Problem 2.1

1. Show that $5n^2 + 2n + 1$ is $O(n^2)$

$$5n^2 + 2n + 1 \le 5n^2 + 2n + n \quad ,\forall n \ge 1$$

$$= 5n^2 + 3n \le 5n^2 + n^2 \quad ,\forall n \ge 3$$

$$= 6n^2 = O(n^2)$$

2. What is the Big oh of $n^2 + n\log(n)$? prove your answer.

$$n^2 + n \log n = O(n^2)$$

3. Show that $2n^3 \in / O(n^2)$.

$$2n^3 = O(n^3)$$

$$f(n) = O(n^2) \le cn^2 \ne 3n^3$$

4. Assume that the expression below gives the processing time $f(n)$ spent by an algorithm for solving a problem of size n.

$$10n + 0.1n^2$$

(a) Select the dominant term(s) having the steepest increase in n.

$$0.1n^2$$

(b) Specify the lowest Big-Oh complexity of the algorithm.

$$O(n^2)$$

5. Determine whether each statement is *true* or *false* and correct the expression in the latter case:

(a) $100n^3 + 8n^2 + 5n$ is $O(n^4)$. (true)
(b) $100n^3 + 8n^2 + 5n$ is $O(n^2 \log n)$. (false)

Problem 2.2

1. Find the simplest $g(n)$, c and n_0 for the following $f(n)$ s.t: $f(n) \le cg(n), \forall n \ge n_0$.

(a) $6n^2 + n - 4$.

$$6n^2 + n - 4 \le 6n^2 + n \le 6n^2 + n^2 \quad ,\forall n \ge 1$$

$$= 7n^2 = O(n^2)$$

$$g(n) = n^2, \qquad c = 7, \ \ n_0 = 1$$

(b) $4\log(n) + 2$.

$$4 \log n + 2 = 4 \log n + 2 \log 10 \le 4 \log n + 2 \log n \quad ,\forall n \ge 10$$

$$= 6 \log n$$

(c) $3n^3 - 20n^2 + 10\log(n)$.

$$3n^3 - 20n^2 + 10\log n \le 3n^3 + 10\log n \le 3n^3 + 10n^3 \quad ,\forall n \ge 1$$
$$= 13n^3$$

2. Find the big Oh notation for the following functions:

(a) $n + \log n^{n^3} + n^2\log(n)$.

$$O(n^3 \log n)$$

(b) $2^{\log(n!)+2} + 3^n$.

$$O\left(2^{\log n!+2}\right)$$

Problem 2.3

1. Order the following functions by asymptotic growth rate: $4n\log n + 2n$, 2^{10}, $2^{\log n}$, $3n + 100\log n$, $4n$, 2^n, $n^2 + 10n$, n^3, $n\log n$. (Question R-4.8 page 182 of the textbook)

$$2^{10}, 2^{\log n}, 4n, 3n + 100\log n, n\log n, 4n\log n + 2n, n^2 + 10n, n^3, 2^n$$

2. Show that $\log n^{2n} + n^2$ is $O(n^2)$

$$\log n^{2n} + n^2 = 2n\log n + n^2 \le n^2 + 2n^2 = 3n^2 \quad ,\forall n \ge 1$$
$$= O(n^2)$$

3. Show that $\sum_{i=1}^{5} i^3$ is $O(1)$

$$\sum_{i=1}^{5} i^3 = 1^3 + 2^3 + 3^3 + 4^3 + 5^3 = O(1)$$

4. Show that $\sum_{i=1}^{n} \lceil \log i \rceil$ is a $O(n\log n)$

$$\sum_{i=1}^{n} \lceil \log i \rceil = n\log n = O(n\log n)$$

5. Using the definition of the Big-Oh, prove that $f(n) = 10n + 5\log n$ is a big-oh of $g(n) = n$

$$f(n) = 10n + 5\log n \le 10n + 5n \quad ,\forall n \ge 1$$
$$= 15n = O(n)$$

Problem 2.5

Write the frequency for each line of the following code excerpts as a sum.

1. **for** (i = 1; i < n - 1; i ++)

Sol.: $\left(\sum_{i=1}^{n-2} 1\right) + 1$. The $+1$ is for the last check.

2. **for** (i = n; i >= 0 ; i --)

Sol.: $\left(\sum_{i=1}^{n-2} 1\right) + 1$

3. **for** (i = 0; i < n ; i += 2)

Sol.: $\frac{n}{2} + 1$

4. **for** (i = 0; i < n ; i += 3)

Sol.: $\frac{n}{3} + 1$

5. **for** (i = 0; i < n ; i++)

for (j = 2; j < i; j++)

Sol. for line 2: $\left(\sum_{i=0}^{n-1} \sum_{j=2}^{i-1} 1\right) + 1$. The +1 is for the last check.

6. **for** (i = 0; i < n ; i++)
for (j = i; j > 0; j --)

Sol. for line 2: $\left(\sum_{i=0}^{n-1} \sum_{j=1}^{i} 1\right) + 1$

7. **for** (i = 1; i <= n ; i *= 2)

Sol.: $\log_2 n$

8. **for** (i = 1; i <= n ; i *= 3)

Sol.: $\log_3 n$

Problem 2.6

Analyze the following code excerpts:

1. **int** sum = 0;	1
for (**int** i = n; i > 0; i = i - 2)	$\frac{n}{2} + 1$
sum = sum + i;	$\frac{n}{2}$

$O(n)$

2. **int** sum = 0;	1
for (**int** i = 1; i < n; i = 2 * i)	$\log_2 n + 1$
sum = sum + i;	$\log_2 n$

$O(\log_2 n)$

3. **int** sum = 0;	1
for (**int** i = 1; i <= n ; i++)	$n + 1$
for (**int** j = 0; j < 2 * i ; j++)	$\sum_{i=1}^{n} 2i + 1$
sum += j;	$\sum_{i=1}^{n} 2i$
return sum ;	1

$O(n^2)$

4. **for** (**int** i = 0; i < n * n * n; i++) {	$n^3 + 1$
System .out. println (i);	n^3
for (**int** j = 2; j < n; j++)	$n^3(n - 2 + 1)$
System .out. println (j); }	$n^3(n - 2)$
System .out. println ("End!");	n^3

$O(n^4)$

5. **int** k = 100 , sum = 0;	1
for (**int** i = 0; i < n; i++)	$n + 1$
for (j = 1; j <= k; j++) {	$n(k + 1)$
sum = i + j;	nk
System .out. println (sum);	nk
}	

$O(nk)$

6. **int** sum = 0;	1
for (**int** i = 0; i < n * n; i ++) {	$n^2 + 1$
for (**int** j = n - 1; j >= n - 1 - i; j --) {	$\sum_{i=0}^{n^2-1} i + 1$
sum = i + j;	$\sum_{i=0}^{n^2-1} i$
System .out. println (sum);	$\sum_{i=0}^{n^2-1} i$
}	
}	

$O(n^4)$

7. **int** sum = 0;	1
for (**int** i = 1; i <= 2^n; i = i * 2) {	$\log_2 2\hat{}n + 1$
for (**int** j = 0; j <= log(i); j ++) {	$\sum_{i=1}^{2\hat{}n} \log i + 1$
sum = i + j;	$\sum_{i=1}^{2\hat{}n} \log i$
System .out. println (sum);	$\sum_{i=1}^{2\hat{}n} \log i$
}	
}	

$o(2\hat{}n \log 2\hat{}n)$

8. **int** sum = 0; **int** k = 2^3 ;	1
for (**int** i = k; i <= 2^(n - k); i = i * 2) {	$\log\left(2^{(n-k)} - k + 1\right) + 1$
for (**int** j = 2^(i - k); j< 2^(i + k); j = j * 2) {	$\sum_{i=1}^{2\hat{}n} \log\left(2^{(i+k)} - 2^{(i-k)}\right) + 1$
sum = i + j;	$\sum_{i=1}^{2\hat{}n} \log\left(2^{(i+k)} - 2^{(i-k)}\right)$
System .out. println (sum);	$\sum_{i=1}^{2\hat{}n} \log\left(2^{(i+k)} - 2^{(i-k)}\right)$
}	
}	

$O(2^{2n})$

9. **int** sum = 0;	1
for (**int** i = 2^n; i >= 1; i = i / 2) {	$\log_2 2^n + 1$
for (**int** j = i; j >= 1; j = j / 2) {	$\sum_{i=1}^{2\hat{}n} \log(i) + 1$
sum = i + j;	$\sum_{i=1}^{2\hat{}n} \log(i)$
System .out. println (sum);	$\sum_{i=1}^{2\hat{}n} \log(i)$
}	
}	

$o(2^n \log 2^n)$

10. **int** sum = 0;	1
for (**int** i = n; i > 0; i --) {	$n + 1$
for (**int** j = i; j <= n; j++) {	$\sum_{i=1}^{n} (n - i) + 1$
sum = i + j;	$\sum_{i=1}^{n} (n - i)$
System .out. println (sum);	$\sum_{i=1}^{n} (n - i)$
}	
}	

$o(n^2)$

11. **int** sum = 0;	1
for (**int** i = 0; i < n; i++) {	$n + 1$
for (**int** j = 0; j < i; j++) {	$\sum_{i=1}^{n} i + 1$
for (**int** k = n; k > 0; k --)	$(n + 1) \sum_{i=1}^{n} i$
sum = i + j + k;	$n \sum_{i=1}^{n} i$
}	
}	

$O(n^3)$

12. **int** k = 1;	1
for (**int** i = 1; k <= n; i *= ++k) {	$n + 1$
for (**int** j = 0; j < n; j++)	$n(n + 1)$
sum = i + j;	n^2
}	

$O(n^2)$

13. **int** sum = 0;	1
for (**int** i = 0; i < n; i++) {	n
for (**int** j = i + 1; j < n; j ++)	$\sum_{i=0}^{n-1}(n-i)+1$
sum = sum + A[j];	$\sum_{i=0}^{n-1}(n-i)$
A[i] = A[i] + sum;	$\sum_{i=0}^{n-1}(n-i)$
}	

$O(n^2)$

14. **int** k = 3, j = 5, sum = 0;	1
for (**int** i = 0; i < n; i++)	$n+1$
for (j = 1; j <= k; j++) {	$n(k+2)=5n$
sum = i + j;	$n(k+1)=4n$
System .out. println (sum);	$n(k+1)=4n$
}	

$O(n)$

15. **for** (**int** i = 0; i < n * n * n; i++) {	n^3+1
System .out. println (i);	n^3
for (**int** j = 2; j < n; j++) {	$n^3(n)$
System .out. println (j);	$n^3(n-1)$
}	
}	
System .out. println (" Goodbye !");	1

$O(n^4)$

16. **for** (**int** i = 0; i < n * n; i ++) {	n^2+1
System .out. println (i);	n^2
for (**int** j = 4; j <= n; j++) {	$n^2(n-1)$
System .out. println (j);	$n^2(n-2)$
}	
}	
System .out. println (" Goodbye !");	1

$O(n^2)$

17. m = 1;	1
while (m < 100) {	101
system .out. println (m);	100
i = 0;	100
while (i < n) {	$100(n+1)$
system .out. println (n * m);	$100n$
i++;	$100n$
}	
m++;	100
}	

$O(n)$

18. **for** (**int** i = 0; i < 2 * n; i = i + 2) {	$(n+1)$
for (**int** j = 0; j < n; j++)	$n(n+1)$
if (j % 2 == 0)	n^2
system .out. println (j);	n^2
}	

$O(n^2)$

19. **for** (**int** i = 0; i < n * log (n); i++) {	$n\log n+1$
System .out. println (i);	$n\log n$
for (**int** j = 2; j < n; j++) {	$(n)n\log n$
System .out. println (j);	$(n-1)n\log n$
}	
}	

$O(n^2\log n)$

20. **for** (**int** i = 0; i < n * n; i ++) {	n^2+1
System .out. println (i);	n^2

for (**int** j = 2 * n; j > n; j --) {	$n^2(n+2)$
System .out. println (j);	$n^2(n+1)$
}	
}	

$O(n^3)$

21. **int** m = 1;	1
while (m <= n) {	$\log_2 n + 1$
system .out. println (m);	$\log_2 n$
i = n;	$\log_2 n$
while (i > 0) {	$\log_2 n\,(\log_2 n + 1)$
system .out. println (i);	$\log_2^2 n$
i = i / 2;	$\log_2^2 n$
}	
m++;	$\log_2 n$
}	

$O(\log_2^2 n)$

22. **for** (**int** i = 0; i < 2 * n; i = i + 2) {	$n+1$
for (**int** j = 0; j < i; j++)	$\sum_{i=0}^{2n-1} i + 1$
if (j % 2 == 0)	$\sum_{i=0}^{2n-1} i$
system .out. println (j);	$\sum_{i=0}^{2n-1} i$
}	

$O(n^2)$

Problem 2.7

1. Given an *n*-element array *X*, Algorithm *B* chooses log*n* elements in *X* at random and executes an *O*(*n*)-time calculation for each. What is the worst-case running time of Algorithm *B*? (Question R-4.30 page 184 of the textbook)

$$O(n \log n)$$

2. Given an n-element array *X* of integers, Algorithm *C* executes an *O*(*n*)-time computation for each even number in *X*, and an *O*(log*n*)-time computation for each odd number in *X*. What are the best-case and worst-case running times of Algorithm *C*? (Question R-4.31 page 184 of the textbook)

Best case: All odd → $O(n \log n)$

Worst case: All Even → $O(n^3)$

Problem 2.9

For the following functions:

1. Give two example inputs leading to the best and worst running time respectively.
2. Analyze the performance of the function in each case (best and worst).

```
a) public int func1 ( int A[], int n) {
int maxr = 0;
int maxi = 0;
int i= 0;
while (i < n) {
int j= i+1;
int nbr = 1;
while ((j < n) && (A[i] == A[j])) {
nbr ++;
j++;
}
if (nbr > maxr ) {
maxr = nbr;
maxi = i;
```

```
}
i= j;
}
return maxi ;
}
```

Best: $[5,3,1,2,3,7]$

Worst: $[5,5,5,5,5,5,5,5]$

Best case: $O(n)$

Worst case: $O(n)$

```
b) public int func2 ( int A[], int n) {
int maxr = 0;
int maxi = 0;
int i= 0;
while (i < n) {
int j= i+1;
int nbr = 1;
while (j < n) {
if (A[i] == A[j])
nbr ++;
j++;
}
if (nbr > maxr ) {
maxr = nbr;
maxi = i;
}
i++;
}
return maxi ;
}
```

Best: $[5,3,1,2,3,7]$

Worst: $[5,5,5,5,5,5,5,5]$

Best case: $O(n^2)$

Worst case: $O(n^2)$

```
c) public void func3 ( int A[], int n) {
int i= 0;
int j= n -1;
while (i < j) {
while ((A[i] <= 0) && (i<j)) {
i++;
}
while ((A[j] > 0) && (i<j)) {
j --;
}
int tmp = A[i];
A[i]= A[j];
A[j]= tmp;
}
}
```

Best: $[5,3,1,2,3,7]$

Worst: $[-5,3,-1,2,-3,7]$

Best case: $O(n)$

Worst case: $O(n^2\ )$

```
d) public void func4 ( int A[], int B[], int C[], int n) {
int i= 0;
int j= 0;
int k= 0;
while ((i < n) && (j < n)){
if (A[i] <= B[j])
C[k ++]= A[i ++];
else
C[k ++]= B[j ++];
}
if (i == n) {
while (j < n)
C[k ++]= B[j ++];
}
else {
while (i < n)
C[k ++]= A[i ++];
}
}
```

Best: $A = [3,2,1,3,2]$ $B = [7,5,8,9,6]$

Worst: $A = [5,4,10,2,21]$ $B = [10,2,5,11,3]$

Best case: $O(n)$

Worst case: $O(n)$