University of Washington
Department of Computer Science and Engineering
CSE 373, Autumn 2014

Assignment 2, Due Friday, October 17, in class at 2:30 PM

For each of the problems below, show your work, unless otherwise specified. For the inductive proofs, please format into the four parts shown in class, or points may be deducted.

Problem 1:

For each of the three groups of functions below, arrange the functions from slowest growth rate to fastest growth rate. For this problem, you do not need to show work for full credit. If any of the functions grow at the same rate, be sure to indicate this.

a)
(i) $1000n^2$
(ii) n^3
(iii) $n(\log n)^{1000}$
(iv) $\frac{(n+3)!}{n!}$

b)
(i) $n \log(n^2)$
(ii) $\lceil \log n \rceil$
(iii) $\frac{2}{n}$

c)
(i) $2^{\sqrt{n}}$
(ii) $2^{n \log n}$
(iii) $2^{(\log n)^{.9}}$

Hint: Try taking the log of each function to help compare.

Problem 2:

Prove by induction that:

$\sum_{i=1}^{n} i(i+1) = \frac{n(n+1)(n+2)}{3}$, where $n \geq 1$.

Problem 3:

An **almost balanced binary tree** (ABBT) is defined recursively by the following two rules:

1. Basis: A single node is an ABBT.
2. Recursive Step: Let T_1 and T_2 be ABBT trees, where either $\textbf{height}(T_1) = \textbf{height}(T_2)$, or $\textbf{height}(T_1)$ and $\textbf{height}(T_2)$ differ by 1. Then the tree constructed by an edge from the root node to T_1 and an edge from the root node to T_2 is also an ABBT.

The **size** and **height** of an ABBT are defined as follows:

- The **size** of an ABBT consisting of a single (root) node is 1. The **height** of an ABBT consisting of a single (root) node is 0.
- The size of a tree constructed by connecting two ABBT trees, T_1 and T_2, as defined in the Recursive Step of the definition of an ABBT, is $1 + \textbf{size}(T_1) + \textbf{size}(T_2)$. The **height** of the tree constructed is $1 + \max\{\textbf{height}(T_1), \textbf{height}(T_2)\}$.

Prove by induction that every almost balanced binary tree T satisfies:

$$\textbf{size}(T) \geq f_{\textbf{height}(T)+1}$$

where f_m denotes the m-th Fibonacci number. (As usual, $f_0 = 0$, $f_1 = 1$, and $f_{m+1} = f_m + f_{m-1}$ for $m \geq 1$).

Hint: Think about the cases for height by using the definition of an ABBT.

Problem 4:

Knowledge of combinations and permutations is useful for a variety of algorithms (and they are often found in programming challenges and interview questions). Recall that a **combination** is an unordered collection of items from a set, such as $\{43, 2, 34\}$ from the set of integers between 0 and 100. A **permutation** of an n-element set is an ordering of the elements in that set. For example, "FACE" represents a permutation of the letters in the word "CAFE".

The general formulas for counting permutations and combinations are as follows:

- The number of distinct permutations of an n-element set is given by $n!$.
- A sequence of r elements taken with replacement from an n-element set is analogous to an r-element string of characters from an n-element alphabet. There are n^r distinct cases.
- A partial permutation with r elements from an n-element set is a permutation of an r-element subset. This is sometimes called a sequence without repetition. The formula is $\frac{n!}{(n-r)!}$.
- Combinations with repetition: $\binom{n+r-1}{r} = \frac{(n+r-1)!}{r!(n-1)!}$
- Combinations without repetition: $\binom{n}{r} = \frac{n!}{r!(n-r)!}$. Unless specified otherwise, a combination is of this kind. The expression $\binom{n}{r}$ is often called a binomial coefficient and it is often read as "n choose r."

a) How many permutations of the letters in the string "ALGORITHMS" contain the substring "MATH"? (Note: These strings must be the same length as "ALGORITHMS", since we are just mixing up the characters, not looking at all possible substrings. Also, the letters "MATH" should be contiguous, meaning there should not be any other letters in between M, A, T, and H in the permutation.)

b) You have a sequence of 12 integers, where exactly two integers in the sequence are the same (i.e. there is exactly one repeat). Determine the number of unique subsequences containing exactly 4 integers, where order does not matter (i.e., the sequence 1, 2, 3, 4 is equivalent to 2, 1, 3, 4 and any other permutation of those four integers).

c) Consider the linear equation:

$$x + y + z = 15$$

How many non-negative integer solutions are there to this equation? (Where x, y, or z can equal the same number.)

Hint: See the Wikipedia article on numbers of combinations with repetition:

`http://en.wikipedia.org/wiki/Combination#Number_of_combinations_with_repetition`

Problem 5:

Recall that a geometric series can be represented by the general form:

$$\sum_{i=0}^{n-1} ar^i = a + ar + ar^2 + ar^3 + ...ar^{n-1}$$

where a is a real constant and r is the common ratio.

The sum of a finite geometric series is given by:

$$\sum_{i=0}^{n-1} ar^i = a(\frac{1-r^n}{1-r})$$

If the series is infinite and $|r| < 1$, then the geometric series converges to $\frac{a}{1-r}$.
If the series is infinite and $|r| \geq 1$, then the geometric series diverges.

1. Determine the sum of the following geometric series. If it is infinite and converges, specify the value it converges to. If it converges for only some values of the parameter k, specify the domain for which it converges. (Simplify as much as possible, and round decimals to two places or leave in fractional form.) You may use a graphing calculator for this problem.

   a) $\sum_{i=0}^{\infty} 7(-\frac{4}{45})^i$

   b) $\sum_{i=0}^{\infty} (\frac{100k^2}{2^{k-5}})^i$

   c) $\sum_{i=0}^{9} \frac{1}{20}(\frac{9}{4})^i$

2. Write $0.27272727\overline{27}$ as a fraction.

Problem 6:

Let the running time for the following four snippets of code be given by $f_1(n)$, $f_2(n)$, $f_3(n)$, and $f_4(n)$.

For each one, either find a corresponding function $g_1(n)$, $g_2(n)$, $g_3(n)$, or $g_4(n)$, such that $f(n)$ is $\Theta(g(n))$ and g is a "simple" function, or show that no such function exists. "Simple" means it should have no unneeded constants or low-order terms. (E.g., $n^5 \log n$ and x^{17} are simple, but $3n^3$ and $x^2 + x$ are not.)

1.
```
sum = 0
for i in range(0, n):
    for j in range(0, i):
        sum += 1
```

2.
```
sum = 0
for i in range(0, n):
    for j in range(0, i * n):
        sum += 1
```

3.
```
sum = 0
for i in range(0, n):
    for j in range(0, i):
        sum += 1
    for k in range(0, 20000):
        sum += 1
```

4.
```
sum = 0
for i in range(0, n):
    for j in range(0, i**2):
        if j % 2 == 0:
            for k in range(0, j):
                sum += 1
```