INT202 Complexity of Algorithms

Tutorial 2

Q1. Compute the following expressions that are given in postfix notation.

(a) 1 2 + 4 * 3 + $\Rightarrow (1+2)\times 4+3=15$

(b) 5 7 * 3 4 1 + * - $\Rightarrow 5\times 7-3\times(4+1)=35-15=20$

(c) 3 4 2 * 1 5 - 2 ^ / + $\Rightarrow 3+(4\times 2)\div(1-5)^2=3+8\div 16=3.5$

Note: The ^ symbol means exponentiation, i.e. x y ^ means x^y in the usual infix notation.

Q2. Compute Big-Oh notation.

a) $\Rightarrow O(\sqrt[3]{n})$

```
void func(int n){
  int i = 0;
  while(i * i * i <= n)
    i++;
}
```

1

assume $x^3=n$, so $x=\sqrt[3]{n}$

b)

```
void recursive(int n, int m, int o) {
  if (n <= 0) {
    printf("%d, %d\n", m, o);
  } else {
    recursive(n - 1, m + 1, o);
    recursive(n - 1, m, o + 1);
  }
}
```

$R(n)=R(n-1)+R(n-1)$

$=2R(n-1)=4R(n-2)=2^nR(0)$

$\Rightarrow O(2^n)$

c)

```
void func(int n) {
  for (int i = 1, s = 0; i <= n; ++i) {
        int t = 1;
        for (int j = 1; j <= i; ++j)
              t = t * j;
  }
}
```

n

$1, 2 \cdots n,\ \frac{n^2+n}{2}$

$\Rightarrow O(n^2)$

注意这里不是$O(n^3)$,因为在第二个loop的时候,已经计算了第一个loop的n.

Q3: a), assume that $f(n) = C_1 h(n)$, $g(n) = C_2 h(n)$, so, $f(n)+g(n) = h(n)(C_1+C_2)$, so True.

b), $\sum_{i=1}^{n} \log i = \log 1 + \cdots + \log n = \log(1 \times \cdots \times n) \le \log(n \times n \cdots \times n) = \log(n^n) = n\log n$, so, False,

注意，b的答案不是n，这里是算术式，而不是程序，需要找到结果中的最大量级，而不是运算次数。

Q3. State whether the following statement is true or false, and give a brief justification for your answer.

a) If $f(n)$ and $g(n)$ are both $O(h(n))$, then $f(n)+g(n)$ is $O(h(n))$.

b) The asymptotic complexity of $\sum_{i=1}^{n} \log i$ is $O(\log n)$

Q4: 1), $\sum_{i=1}^{n} \sum_{j=i}^{2i} 1$

2), $\sum_{i=1}^{n} \sum_{j=i}^{2i} 1 = \sum_{i=1}^{n} (2i-i+1) = \sum_{i=1}^{n} (i+1)$

for general $\sum_{i=1}^{n} i = \frac{(n+1)n}{2}$, so $\sum_{i=1}^{n} \sum_{j=1}^{2i} 1 = \frac{(n+2)(n+1)}{2}$

So, $O(n^2)$

Q4. Consider the following code fragment.

```
for i=1 to n do
    for j=i to 2*i do
        output hello
```

Let T(n) denote the number of times 'hello' is printed as a function of n.

1) Express T(n) as a summation;

2) Simplify the summation and give the worst-case running time using Big-Oh notation.

注：当有如 $\sum_{i=1}^{n} 1$ 这种的式子，结果是式子的运算次数，如 $\sum_{i=1}^{n} 1 = n$，但 $\sum_{i=1}^{n} i$ 的结果为 $\sum_{i=1}^{n} i = 1+\cdots+n$。

Q5. Consider the following code fragment.

```
for i=1 to n-1 do
    for j=i+1 to n do
        for k=1 to j do
            output hello
```

Let T(n) denote the number of times 'hello' is printed as a function of n.

1) Express T(n) as a summation;

2) Simplify the summation, and give the worst-case running time using Big-Oh notation.

Q5:

1), $\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{k=1}^{j} 1$

2), $\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{k=1}^{j} 1 = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} j = \sum_{i=1}^{n-1} \left(\sum_{j=1}^{n} j - \sum_{j=1}^{i} j\right)$

$= \sum_{i=1}^{n-1} \left(\frac{(n+1)\cdot n}{2} - \frac{(i+1)\cdot i}{2}\right)$

$= \sum_{i=1}^{n-1} \left(\frac{n^2+n-i^2-i}{2}\right) = (n-1)\cdot\left(\frac{n^2+n}{2}\right) - \sum_{i=1}^{n-1} \frac{i^2}{2} - \sum_{i=1}^{n-1} \frac{i}{2}$

$= \frac{1}{2}\left\{(n-1)\cdot n^2 + (n-1)\cdot n - \frac{(n-1)\cdot n\cdot[2(n-1)+1]}{6} - \frac{n\cdot(n-1)}{2}\right\} = \cdots \Rightarrow O(n^3)$

Q6. Give a tight bound of the runtime complexity class for the following code fragment in Big-Oh notation, in terms of the variable N.

注：$\sum_{k=1}^{n} k^2 = 1^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6}$

```
int sum = N;
    for (int i = 0; i < 1000; i++) {   1000
    for (int j = 1; j <= i; j++) {   1000
    sum += N;}
    for (int j = 1; j <=i; j++) {   1000
    sum += N;}
    for (int j = 1; j <= i; j++) {   1000
    sum += N;}
}
```

Q6: $O(1)$

Q7. Prove that $3\log n + \log\log n$ is $\Theta(\log n)$

Q7: If $3\log n + \log\log n$ is $O(\log n)$, it

$C_1 \log n \le 3\log n + \log\log n \le C_2 \log n$

$3\log n \le 3\log n + \log\log n$, so $C_1 = 3$,

since $\log\log n \le \log n$, so, $3\log n + \log\log n \le 4\log n$,

so, $C_2 = 4$

thus ..