6.006 Introduction to Algorithms
Spring 2008

Introduction to Algorithms: 6.006
Massachusetts Institute of Technology
Professors Srini Devadas and Erik Demaine

March 7, 2008
Handout 6

Quiz 1 Practice Problems

1 Asymptotic Notation

Decide whether these statements are **True** or **False**. You must briefly justify all your answers to receive full credit.

1. If $f(n) = \Theta(g(n))$ and $g(n) = \Theta(h(n))$, then $h(n) = \Theta(f(n))$

2. If $f(n) = O(g(n))$ and $g(n) = O(h(n))$, then $h(n) = \Omega(f(n))$

3. If $f(n) = O(g(n))$ and $g(n) = O(f(n))$ then $f(n) = g(n)$

4. $\dfrac{n}{100} = \Omega(n)$

5. $f(n) = \Theta(n^2)$, where $f(n)$ is defined to be the running time of the program `A(n)`:

```
def A(n):
  atuple = tuple(range(0, n)) # a tuple is an immutable version of a
                              # list, so we can hash it
  S = set()
  for i in range(0, n):
    for j in range(i+1, n):
      S.add(atuple[i:j])   # add tuple (i,...,j-1) to set S
```

2 Linked List Equivalence

Let S and T be two sets of numbers, represented as unordered linked lists of distinct numbers. All you have are pointers to the heads of the lists, but **you do not know the list lengths**. Describe an $O(\min\{|S|, |T|\})$-expected-time algorithm to determine whether $S = T$. You may assume that any operation on one or two numbers can be performed in constant time.

3 Hash Table Analysis

You are given a hash table with n keys and m slots, with the simple uniform hashing assumption (each key is equally likely to be hashed into each slot). Collisions are resolved by chaining.
What is the probability that the first slot ends up empty?

4 Heaps

1. The sequence $\langle 20, 15, 18, 7, 9, 5, 12, 3, 6, 2\rangle$ is a max-heap.
   True **False**

   Explain:

2. Where in a max-heap can the smallest element reside, assuming all elements are distinct? Include both the location in the array and the location in the implicit tree structure.

3. Suppose that instead of using `Build-Heap` to build a max-heap in place, the `Insert` operation is used n times. Starting with an empty heap, for each element, use `Insert` to insert it into the heap. After each insertion, the heap still has the max-heap property, so after n `Insert` operations, it is a max-heap on the n elements.

   (i) Argue that this heap construction runs in $O(n \log n)$ time.

   (ii) Argue that in the worst case, this heap construction runs in $\Omega(n \log n)$ time.

5 Python and Asymptotics

1. Give an example of a built-in Python operation that does not take constant time in the size of its input(s). What time does it take?

2. Since dictionary lookup takes constant expected time, one can output all n keys in a dictionary in sorted order in expected time $O(n)$.
   True **False**

   Explain:

3. Write a recurrence for the running time $T(n)$ of $f(n)$, and solve that recurrence. Assume that addition can be done in constant time.

```
def f(n):
  if n == 1:
    return 1
  else:
    return f(n-1)+f(n-1)
```

4. Write a recurrence for the running time of $g(n)$, and solve that recurrence. Assume that addition can be done in constant time.

```
def g(n):
  if n == 1:
    return 1
  else:
    x = g(n-1)
    return x+x
```

5. Now assume addition takes time $\Theta(b)$ where b is the number of bits in the larger number. Write a new recurrence for the running time of $g(n)$, and solve that recurrence. Express your final answer in Θ-notation.

6 Hashing

1. Given a hash table with more slots than keys, and collision resolution by chaining, the worst case running time of a lookup is constant time.
   True **False**

   Explain:

2. Linear probing satisfies the assumption of uniform hashing.
   True **False**

   Explain:

3. Assume that $m = 2^r$ for some integer $r > 1$. We map a key k into one of the m slots using the hash function $h(k) = k \bmod m$. Give one reason why this might be a poorly chosen hash function.