Prof. Mo Chen | **review session** | CMPT 125

Simon Fraser University | | Spring 2020

Problem 1

Take the following list of functions and arrange them in ascending order of growth rate. That is, if function $g(n)$ immediately follows function $f(n)$ in your list, then it should be the case that $f(n)$ is $O(g(n))$.

$$g_1(n) = 2^{\sqrt{\log n}}$$

$$g_2(n) = 2^n$$

$$g_3(n) = n(logn)^3$$

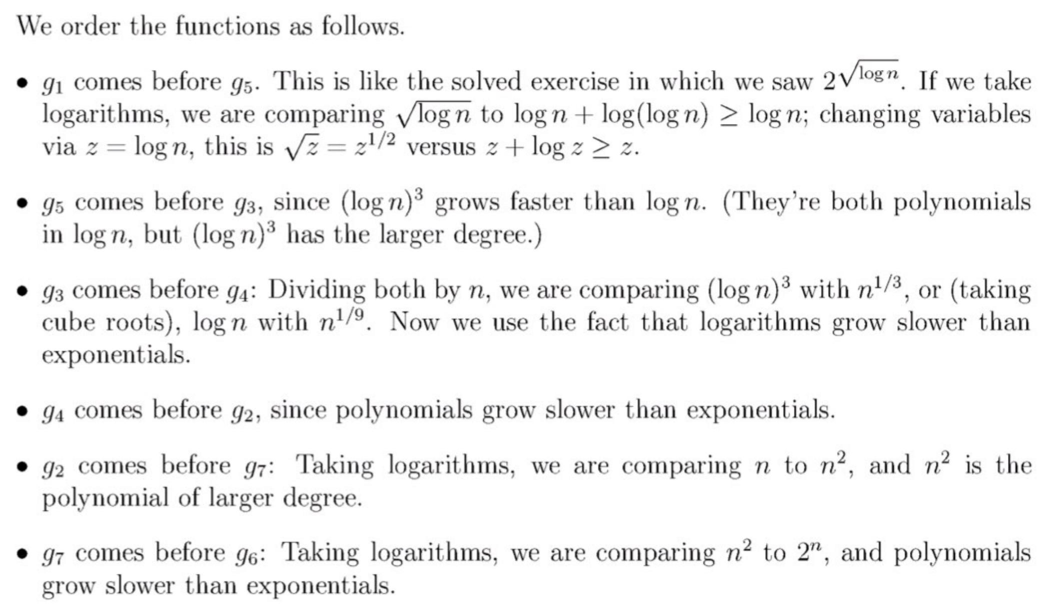
$$g_4(n) = n^{\frac{4}{3}}$$

$$g_5(n) = n^{(logn)}$$

$$g_6(n) = 2^{2^n}$$

$$g_7(n) = n^{n^2}$$

solution:



We order the functions as follows.

- g_1 comes before g_5. This is like the solved exercise in which we saw $2^{\sqrt{\log n}}$. If we take logarithms, we are comparing $\sqrt{\log n}$ to $\log n + \log(\log n) \geq \log n$; changing variables via $z = \log n$, this is $\sqrt{z} = z^{1/2}$ versus $z + \log z \geq z$.
- g_5 comes before g_3, since $(\log n)^3$ grows faster than $\log n$. (They're both polynomials in $\log n$, but $(\log n)^3$ has the larger degree.)
- g_3 comes before g_4: Dividing both by n, we are comparing $(\log n)^3$ with $n^{1/3}$, or (taking cube roots), $\log n$ with $n^{1/9}$. Now we use the fact that logarithms grow slower than exponentials.
- g_4 comes before g_2, since polynomials grow slower than exponentials.
- g_2 comes before g_7: Taking logarithms, we are comparing n to n^2, and n^2 is the polynomial of larger degree.
- g_7 comes before g_6: Taking logarithms, we are comparing n^2 to 2^n, and polynomials grow slower than exponentials.

Problem 2

How would you test whether x is the ascii code of an upper-case letter, without using any library functions and without mentioning any numbers, and only one line of code?

solution:

```
if('A' <= x && x <= 'Z')
```

Problem 3

How many bytes are required to store the string, "without doubt"?

solution:

14, one for each character plus one for the null terminator.

Problem 4

If you have to compute with integers that will go up to a few million, what data type should you use?

solution:

int

Problem 5

If you have to compute with money values representing the Federal budget (with numbers into the trillions of dollars, but accuracy to the penny is also needed), what data type should you use?

solution:

long long. Not doubles because we need perfect accuracy, so we have to represent amounts in pennies.

Problem 6

The following program finds the common elements in two different integer arrays (`fibArray` and `primeArray`) and stores them in another array called `commonArray`. At the end of the program, it prints out how many common elements there are. There are five bugs in the code. Identify them and then fix them.

```
1  int main()
2  {
3          int fibArray[] = {1,2,3,5,8,13,21,34,55,89};
4          int primeArray[] = {2,3,5,7,11,13,17,19,23,29};
5          int commonArray[];
6          int i, j;
7
8          for(i=0;i<10;++i) 9
9          {
10                 for ( j = 0; j < 10; ++j )
11                 {
12                         if (fibArray[i] = primeArray[j])
13                         {
14                                 commonArray[j] = primeArray[j];
15                                 ++n;
16                         }
17                 }
18         }
19
20         printf("The total number of common elements is %n\n", n);
21         return 0;
22 }
```

solution:

Line 5: `commonArray` has no size. **Fix:** int `commonArray[10]`;
Line 6: n is not declared. **Fix:** add declaration int n;
Line 12: assignment operator used instead of equality operator. **Fix:** change = to ==.
Line 14: logic bug. Index to `commonArray` should be n, not j. **Fix:** change j to n.
Line 20: type field error. **Fix:** change %*n* to %*d*.

Problem 7

write a code for iterating over the list.

solution:

```
Void print_list(node_t * head) {
        node_t * current = head;

        while(current != Null) {
                printf(   %d\ n   , current->val);
                current = current->next;
        }
}
```

Problem 8

write a code for removing the last item of the list.

solution:

```
Int remove_last(node_t * head) {
        int retail = 0;
        // if there is only one item
        if (head->next == NULL) {
                retval = head->val;
                free(head);
                return retval;
        }

        //get to the second to the last node
        node_t * current = head;
        while(current->next->next !=NULL) {
                current = current->next;
        }

        retval = current->next->val;
        free(current->next);
        current->next = NULL;
        return retval;

}
```

Problem 9

What is an overflow? Give an example of an overflow assuming that numbers are represented in 3 bits.

solution:

Overflow is an error in calculation due to the limited number of bits that represents a number in a computer. An example using 3 bit representation is $2 + 3 = -3$ since $010 + 011 = 101$.

Problem 10

Suppose x is an array of integers, and we have just executed this code:

```
for(i=0;i<10;i++)
        x[i] = i+1;
```

Suppose that `x[0]` is stored at address 4500. What is the value of each of the following expressions?

a) x

b) x[0]

c) *x

d) x[1]

solution:

a) 4500

b) 4500, same as x.

c) 1, the value that was placed in x[0].

d) 2