EECS 477. HOMEWORK 3.

Due on Thursday 10/3/2002 before noon
in mailbox labeled 477 in room 2420 EECS

You must show all work to receive credit!

Please read the statement below and sign your name; otherwise, your homework will not be graded. The text of the College of Engineering's Honor Code can be found at http://honor.personal.engin.umich.edu/

I hereby acknowledge that I understand the College of Engineering's Honor Code and have pledged to uphold it and abide by it.

Signature: ____

1. Asymptotics (25pts)

Order the following nine functions in such a way that $f_k = O(f_{k+1})$. Make sure to replace O by Θ whenever possible, like in the following example: $n = O(n^2), n^2 = \Theta(n^2 + \log n), n^2 + \log n = O(n^3)$.

Here are the functions you will need to arrange:

- $\log(n + 1/n)$
- $n \log n$
- $\log \log n$
- $2^{n - \log n}$
- $(\log n)^n$
- $(5n + \log n/n)2^{(4+\log n)}$
- $3^{\log n - n}$
- $(3 + \log n)!$
- $n^3 + (\log n)^n$

2. k-subsets (30pts)

The questions below will refer to the following piece of code that is also available on the web as a supplement.

```
void print_subset(vector<unsigned>& s) {
  cout << "{ ";
  for(int i=0; i<s.size(); ++i)
    cout << s[i] << " ";
  cout << "}" << endl;
}

void rec_subset(vector<unsigned>& s, int n, int k) {
  if(n<k) {
    return;
  }
  if(k==0) {
    print_subset(s);
    return;
  }
  s[k-1] = n-1;
  rec_subset(s, n-1, k-1);
  rec_subset(s, n-1, k);
}


void generate_subsets(int n, int k) {
  vector<unsigned> s(k);
  rec_subset(s, n, k);
}
```

A(5pts) Prove that a call to the function `generate_subsets(n, k)` ($0 \leq k \leq n$) will print all the subsets of $\{0, \ldots, n-1\}$ that contain k elements.

Important: Suppose that we remove printing commands from the body of `print_subset(s)` function, so that a call to `print_subset(s)` takes *constant time*. The following questions B through F will assume that.

B(5pts) Let $T(n, k)$ be the running time of a call to `rec_subset(s,n,k)` function. Find a recurrence relation for $T(n, k)$. Consider all the cases satisfying $0 \leq k \leq n+1$.

C(5pts) Introduce a new variable $T'(n, k) = T(n, k) + C_1$ and prove that it satisfies the following recurrence relation:

$$T'(n,k) = \begin{cases} T'(n-1,k-1) + T'(n,k) & \text{when } 0 < k \leq n, \\ C_2 & \text{when } k = n+1, \\ C_3 & \text{when } k = 0. \end{cases}$$

What choice of the constant C_1 will make it work?

D(5pts) Let $C_4 = \max(C_2, C_3)$. Prove by induction that $T'(n,n) \le C_4(n+1)$.

E(5pts) Prove by induction that

$$T'(n,k) \le C_4\binom{n+1}{k}.$$

F(5pts) Prove that for $k \le \lfloor n/2 \rfloor$ we have

$$T'(n,k) \le 2C_4\binom{n}{k}$$

Conclusion Thus, we have proven that

$$T(n,k) \le 2C_4\binom{n}{k} - C_1 \le 2C_4\binom{n}{k},$$

that is the time per one generated k-subset is constant (when $k \le \lfloor n/2 \rfloor$).

EXTRA(10pts) What happens when $\lfloor n/2 \rfloor < k < n$? Find an upper bound on the time per one generated k-subset. Is it $O(1)$? $O(n)$? $O(k)$?

3. Asymptotics (30pts)

A function $t(n)$ is defined by recurrence relation:

$$t(n) = \begin{cases} a, & \text{for } n = 1 \\ 4t(\lceil n/3 \rceil) + bn, & \text{for } n > 1 \end{cases}$$

A.(15pts) Prove by induction that $t(n)$ is an eventually non-decreasing function.

B.(15pts) Find the exact order of $t(n)$ in the simplest possible form.

4. Algorithm analysis (15pts)

Consider an algorithm $\mathcal{A}$ that has average-case time complexity $O((n \log n)^2)$ and $\Omega(n \log n)$. For the following statements state whether it could or could not be true, and justify your answer.

A. $\mathcal{A}$ has worst-case time complexity $O(n^2)$.

B. $\mathcal{A}$ has worst-case time complexity $\Theta(n)$.

C. $\mathcal{A}$ has average-case time complexity $\Theta(n^2)$.