume elements are stored in A[1] ... A[n]
Average Case: Linear Search

Algorithm

1. Find # of comparisons needed for each case
     ...

2. Calc average = total # of comparisons / # cases
   - Total # comparisons = 1 + 2 + ... + n = ½ n² + 1
   - # cases = n
   - Average ≈ ½ n comparisons / number
Average Case: Binary Search

Algorithm

1. Find # of comparisons needed for each case
   - $A[n/2]$ → 1 comp $(A[n/2])$
   ... 
   - $A[1], A[3]...A[n]$ → $\log_2(n)$ comparisons 
     $(A[n/2], A[n/4], A[n/8]...A[1])$

2. Calc average = total # of comparisons / # cases
   - Total # comparisons = $n/2 \times \log_2(n) + n/4 \times \log_2(n)–1 + ... + 1 = n \log_2(n)$
   - # cases = $n$
   - Average $\approx \log_2(n)$ comparisons / number