Assignment 2, CISC 365, Fall 2010

Due Friday, October 1 at the 10:30AM lecture.

1. The input to Algorithm A is an image of k by k pixels, declared as "`array [1..k, 1..k] of integer`". In the worst case, Algorithm A uses $57*k^4*\log k + 23*k^4 + 3*k^3 + 17*k^2 + 1024$ operations. Find the worst-case run time of algorithm A, on an input of size n, and briefly justify your answer. Write your answer using Θ notation, for example $\Theta(n^5)$ or $\Theta(n^5 \log n)$.
Reminder: n is the size of the input (the number of bits needed to write down the input). Don't confuse n and k.

2. This problem reviews subsets and permutations of N elements. N elements can be formed into 2^N different subsets (including the empty set), and can be arranged into N! different permutations. The total number of sets (2^N) is equal to the sum $\sum_{k=1}^{n} \textit{number of sets of size } k$. The case N=3 is illustrated below. Write a similar illustration showing the subsets and permutations for N=4.

Subsets of the set {A, B, C}, to illustrate Equation 2.1

subsets of size 0	{}			"3 choose 0" equals 1
subsets of size 1	{A}	{B}	{C}	"3 choose 1" equals 3
subsets of size 2	{A, B}	{A, C}	{B, C}	"3 choose 2" equals 3
subsets of size 3	{A, B, C}			"3 choose 3" equals 1
				Total is 8 = $2^3 = 2^N$

Permutations of the three elements A, B, C:

There are 3! permutations (also called *sequences*) of 3 items: ABC ACB BAC BCA CAB CBA. Use a systematic way to enumerate the permutations of four items. Get help during office hours if you don't know how to do this.

3. Program P takes T seconds to execute on an input of size N. We double the input size, to 2N. How long does execution take now, if P runs

(a) in time proportional to N

(b) in time proportional to N^2

(c) in time proportional to N^B

(d) in time proportional to lg N (where "lg" is logarithm base 2)

(e) in time proportional to log N (where "log" is logarithm base 10)

(f) in time proportional to 2^N

In (a), "time proportional to N" means that the runtime is kN, for some real number k>0. Give answers that do not depend on N. State the new runtime as a function of T. (It's ok if the answer for part (f) depends on N.)

4. Fill in the following table, to check your results from problem 3. Write down the run time for N=100 and for twice that size (N=200). Assume that k, the constant of proportionality, is equal to 1. It's ok to answer with an expression, e.g. 2^{100}. Row (a) is done: the computation for N=200 takes twice as long as for N=100.

	N=100	N=200	Time for N=200 in terms of time for N=100
(a)	100	200	= 2 * 100
(b)			
(c)			
etc.			

5. For each of the functions defined in (a) to (d), name all of the following sets to which it belongs:

$o(n \log n)$ $O(n \log n)$ $\Theta(n \log n)$ $\Omega(n \log n)$ $\omega(n \log n)$

(a) $g(n) = n - 15$

(b) $h(n) = \sin n$

(c) $A(n) = n \log (n/2)$

(d) $B(n) = n! * n^n$

6. Which of the following are possible? For cases that are possible, give an example of a particular function that fits this description. Otherwise, briefly justify why no functions satisfy this description.

a) $f(n) \in O(n^2)$ and $f(n) \in O(n^3)$

b) $g(n) \in O(n^2)$ and $g(n) \in \Omega(n^3)$

c) $h(n) \in O(n^2)$ and $h(n) \in \Omega(n \log n)$

d) $A(n) \in \Theta(n \log n)$ and $A(n) \in \Omega(n \log n)$ and $A(n) \in O(n \log n)$

e) $B(n) \in \omega(n^n)$

f) $C(n) \in \omega(n^2 \log n)$ and $C(n) \in o(n^3)$

7. (a) True or false: For any positive constant c, $cf(n) \in \Theta(f(n))$.

(b) True or false: For any positive constant c, $f(cn) \in \Theta(f(n))$.

Hint: consider a fast-growing function such as $f(n) = 2^n$.

8. For (a) to (c): if possible, compute the Θ runtime for the given code; if there is insufficient information, then find the best O bound you can. (The variable *n* is the size of the input.)

```
(a) for i:=1 to 50 do {
      <execute code that takes time Θ(n)>
      <execute code that takes time Θ(n log n)>
      if <condition that takes Θ(1) time> then <execute code that takes time Θ(n^2)>
                else <execute code that takes time Θ(n log n)>
      }
(b) for i:=1 to n do {
      for j:=1 to n do {
          <execute code that takes time Θ(n^2)>
          }
      }
(c) for i:=1 to n do {
      for j:=1 to i do {
          <execute code that takes time Θ(n^2)>
          }
      }
```