Analysis, Searching, Sorting, and Assessment

Overview

Analysis of algorithms
Searching
- Linear search
- Binary search

Sorting
- Insertion sort
- Mergesort

Assessment

Analysis of algorithms

https://www.youtube.com/watch?v=w7-6h64HSQ8

Searching

Linear search

```
public static boolean linearSearch(int key, int[] data) {
    for (int i = 0; i < data.length; i++) {
        if (data[i] == key) {
            return true;
        }
    }
    return false;
}
```

Best case: $\Theta(1)$

Average case: $\Theta(n)$

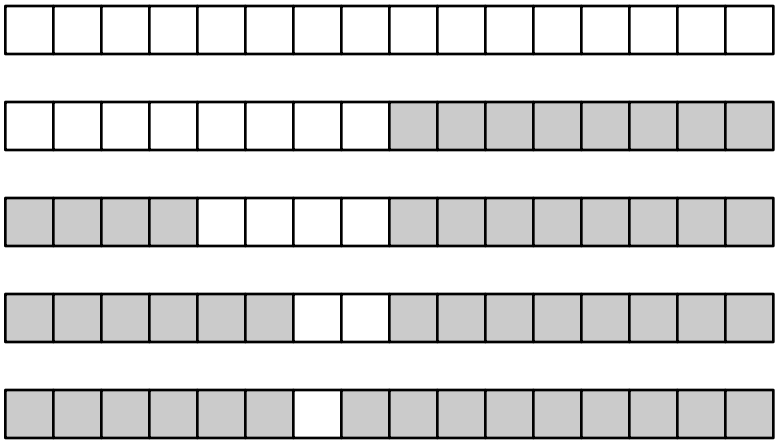
Worst case: $\Theta(n)$

Binary search

Requires that the array is already sorted!

```
public static boolean binarySearch(int key, int[] data) {
    int lo = 0;
    int hi = data.length;
    while (lo < hi) {
        int mid = lo + (hi - lo) / 2;
        if (key < data[mid]) {
            hi = mid;
        } else if (key == data[mid]) {
            return true;
        } else {
            lo = mid + 1;
        }
    }
    return false;
}
```

Analysis of binary search



Best case: $\Theta(1)$

Average case: $\Theta(\log n)$

Worst case: $\Theta(\log n)$

Sorting

Insertion sort

Best case: $\Theta(n)$

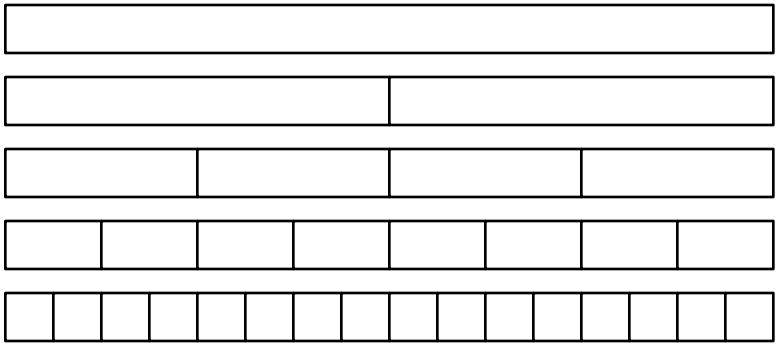
Average case: $\Theta(n^2)$

Worst case: $\Theta(n^2)$

Mergesort

For a recursive divide-and-conquer algorithm, we use time for:

- Dividing
- Recursively conquering subproblems
- Recombining results



Best case: $\Theta(n \log n)$

Average case: $\Theta(n \log n)$

Worst case: $\Theta(n \log n)$

Assessment

Feedback: is it working?

Mistakes aren’t bad as long as you learn from them.

Reflect on your level of understanding and confidence.

What works? What doesn’t? What can be improved?

Differentiate formative from summative assessment.

Review

Asymptotic notation concisely describes the efficiency of algorithms.

Binary search is faster than linear search for large n if the data are sorted.

Mergesort is faster than insertion sort for large n.

Assessment is a key part of improving one's work.