Outline

1. Example exam question
2. Sorting review
3. More examples
Outline

1. Example exam question
2. Sorting review
3. More examples
Example exam question

Prove or disprove:

a) \( \sqrt{n} = \Omega((\lg n)^2) \)

b) \( 2^n + (-2)^n + 1 = \Theta(2^n) \)

c) Let \( T(n) = \begin{cases} 1 & n = 1 \\ T([\sqrt{n}]) + 1 & n > 1 \end{cases} \)

Is \( T(n) = \mathcal{O}(\lg \lg n) \) ?

d) \( 2^{n+1} = \mathcal{O}(2^n) \)

e) \( \lg(f(n)) = \mathcal{O}(\lg(g(n))) \implies f(n) = \mathcal{O}(g(n)) \)

f) \( \max(f(n), g(n)) = \Theta(f(n) + g(n)) \)
Outline

1. Example exam question
2. Sorting review
3. More examples
**Insertion Sort**
- Simple
- In place sorting algorithm
- $O(n^2)$ worst case behaviour

**Merge Sort**
- Does not sort in place
- $\Theta(n \lg n)$ worst case behaviour
Heapsort

- Relies in the heap data structure
- Max-Heapify is used to “bubble” up the values
- Sorts in place
- $\Theta(n \lg n)$ worst case behaviour

Quicksort

- Very common in practice
- Works with a pivot element and partitioning
- $\Theta(n^2)$ worst case behaviour
- $\mathcal{O}(n \lg n)$ average case behaviour
Counting Sort

- Assumes all keys are in the interval $[0, k)$
- Works by counting the number of occurrences of each key
- $\Theta(n + k)$ worst case

Radix Sort

- Sorts a “column” at a time using a stable sorting algorithm
- $O(d(n + k))$ to sort $n d$-digit numbers in the worst case

Bucket sort

- Assumes input is uniformly distributed in $[0, 1)$
- Works by dividing input into buckets
- Expected running time $\Theta(n)$
Outline

1. Example exam question
2. Sorting review
3. More examples
Binary-Search($A$, $start$, $stop$, $key$)

1. ▶ Calculate the middle of the search-block
2. $i \leftarrow start + \lfloor (stop - start + 1)/2 \rfloor$
3. ▶ If we have failed in our search, we return a negative index
4. if $start = stop$ and $A[i] \neq key$
5. then return $-1$
6. ▶ If we have found the key, we return its index
7. if $A[i] = key$
8. then return $i$
9. ▶ Otherwise, we recurse in the correct part of the array
10. if $A[i] > key$
11. then Binary-Search($A$, $start$, $i - 1$, $key$)
12. else Binary-Search($A$, $i + 1$, $stop$, $key$)

Derive the recurrence and its worst-case upper bound.
Problem

(15p)

Assume that we a set of four digit numbers that we want to translate into integers 0, ..., 9:

1066 1789 1945 1600 1915 2005 1000

Consider two hash functions $\text{hashCode}_1(x) = x \mod 10$ and $\text{hashCode}_2(x) = \frac{x-(x \mod 1000)}{1000}$. Assume numbers arrive from left to right.

a) Draw the resulting hash table if we use $\text{hashCode}_1$ and linear probing to resolve collisions.

b) Draw the resulting hash table if we use $\text{hashCode}_2$ and chaining to resolve collisions.

c) With the additional knowledge that the input numbers are all years, which of the two hash functions would be the better choice for arbitrary input?