Tutorial 4 Solutions

Preparation

4.1

9.

```
Input: $S$, sequence, $n$, length of $S$
Output: index, index of first occurrence of largest member of $S$
procedure findFirst($S$, $n$) {
  $max = s_1$
  $index = 1$
  for $i$=2 to n
    if ($s_i > s_{index}$)
      $index = i$
  return(index)
}  findFirst
```

4.2

4. I will use the table below to show the progress of the insertion sort algorithm, as done in documents at the Help Centre website. The sorted portion of the sequence will be in bold italics, and the unsorted portion in plain text. Note that the algorithm starts with one sorted item, as a sequence of length 1 is always sorted. Note that the sorted portion is always sorted, but not necessarily in final order.

34	20	144	55
20	***34***	144	55
20	***34***	***144***	55
20	***34***	***55***	***144***

12. We shall prove this theorem by induction.

<u>BS (*n*=1)</u>
Now Algorithm 4.2.3 returns a sequence of length 1 immediately, ie sorted (no second item to be out of sequence). Hence the algorithm correctly sorts a sequence of length 1.

<u>IS</u>
Assume that it correctly sorts a sequence of length n, for some $n \geq 1$.
Try to prove that it will correctly sort a sequence of length $n+1$.

Now the algorithm will correctly sort the first *n* items, by our inductive assumption. After that item *n*+1 is copied to *val*. The next few lines repeatedly compare s_j to *val*, and copy it up one item if *val* is smaller. When the while loop finishes, we have located the first item / cell where the test ($j \geq 1$ and $val < s_j$) fails. Either we have reached the start of the sequence, in which case *val* was smaller than all items $s_1, s_2, \ldots, s_n$ and should be inserted at the front, or we have reached the first cell in which $val \geq s_j$. In either case, *val* should be copied to the preceeding cell, which it is. Hence the last item is also inserted into its correct place in the sequence, ie the sequence is correctly sorted.

Proved.

4.3

7.

$$2+4+6+\ldots+2n$$
$$=2(1+2+3+\ldots+n)$$
$$=2*\frac{n(n+1)}{2}=n^2+n$$
$$f(n)=n^2+n \leq n^2+n^2=2n^2$$
$$\text{ie } f(n)=\mathrm{O}\left(n^2\right), \text{ with constant } 2.$$

Also

$$f(n)=n^2+n \geq n^2$$
$$\text{ie } f(n)=\Omega\left(n^2\right), \text{ with constant } 1.$$
$$\text{Hence } f(n)=\Theta\left(n^2\right)$$

13.

$$f(n)=\mathrm{O}(1), \text{ ie } f(n) \leq c_1*1=c_1, n \geq n_1$$
$$f(n)=\Omega(1), \text{ ie } f(n) \geq c_2*1, n \geq n_2$$

For some positive constants c_1, c_2, n_1 and n_2.

$$g(n)=\mathrm{O}\left(n^2\right), \text{ ie } g(n) \leq c_3 n^2, n \geq n_3$$
$$g(n)=\Omega\left(n^2\right), \text{ ie } g(n) \geq c_4 n^2, n \geq n_4$$

For some positive constants c_3, c_4, n_3 and n_4.

Hence, applying the appropriate inequalities, we find

$$f(n)+g(n) \leq c_1*1+c_3 n^2 \leq c_1 n^2+c_3\, n^2=(c_1+c_3)n^2, n \geq n_1, n_3$$
$$f(n)+g(n) \geq c_2*1+c_4 n^2 \geq c_4 n^2, n \geq n_2, n_4$$

Ie $f(n)+g(n)=\mathrm{O}\left(n^2\right)$, $f(n)+g(n)=\Omega\left(n^2\right)$, so $f(n)+g(n)=\Theta\left(n^2\right)$

19. This is a good illustration of the kind of table based approach that is so useful in these questions.

i	*j*	Ops this *i*	Total so far
1	1,2, 3,…,n	*n*	*n*
2	1,2, 3,…,n	*n*	$n+n=2n$
3	1,2, 3,…,n	*n*	$3n$
…			
$2n$	1,2, 3,…,n	*n*	$2n*n=2n^2$

Hence the total number of operations is $2n^2$, which is $\Theta(n^2)$.

22.

i	*j*	*k*	Ops this *j*	Ops this *i*	Total so far
1	1	1	1	*n*	*n*
	2	1	1		
	3	1	1		
	…				
	n	1	1		
2	1	1, 2	2	$2n$	$n+2n$
	2	1, 2	2		
	3	1, 2	2		
	…				
	n	1, 2	2		
3	1	1,2,3	3	$3n$	$n+2n+3n$
	2	1,2,3	3		
	3	1,2,3	3		
	…				
	n	1,2,3	3		
4					$n+2n+3n+4n$
…					
n	1	1,2,3,…,*n*	*n*	n^2	$n+2n+3n+…+n^2$
	2	1,2,3,…,*n*	*n*		
	3	1,2,3,…,*n*	*n*		
	…				
	n	1,2,3,…,*n*	*n*		

Hence the total number of operations is

$$n(1+2+3+...+n)=n*\frac{n(n+1)}{2}=\frac{n^3+n^2}{2}=\Theta\left(n^2\right).$$

4.4

10. (a), (b) not done here.

```
Input: $n$
Output: $S_n=2+4+6+…+2n$
procedure doubleSum($n$) {
  if ($n=1$)
    return(2)
  return(doubleSum($n-1$)+$2n$)
}
```

18. It is easily seen that there is one pair the first month, and this (adult) pair gives birth to one pair the second month, ie $a_1=1$, $a_2=2$.

Now the number of pairs alive in the current month, a_n, is equal to the number of pairs alive in the previous month plus the children of these pairs. Of course the number of pairs in the previous month is a_{n-1}, but of these only the pairs alive the month before that give birth. Hence the number of pairs born is a_{n-1}. So the total number of pairs alive in the current month is $a_n=a_{n-1}+a_{n-2}$.

Now the Fibonacci sequence is 1, 1, 2, … with $f_0=1$, $f_1=1$ and $f_n=f_{n-1}+f_n-2$. So it can easily be seen that the first sequence is just the Fibonnaci sequence shifted left one term, although a proper proof would be inductive. I might do this later if there is plenty of interest.

19. There is still just one pair after 1 year, because they have not yet reproduced.

Tutorial

4.1

20.

```
Input: $A$, matrix of a relation $R$, $n$, the size of the matrix
Output: antisymm, a Boolean value that is true if the relation $R$ is antisymmetric, false if it is not
procedure testAntisymm($A$,$n$) {
  for $i=1$ to $n-1$
    for $j=i+1$ to $n$
      if ($a_{ij}=1$ and $a_{ji}=1$)
        return(False)
  return(True)
}
```

Note that if we find that a_{ij} is 0, we do not need to test a_{ji}, as it cannot prevent R from being antisymmetric, no matter what its value.

4.3

3.

$f(n) = 6n^3 + 12n^2 + 1$

$f(n) \le 6n^3 + 12n^3 + n^3 = 19n^3$

ie $f(n) = \mathrm{O}\left(n^3\right)$, with constant 19

$f(n) \ge 6n^3$

ie $f(n) = \Omega\left(n^3\right)$, with constant 6.

$f(n) = \mathrm{O}\left(n^3\right), \Omega\left(n^3\right)$, ie $f(n) = \Theta\left(n^3\right)$

Note that the n^3 term is the fastest growing term here, hence we know the theta notation will be n^3.

5.

$f(n) = 2\lg(n) + 4n + 3n\lg(n)$

$f(n) \le 2n\lg(n) + 4n\lg(n) + 3n\lg(n) = 9n\lg(n)$

ie $f(n) = \mathrm{O}\left(n\lg(n)\right)$, with constant 9

$f(n) \ge 3n\lg(n)$, ie $f(n) = \Omega\left((n\lg(n)\right)$ with constant 3

$f(n) = \mathrm{O}\left(n\lg(n)\right), \Omega\left((n\lg(n)\right)$ ie $f(n) = \Theta\left(n\lg(n)\right)$

14. I am not going to prove this one fully, see the Help Centre website for the document ComplexityExamps, namely Example 4 and similar exercise. However, the fastest growing term in *f*(*n*) is n^3, and the fastest growing term in *g*(*n*) is an *n*lg(*n*) term. (We can say this without knowing the exact makeup of *g*(*n*), because we are told the theta notation, which is always the fastest growing term.) Hence the fastest growing term in *f*(*n*)+*g*(*n*) is n^3, ie $f(n) + g(n) = \Theta\left(n^3\right)$.

23. Another example of the use of a table.

i	*j*	*k*	Ops this *j*	Ops this *i*	Total so far
1	1	1	1	1	1
2	1	1	1	1+2	1+(1+2)
	2	1, 2	2		
3	1	1	1	1+2+3	1+(1+2)+(1+2+3)
	2	1, 2	2		
	3	1, 2, 3	3		
4					1+(1+2)+(1+2+3)+(1+2+3+4)
…					
n	1	1	1	1+2+3+…+*n*	1+(1+2)+(1+2+3)+…+(1+2+3+…+*n*)
	2	1, 2	2		
	3	1, 2, 3	3		
	…				

	n	1, 2, 3,…,*n*	*n*		

Hence the total number of operations is $1+(1+2)+(1+2+3)+...+(1+2+3+...+n)$. Now each term is of the form $1+2+...+i=\frac{1}{2}i(i+1)=\frac{1}{2}i^2+\frac{1}{2}i$. Hence the number of operations is given by

$$\left(\frac{1}{2}*1^2+\frac{1}{2}*1\right)+\left(\frac{1}{2}*2^2+\frac{1}{2}*2\right)+\left(\frac{1}{2}3^2+\frac{1}{2}*3\right)+...+\left(\frac{1}{2}n^2+\frac{1}{2}n\right)$$
$$=\frac{1}{2}\left(1^2+2^2+3^2+...+n^2\right)+\frac{1}{2}\left(1+2+3+...+n\right)$$
$$=\frac{1}{2}*\frac{1}{6}n(n+1)(2n+1)+\frac{1}{2}*\frac{1}{2}*n(n+1)$$
$$=\frac{1}{12}\left(2n^3+3n^2+n\right)+\frac{1}{4}\left(n^2+n\right)$$
$$=\Theta\left(n^3\right)$$

24. Now on the first trip through the for loop, *j* has the value *n*, and so the operation *x* = *x*+1 is executed at least once, ie the number of operations is $\Theta\left(n\right)$.

The next value that j takes is $\left\lfloor \frac{n}{3} \right\rfloor$, and this is the number operations performed next. Now $\left\lfloor \frac{n}{3} \right\rfloor \le \frac{n}{3}$, ie the number of operations performed now is most $n+\frac{n}{3}$.

Next j takes the value $\left\lfloor \frac{\left\lfloor \frac{n}{3} \right\rfloor}{3} \right\rfloor \le \frac{n}{9}$, hence the number of operations so far is at most $n+\frac{n}{3}+\frac{n}{3^2}$. The next value of j is at most $\frac{n}{27}=\frac{n}{3^3}$, etc. Continuing in this way, we have, for some power of 3, 3^k, the number of operations is at most

$$n+\frac{n}{3}+\frac{n}{3^2}+...+\frac{n}{3^k}=n\left(1+\frac{1}{3}+\frac{1}{3^2}+...+\frac{1}{3^k}\right)$$
$$=n\frac{1-\left(\frac{1}{3}\right)^{k+1}}{1-\frac{1}{3}}$$
$$=\frac{3}{2}n\left(1-\left(\frac{1}{3}\right)^{k+1}\right)$$
$$\le\frac{3}{2}n*1=\frac{3}{2}n$$

Hence the number of operations is $\mathrm{O}(n)$, ie $\Theta(n.)$

4.4

11. If the robot is to walk 1 metre, there is only 1 way of doing this (1). If the robot is to walk 2 metres, there are 2 ways of doing this (11, 2). If the robot is to walk 3 metres, there are 4 ways of doing this (111, 12, 21, 3). Hence W_1=1, W_2=2, W_3=4.

Now if the robot's first step is a 1 metre step, there are n-1 metres left to walk, and this can be done in W_{n-1} ways. However if the robot's first step is 2 metres, then there are n-2 metres left to walk, and that can be done in W_{n-2} ways. And if the robot's first step is 3 metres, then there are n-3 metres left to walk, which can be done in W_{n-3} ways. Hence the total number of ways to walk n metres is given by
$W_n=W_{n-1}+W_{n-2}+W_{n-3}$, with the above initial conditions. This leads to the following recursive algorithm.

```
procedure Walk(n) {
  if (n≤2) then
    return(n)
  else if (n=3) then
        return(4)
  else
    return(Walk(n-1)+Walk(n-2)+Walk(n-3))
}
```