Sorting

Jasleen Kaur

Spring 2014

Topic Overview

- Application for Sorting:
  - Binary Search
- Sorting Algorithms
  - Insertion Sort
  - Merge Sort
  - Quick Sort
  - Bucket Sort
  - External Sorting
BINARY SEARCH

Why is Sorting Useful?

Review: Linear Search in Sorted Array

- How to search for an element in sorted array?
  ```java
  boolean findInteger(int[] arr, int k) {
      for(int i=0; i < arr.length; i++)
          if (arr[i] == k)
              return true;
          else if (arr[i] > k)
              return false;
      return false;
  }
  ```

- What is the run-time complexity for this method?
  - Worst-case:
  - Average case:

- Will above routine work if data is not sorted?

- Can you think of ways to make this routine efficient?
Binary Search

```java
public static int binarySearch(int[] a, int x) {
    int low = 0, high = a.length - 1; // window
    while (low <= high) {
        int mid = (low + high)/2;
        if (a[mid] < x)
            low = mid + 1;
        else if (a[mid] > x)
            high = mid - 1;
        else
            return mid; // found
    }
    return NOT_FOUND; // defined as -1
}
```

Complexity of Binary Search

- **Stuff inside the loop:**
  - O(1)
  - So, complexity depends on number of times loop runs

- **Loop iterations analysis:**
  - Loop starts with: high – low = N-1
    - Ends with: high – low ≤ -1
  - Every time through loop
    - (high – low) at least halved from previous value
      - e.g., 128, 64, 32, 16, 8, 4, 2, 1, 0, -1
  - Number of loop iterations ≤ ceiling(log(N-1)) + 2

- **Run-time is:** O(log N)
How to Sort?

- **Applications of sorting:**
  - Clearly, searching in a sorted array can be more efficient
    - $O(\log N)$ vs $O(N)$
  - Many databases list elements in a sorted order
    - e.g., alphabetical

- **How to sort?**
  - Insertion sort
  - Merge sort
  - Quick sort
  - Bucket sort
  - External sorting

Setup

- **Input:**
  - An array $A$ of data records
  - A key value in each data record
  - A comparison function

- **Output:**
  - Rearranged array $A$, such that for any $i < j$, $A[i] \leq A[j]$
### Criteria

- **Run-time:**
  - What is the run-time complexity of the algorithm?

- **Space:**
  - How much space does the sorting algorithm require for sorting?
    - Is copying needed? – O(N) additional space
    - In-place sorting – O(1) additional space

- **Stability:**
  - Does sorting retain input order for duplicates?
    - e.g., if names originally in alphabetical order, then sorted according to county, would names still be sorted within a county?
  - Important property for databases

### Note on Run-time Complexity

- **Speed of sorting:**
  - Output of sorting:
    - Rearranged array A, such that for any \( i < j \), \( A[i] \leq A[j] \)
  - Each element must be checked at least once
    - At least O(N)
  - If you check each element against every other element
    - O(N^2)
  - Efficient sorting – how close to O(N) can you get?

- **Examples:**
  - O(N^2) – insertion sort, bubble sort
  - O(N log N) – merge sort, quick sort
  - O(K N) – bucket sort, radix sort