15-451 Algorithms, Fall 2000
Problem Set 1

Al Gore-ithm*

August 29, 2000

1 Problem 1

In order to compile this, run **latex sample.tex** on your favorite unix machine. This will create a file called *sample.dvi*. In order to view this, run **xdvi sample.dvi**. You can leave this running and it will automatically update when you re-run latex on your revised document. Alternatively, you can then produce postscript with **dvips -o sample.ps sample.dvi** and view the postscript with ghostview (gv). To print your document, create the postscript file and run it through lpr. Just edit this file to do your next assignment.

2 Problem 2

You can make numbered lists like this:

1. Here is the first step of my amazing algorithm.
2. Here is the second step.
3. Don't all algorithms have to have 3 steps?

and bulleted lists like this:

- blah blah blah...
- ...and furthermore, dividing both sides by e^{x^2} gives us the final result.

The "description" environment is also handy:

Base case: The base case of the algorithm is the case $n < 23$ in which case the result follows by a simple calculation.

Inductive case: We now show that the result follows for general values of n by induction on the height of the tree and fragrance of the leaves.

*Presidential candidate and amateur hacker.

3 Problem 3

Here is some of the mathematical stuff you can do. Inline formulae are done with $ signs, like this $x, x^2, x_i, x_{i+1}, \sqrt{x+y}, f(x) = O(n \log n)$ and $f(x) = \Omega(n \log n)$, therefore $f(x) = \Theta(n \log n)$. There are lots of built-in symbols: $\leq \div \star \dagger \in \subseteq \rightarrow \spadesuit \alpha \beta \gamma$ and built-in functions and operators: $\ln \log \sum \int \max$.

You can also do formulae on a line by themselves with double $ signs:

$$\sum_{i=1}^{n} \frac{\binom{n}{i}}{n!}$$

There is truly no end to the magic.

You can even write pseudo-code if you import the special **code.tex** file:

COUNT(L, R, l, r)

```
1  if L = R + 1
2     then if l = NIL and r = NIL
3             then return 1
4             else return 0
5     else total ← 0
6          for W ← L + 1 to R − 1
7              do while i > 0
8                    do if l ≠ NIL and left[d] ≠ NIL and MATCH(l, left[d])
9                          then leftcount ← COUNT(L, W, next[l], next[left[d]])
10                         else leftcount ← 0
11         return total
```