CS161: Algorithm Design and Analysis Recitation Section 4 (Week 5)
Stanford University Week of 1 February, 2016

Problem 1-1. (Counting Sort)

Algorithm 1 Counting-Sort

```
 1: function COUNTING-SORT($A, B, k$)
 2:     $C[0..k] \leftarrow$ new array of zeros
 3:     for $j \leftarrow 1$ to $A.length$ do
 4:         $C[A[j]] \leftarrow C[A[j]] + 1$
 5:     end for                ▷ $C[i]$: now contains the number of elements in A equal to i
 6:     for $i \leftarrow 1$ to $k$ do
 7:         $C[i] \leftarrow C[i] + C[i-1]$
 8:     end for               ▷ $C[i]$: contains the number of elements in A no greater than i
 9:     for $j \leftarrow A.length$ to 1 do
10:         $B[C[A[j]] \leftarrow A[j]$
11:         $C[A[j]] \leftarrow C[A[j]] - 1$
12:     end for
13: end function
```

- **(a)** What is the running-time of counting-sort?
- **(b)** Is counting sort stable? Prove your claim.
- **(c)** Would the algorithm remain correct if we used a more standard for loop from 1 to *A.length* in lines 9-12? What effect on the result would there be in this case?
- **(d)** How is counting-sort affected in practice by an input of relatively few integers with a large range. i.e. k » A.length.
- **(e)** Is there a linear time sorting algorithm that addresses this problem?

Problem 1-2. (K Largest Elements)

- **(a)** Given an array of n integers, we want to find an algorithm to return the largest k elements. Consider algorithm 2.

  What is the time complexity of Naive-Top-K?
- **(b)** Can this problem be solved more efficiently?
- **(c)** What if the array is no longer composed of countable elements? How quickly can you return the top k elements?
- **(d)** Now again assume we have an input array of integers. How would you choose to find the largest k elements?

Algorithm 2 Repeatedly find the next largest element

```
function NAIVE-TOP-K($A, k$)
    $topElems \leftarrow []$
    for $i \leftarrow 1$ to $k$ do
        $max \leftarrow Select(A, A.length - i)$
        $topELems \leftarrow topElems.append(max)$
    end for
    return $topElems$
end function
```

Problem 1-3. (Heaps)

(a) Fill in the time complexities of the following operations on heaps:

- Heapify:
- HeapSort:
- HeapMaximum:
- HeapExtractMax:

(b) What height are the leaves of a heap?

(c) What is the maximum number of nodes at height h?

(d) Recall the BuildHeap algorithm. What is the time complexity of BuildHeap()? Justify your answer.

Algorithm 3 Given an unsorted array A, make A a heap

```
function BUILDHEAP($A$)
    $heapSize(A) \leftarrow A.length$
    for $j \leftarrow \lfloor A.length/2 \rfloor$ to 1 do
        Heapify(A,j)
    end for
end function
```

Problem 1-4. (Hashing Probability)

(a) What is the probability after placing the first 3 items into a hash map of k buckets there are no collisions? Assume uniform hashing.

(b) What is the probability after placing n items into a hash map of k buckets there is a collision? Assume uniform hashing.

(c) Guess and then calculate the probability from part (b) for $n = 50$ and $k = 500$? How does the actually probability compare to your guess?