Homework 2

1) [1.5 points] Order the following functions by growing rate. Show all your work.

F1 = n

F2 = $\sqrt{n}$

F3 = $n^{1.5}$

F4 = n^2

F5 = nlogn

F6 = nlog(logn)

F7 = n $\log^2 n$

F8 = $n\log(n^2)$

F9 = 2/n

F10 = 2^n

F11 = $2^{n/2}$

F12 = 37

F13 = $n^2\log n$

F14 = n^3

Answer:

Constants <	Logarithms <	Polynomials <	Exponentials
37		n	2^n
2/n		$\sqrt{n}$	$2^{n/2}$
		$n^{1.5}$	
		n^2	
		nlogn	
		nlog(logn)	
		n $\log^2 n$	
		$n\log(n^2)$	
		$n^2\log n$	
		n^3	

! $n\log(n^2) = 2n\log n = \Theta(n\log n)$ so f5 = f8

F9 < f12 < f2 < f1 < f6 < f5 = f8 < f7 < f3 < f4 < f13 < f14 < f11 < f10

You have to prove the reasons of each inequality as explained in the problem session (by taking the limit, etc.) otherwise you will lose some points even if you have given the right answer.

2) [1 points] Consider the following algorithm for the maximum independent set size problem. Assume that n is a power of 2.

```
maxk = 0 ;
k=n/2
for i=1..log2(n)
   if there is an independent set of size k
      maxk = k
      k = k + k/2
  else
      k = k - k/2
```

(Reminder1: independent set problem: Given a graph with n nodes, what is the maximum size of an independent set. There is an $O(n^2 * 2^n)$ algorithm in the slides.

Reminder2: You can check if there is an independent set of size k in a graph of n nodes in $O\left(k^2 * \frac{n^k}{k!}\right) = O(n^k)$ time. See the slides.)

Since the for loop in line 2 iterates at most log2n times, as in binary search, we can find an independent set in at most $O(log_2 n * n^k)$ time. But this is much faster than the complexity given in (Reminder1) above. What is wrong with this complexity calculation? (Hint: use Stirling's approximation to n!)

Answer2) Stirling's approximation: $n! \sim \sqrt{2 * \pi * n}\left(\frac{n}{e}\right)^n$

What is wrong in the analysis is as follows: in reminder2 k is a constant, independent of n. On the other hand, for the given algorithm, k is a function of n. When k=n/2, the complexity is:

$$T(k) = k^2 * \frac{n^k}{k!} = \left(\frac{n}{2}\right)^2 * \frac{n^{\frac{n}{2}}}{\left(\frac{n}{2}\right)!} \sim \left(\frac{n}{2}\right)^2 * \frac{n^{\frac{n}{2}}}{\sqrt{2 * \pi * \frac{n}{2}}\left(\frac{n}{2e}\right)^{\frac{n}{2}}} \sim c * n^{1.5}(2e)^{n/2}$$

which is an exponential (not a polynomial) in n. We would also need to consider the sum of the complexities T(k) for k=1…n, because we do not know whether we will be searching in the interval before or after the current value of k.

Note: There was a bug in the given algorithm. It should be like this:

```
0  maxk = 0 ;
1  k=n/2  ;  step=n/4
2  for i=1..log2(n)
3    if there is an independent set of size k
4      maxk = k
5      k = k + step
6    else
7      k = k – step
8     step = step/2
```

3) [1.5points]

Algorithm SetDisjointness(S1,…,Sn)

```
1  foreach set S_i {
2    foreach other set S_j { ->n-1
3      foreach element p of S_i {  -> with the first  line:  1,2,3,….,n -> n(n+1)/2
4        determine whether p also belongs to S_j -> 1
5     }
6     if (no element of S_i belongs to S_j) -> 1
7       report that S_i and S_j are disjoint -> 1
8   }
9  }
```

Assume that line 4 can be executed in constant (O(1)) time. Assume also that the size of each set is equal to its index, i.e. $|S_i|=i$. What is the worst case running time of this algorithm? $n(n+1)(n-1)/2 = O(n^3)$

4) [1 points]
4a) True or False? Explain your answer.
If f = Θ(h) and g = Θ(h) then f * g = Θ(h*h).

True.

Proof:
f = Θ(h) $\rightarrow$ there exists constants c1 and c2 s.t. c1*h<=f<=c2*h for all n>=n0
g = Θ(h) $\rightarrow$ there exists constants d1 and d2 s.t. d1*h<=g<=d2*h for all n>=n1

Therefore:
c1*d1*h*h <= f*g <=c2*d2*h*h
Define new constants a1=c1*d1 a2=c2*d2
a1*h*h <= f*g <=a2*h*h for all n>= max{n0,n1}
Therefore: f * g = Θ(h*h).

4b) Prove that $\log_2(n) = O(n^{0.5})$

Means that $n^{0.5}$ have to grow faster than or equal to $\log_2(n)$. So compare them, take limit:

$\lim_{n\to\infty} \log_2(n) / n^{0.5}$ -> l'hospital ->
$\lim_{n\to\infty} 1/(n*\ln 2) / 1/(2* n^{0.5}) = \lim_{n\to\infty} (2* n^{0.5}) / (n*\ln 2)$ **= 0 -> $n^{0.5}$ have to grow faster than $\log_2(n)$.**