Homework 1

1. **a.**

```
1: Input: Array A[1 : n]
2: for j = 2 to n do
3:     i ← j − 1;
4:     while (i > 0 and A[i] > A[i + 1]) do
5:         swap(A[i], A[i + 1]);
6:         i ← i − 1
7:     end while
8: end for
```

> Each time line 2 is executed it performs n operations: $n-1$ for the times of the loop; 1 for the last check.
>
> Each time line 3 is executed it performs 1 operation.
>
> Each time line 5 is executed in the loop it performs 3 operation.
>
> Each time line 6 is executed in the loop it performs 1 operation.
>
> **For the worst case** when the new number is always the smallest in the current array:
>
> 1. **Each time line 4 is executed it performs $2(1+\sum_{i=1}^{n-1} i)$ operations**: The condition will execute $i+1$ times with 2 comparisons. As we can see, the number of i operations in line 4 will execute in this form: $1, 2, 3, 4, \ldots, n-1$. Therefore, the condition in the while loop will be true for $\frac{1+n-1}{2} * (n-1) = n(n-1)/2$ times with executing $2*(n(n-1)/2+1) = n(n-1)+2$ operations.
> 2. When in the loop, as it enters $n(n-1)$ times, it performs $n(n-1)/2*(3+1) = 2n(n-1)$ operations.
> 3. The total operations will be $n+n-1+n(n-1)+2+2n(n-1) = 3n^2-n+1$.
>
> **For the best case** when the new number is always the largest in the current array:
>
> 1. **Each time line 4 is executed it performs $2(n-1)$ operations**: The condition will execute n times with 2 comparisons, **but always be false**.
> 2. Therefore, The total operations will be $n+n-1+2(n-1) = 4n-3$.
>
> **Hence, as** $T(n) <= 3n^2-n+1 <= 3n^2$,
>
> $T(n) = O(n^2)$.

b.

> As mentioned above, the number of operations in the best cases is $4n-3$.
>
> If the input is sorted in increasing order, i.e. sorted array, like $[1, 3, 5, 6, 7]$, the algorithm performs a minimum number of operations, whose $T(n) = O(n)$.

2. **a.** $ f(n)=n^2,g(n)=4^{\log n}$

> **Ans**: $f = \Theta(g)$

b. $f(n) = n, g(n) = 2^{3\log n}$.

> **Ans:**$f = O(g)$ but $f \neq \Theta(g)$

c. $f(n) = 10n^2, g(n) = 10^{10}n$.

> **Ans:** $g = O(f)$ but $g \neq \Theta(f)$

d. $f(n) = 3^n, g(n) = 3^{n\log n}$

> **Ans:** $f = O(g)$ but $f \neq \Theta(g)$

e. $f(n) = n^{2\log n}, g(n) = (\log n)^n$.

> **Ans:**$f = O(g)$

f. $f(n) = n^4 + 3n^2 + 77, g(n) = n^4/1000$.

> **Ans:**$f = \Theta(g)$

g. $f(n) = 5^n, g(n) = 4^n$.

> **Ans:** $g = O(f)$ but $g \neq \Theta(f)$

h. $f(n) = \log\log n, g(n) = \log n$.

> **Ans:**$f = O(g)$

i. $f(n) = n$ when n is odd, $f(n) = n\log n$ when n is even; $g(n) = n$ when n is even, $g(n) = n^2$ when n is odd.

> **Ans:** None of these.

j. $f(n) = n$ when n is odd, $f(n) = n^2$ when n is even; $g(n) = n^2 + n$.

> **Ans:**$f = O(g)$

3. **a.** Let $f(n) = 5n^2 + 3n$ and $g(n) = 3n^3$. Show that $f(n) = o(g(n))$.

> To show $f(n) = o(g(n))$, we need to prove $f(n) <= cg(n), c > 0$ for $n \geq n_c$, i.e. to prove $\frac{f(n)}{g(n)} <= c$.
>
> Proof:

> $$\begin{aligned}\frac{f(n)}{g(n)} &= \frac{5n^2+3n}{3n^3} = \frac{2}{3n} + \frac{1}{n^2} + 1 \\ \text{As } n \geq 1, &\frac{2}{3n} <= \frac{2}{3}, \frac{1}{n^2} \leq 1 \\ \frac{f(n)}{g(n)} = \frac{2}{3n} + \frac{1}{n^2} + 1 &<= \frac{2}{3} + 1 + 1 = \frac{3}{8}\end{aligned} \tag{1}$$
>
> So for $n \geq n_c = 1, f(n) <= cg(n)$.

b. Show that $2^n = o(3^n)$

> To show $f(n) = o(g(n))$, we need to prove $f(n) <= cg(n), c > 0$ for $n > n_c$, i.e. to prove $\frac{f(n)}{g(n)} <= c$.
>
> Proof:
>
> $$\frac{f(n)}{g(n)} = \frac{2^n}{3^n} = (\frac{2}{3})^n \leq 1^n = 1 \tag{2}$$
>
> So for $n \geq n_c = 1, f(n) <= cg(n)$.

4. **a. Suppose $k = 3$. What is the best order in which to perform the merges? Justify your answer.**

> Suppose k=3, lists would be L_1, L_2, L_3 of lengths $l_1 \leq l_2 \leq l_3$, respectively.
>
> **If we merge from the beginning:**
>
> - Let $L_{1,2}$ denotes the merged list of L_1, L_2. A new length will be $l_{1,2} = l_1 + l_2$. The number of operations would be $l_1 + l_2 + 1$.
> - Let $L_{1,2,3}$ denotes the merged list of L_1, L_2, L_3. A new length will be $l_{1,2,3} = l_{1,2} + l_3 = l_1 + l_2 + l_3$. The number of operations would be $l_{1,2} + l_3 + 1$.
> - The total number of operations would be $l_1 + l_2 + 1 + l_{1,2} + l_3 + 1 = l_1 + l_2 + 1 + l_1 + l_2 + l_3 + 1 = 2l_1 + 2l_2 + l_3 + 2$
>
> **If we merge from the end:**
>
> - Let $L_{2,3}$ denotes the merged list of L_2, L_3. A new length will be $l_{2,3} = l_2 + l_3$. The number of operations would be $l_2 + l_3 + 1$.
> - Let $L_{1,2,3}$ denotes the merged list of L_1, L_2, L_3. A new length will be $l_{1,2,3} = l_{2,3} + l_1 = l_1 + l_2 + l_3$. The number of operations would be $l_{2,3} + l_1 + 1$.
> - The total number of operations would be $l_2 + l_3 + 1 + l_{2,3} + l_1 + 1 = l_2 + l_3 + 1 + l_1 + l_2 + l_3 + 1 = 2l_2 + 2l_3 + l_1 + 2$
>
> **So as $l_1 \leq l_2 \leq l_3$, merging from the beginning has fewer operations compared to the end one. It is because each merging accumulates the former operations again, so merging the lists first that have a lower size will have fewer operations, i.e. merge in an order: 1->2->3**

b. Suppose $k = 3$. What is the best order in which to perform the merges? Justify your answer.

> Suppose k=4, lists would be L_1, L_2, L_3, L_4 of lengths $l_1 \leq l_2 \leq l_3 \leq l_4$, respectively.

If we merge from the beginning:

- Let $L_{1,2}$ denotes the merged list of L_1, L_2. A new length will be $l_{1,2} = l_1 + l_2$. The number of operations would be $l_1 + l_2 + 1$.
- Let $L_{1,2,3}$ denotes the merged list of L_1, L_2, L_3. A new length will be $l_{1,2,3} = l_{1,2} + l_3 = l_1 + l_2 + l_3$. The number of operations would be $l_{1,2} + l_3 + 1$.
- Let $L_{1,2,3,4}$ denotes the merged list of L_1, L_2, L_3, L_4. A new length will be $l_{1,2,3,4} = l_{1,2,3} + l_4 = l_1 + l_2 + l_3 + l_4$. The number of operations would be $l_{1,2,3} + l_4 + 1$.
- The total number of operations would be $l_1 + l_2 + 1 + l_{1,2} + l_3 + 1 + l_{1,2,3} + l_4 + 1 = 3l_1 + 3l_2 + 2l_3 + l_4 + 3$

If we merge from the end:

- Let $L_{3,4}$ denotes the merged list of L_3, L_4. A new length will be $l_{3,4} = l_3 + l_4$. The number of operations would be $l_3 + l_4 + 1$.
- Let $L_{2,3,4}$ denotes the merged list of L_2, L_3, L_4. A new length will be $l_{2,3,4} = l_{3,4} + l_2 = l_3 + l_4 + l_2$. The number of operations would be $l_{3,4} + l_2 + 1$.
- Let $L_{1,2,3,4}$ denotes the merged list of L_1, L_2, L_3, L_4. A new length will be $l_{1,2,3,4} = l_{2,3,4} + l_1 = l_1 + l_2 + l_3 + l_4$. The number of operations would be $l_{2,3,4} + l_1 + 1$.
- The total number of operations would be $l_3 + l_4 + 1 + l_{3,4} + l_2 + 1 + l_{2,3,4} + l_1 + 1 = 3l_3 + 3l_4 + 2l_2 + l_1 + 3$

So as $l_1 \leq l_2 \leq l_3 \leq l_4$, merging from the beginning has fewer operations compared to the end one, i.e. merge in an order: 1->2->3->4