Algorithms - Solutions to Exercise 1

Question 1

The worst case input is $n = 3^k$. In this case the recursion formula is $T(n) = T(n/3) + \Theta(1)$ (where we assume that comparison of n to 1 and check of whether n is divisible by 3 take constant time). It is easy to see that the solution is $T(n) = \Theta(\log n)$. Indeed, $\log n = \log n/3 + 3$.

Question 2

- First solution - calculate F_n by the recursion $F_n = F_{n-1} + 2F_{n-2}$. The recursion ends when $n = 1$ or $n = 2$ where $F_n = 1$. Let $T(n)$ be the number of operations needed to calculate F_n in this way. Then $T(n) = T(n-1) + T(n-2) + \Theta(1)$. You can prove that $T(n)$ grows exponentially with n. If you prove it by induction all you have to note is that there exists $1 < \alpha < 2$ such that $\alpha^{n-1} + \alpha^{n-2} \geq \alpha^n$ (e.g. $\alpha = \sqrt{2}$).

- Second solution: Create an array F of size n to contain the first n numbers. Initialize $F[1] = 1, F[2] = 1$. then conduct:

  ```
  for j=3 up to n do:
       F[j]= F[j-1] + 2F[j-2] ;
  ```

  Correctness and linear running time in n are obvious.

- Third solution: Let M be the matrix $M = \begin{pmatrix} 1 & 2 \\ 1 & 0 \end{pmatrix}$ So $M(x, y) = (x + 2y, x)$. It is easy to see that $M(F_{k+1}, F_k) = (F_{k+2}, F_{k+1})$. Now look at $M^n = M \times M \times ... \times M$ – n times. By induction on the formula above we get: $M^{n-1}(F_1, F_2) = (F_n, F_{n-1})$. So in order to calculate F_n it is enough to calculate M^{n-1}. Multiplying two 2×2 matrixes takes constant time, and recall that we know how to compute M^n using only $\Theta(s)$ multiplications ($s = \lceil \log_2 n \rceil$): we simply compute the square of the result of the previous step in each step until we reach n. Since the number of steps is $\log n$, we end up with an algorithm, linear in s, the size of n (notice that we are assuming that multiplying two numbers takes a constant time, even if they are very large, which is untrue in reality).

Question 3

The functions are asymptoticly decreasing from top-left to bottom-right:

$2^{2^{n+1}}$	2^{2^n}	$(n+1)!$	$n!$	e^n
$n \cdot 2^n$	2^n	$(\frac{3}{2})^n$	$n^{\lg\lg n} =$	$(\lg n)^{\lg n}$
$(\lg n)!$	n^3	$n^2 =$	$4^{\lg n}$	$n \lg n$
$\lg(n!)$	$n =$	$2^{\lg n}$	$(\sqrt{2})^{\lg n}$	$2^{\sqrt{2\lg n}}$
$\lg^2 n$	$\ln n$	$\sqrt{(\lg n)}$	$\ln\ln n$	$2^{\lg^* n}$
$\lg^* n$	$\lg^*(\lg n)$	$\lg(\lg^* n)$	$n^{1/\lg n} = 2$	1

here are several explanations (including all the cases where $f = \Theta(g)$): first one notation: we write $f >> g$, if $f = \Omega(g)$ but f is not $\Theta(g)$.

- $2^{2^n} >> (n{+}1)!$ - to see that, take the log of both functions and note that $2^n >> \Theta(n \lg n)$.
- $n^{\lg\lg n} = (\lg n)^{\lg n}$ - since the log of both functions is $(\lg\lg n \cdot \lg n)$.
- $(\lg n)^{\lg n} >> (\lg n)!$ - simply since $m^m >> m!$ and we may substitute m by $\lg n$.
- $n \lg n = \Theta(lg(n!))$ - as was shown in class.
- $(\sqrt{2})^{\lg n} >> 2^{\sqrt{2\lg n}}$ - to see that, take the log of both functions and note that $\lg n >> \sqrt{(\lg n)}$.
- $\lg^* n = \Theta(\lg^*(\lg n))$ - since $\lg^* n = \lg^*(\lg n) - 1$.
- $n^{1/\lg n} = 2 = \Theta(1)$ - since if we substitute n by 2^m we get: $n^{1/\lg n} = (2^m)^{1/m} = 2$.