



Tutorial 3

Algorithms and Data Structures

Jonathan Cederberg <jonathan.cederberg@it.uu.se>

Friday, October 1st, 2010

Outline

- 1 Example
- 2 Sorting review
- 3 Hashing
- 4 One More Example
- 5 Third assignment



Another example exam question

Problem

Binary search is a way of effectively searching in a sorted array. Given the following implementation of binary search, derive a recurrence and its upper bound.



Another example exam question

Binary-Search($A, start, stop, key$)

- 1 ▷ Calculate the middle of the search-block
- 2 $i \leftarrow start + \lceil (stop - start + 1)/2 \rceil$
- 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($A, i + 1, stop, key$)

Insertion Sort

- Simple
- In place sorting algorithm
- $\mathcal{O}(n^2)$ worst case behaviour



Merge Sort

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



Heapsort

- Relies on 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
- $\mathcal{O}(d(n + k))$ 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 $\Theta(n)$



Example exam question

Problem

*What is the maximum number of times during the execution of quicksort that the largest element can be moved, for an array of N elements? Explain your answer in no more than **three** lines.*



Recall:

Partition(A, p, r)

```
1  $x \leftarrow A[r]$ 
2  $i \leftarrow p - 1$ 
3 for  $j \leftarrow p$  to  $r - 1$ 
4     do if  $A[j] \leq x$ 
5         then  $i \leftarrow i + 1$ 
6             exchange  $A[i] \leftrightarrow A[j]$ 
7 return  $i + 1$ 
```

Quicksort(A, p, r)

```
1 if  $p < r$ 
2     then  $q \leftarrow$  Partition( $A, p, r$ )
3         Quicksort( $A, p, q - 1$ )
4         Quicksort( $A, q + 1, r$ )
```



Hashing

Hashing is good. Google uses hashing. So should you.



Problem

(15p)

Assume that we a set of four digit numbers that we want to store in a hash table:

1066 1789 1945 1600 1915 2005 1000

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

- Draw the resulting hash table if we use hashCode_1 and linear probing to resolve collisions.
- Draw the resulting hash table if we use hashCode_2 and chaining to resolve collisions.
- With the additional knowledge that the input numbers are all years, which of the two hashfunctions would be the better choice for arbitrary input?

Assignment 3.1

- (i) Describe in *two sentences* what a stable sorting algorithm is.
- (ii) Which of the following sorting algorithms are stable: insertion sort, merge sort, heapsort, and quicksort?



Assignment 3.2

- (i) Complete the following table:

Expected time			
	Instruction		
Data structure	Insert	Search	Maximum
Linked List			
Array			
Hash Table			
Binary Search Tree		$\mathcal{O}(\lg n)$	

Make sure you fill in expected time, and *not* worst case. State any assumptions you make, if anything is unclear.

- (ii) With the help of your newly created table, describe a scenario for each data structure, where that particular data structure would be the best choice of the three. **Justify** your answer, and relate it to the table.
- (iii) Do (i) also for worst case, in a new table.