Dylan Forbes

The Typo Problem

Problem: What is the asymptotic complexity of the number of times the following code will print with respect to n?

```
for (int i=0; i < n; i++) {
        for (int j=0; j < i; j++) {
                if (j & i == 0) {
                        System.out.println(i + " " + j);
                }
        }
 }
```

There are essentially two steps in finding the solution. First, derive a formula $N(i)$ for how many natural numbers j below a given i have a binary representation that has a 0 in every place where the binary representation of i has a 1 (i.e. $j\&i == 0$). Then, use that formula to derive a formula $T(n)$ for the sum of $N(i)$ where i spans 0 to n.

To begin, it is helpful to make a table of all relevant values:

n, i	base2(n)	$J(i) = \{j : 0 \leq j \leq i \wedge j\&i = 0\}$	$N(n) = \lvert J(n) \rvert$	$T(n) = \Sigma_{i=0}^{n} N(i)$
0	0		0	0
1	1	0	1	1
2	10	0,1	2	3
3	11	0	1	4
4	100	0,1,10,11	4	8
5	101	0,10	2	10
6	110	0,1	2	12
7	111	0	1	13
8	1000	0,1,10,11,100,101,110,111	8	21
9	1001	0,10,100,110	4	25
10	1010	0,1,100,101	4	29
11	1011	0,100	2	31
12	1100	0,1,10,11	4	35
13	1101	0,10	2	37
14	1110	0,1	2	39
15	1111	0	1	40
16	10000	(16 items)	16	56
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Here, $J(i)$ is the set of integers j for which the program will print for a given i; $N(i)$ is the number of items in $J(i)$, and thus the number of times the program will print within one iteration of the outer loop (with j spanning 0 to i). Finally, $T(n)$ is the sum of all values of $N(i)$ where $i < n$, and is thus the number of times the program will print for the given value of n.

To figure out an expression for $N(i)$, first notice that there is a definite pattern in its values:

i	1	2		4				8								$\cdots$
$N(i)$	1	2	1	4	2	2	1	8	4	4	2	4	2	2	1	$\cdots$

It seems to be a recursive pattern, where each successive block of values of $N(i)$ is composed of a copy of the previous block with its values doubled, followed by a regular copy. That is, if we let $I(x)$ refer to the 2^x-tuple of the values of $N(i)$ where $2^x \leq i \leq 2^{x+1} - 1$, then:

$$I(x) = \begin{cases} \text{Append}(\{2i | i \in I(x-1)\}, I(x-1)) & : x > 0 \\ \{1\} & : x = 0 \end{cases}$$

Where $\text{Append}(A, B)$ gives the ordered $(|A| + |B|)$-tuple of the values of A followed by the values of B with their orders preserved.

We can use this fact to determine a formula for $N(i)$, by summing up the chunks of $N(i)$ that precede i.

Let $S(x) = \Sigma(I(x))$. (x counts 2^x-long blocks of values, so it is essentially $\lfloor \log_2 n \rfloor$ for a given n.)
Then according to the above formula for $I(x)$,

$$S(x) = \begin{cases} \Sigma\{2i | i \in I(x-1)\} + \Sigma I(x-1)) & : x > 0 \\ \Sigma\{1\} & : x = 0 \end{cases}$$

$$= \begin{cases} 2\Sigma I(x-1) + \Sigma I(x-1)) & : x > 0 \\ \Sigma\{1\} & : x = 0 \end{cases}$$

$$= \begin{cases} 3\Sigma I(x-1) & : x > 0 \\ \Sigma\{1\} & : x = 0 \end{cases}$$

$$= \begin{cases} 3S(x-1) & : x > 0 \\ 1 & : x = 0 \end{cases}$$

Thus, a closed formula for $S(x)$ is

$$S(x) = 3^x$$

And indeed, the table reflects this:

x	$I(x)$	$S(x)$	$[2^x, 2^{x+1} - 1]$ (spanned values of n)
0	{1}	1	$[1]$
1	{2,1}	3	$[2, 3]$
2	{4,2,2,1}	9	$[4, 7]$
3	{8,4,4,2,4,2,2,1}	27	$[8, 15]$
$\vdots$	$\vdots$	$\vdots$	

Now, note that if $n = 2^x$ for some integer $x > 0$, then $S(x-1)$ gives the sum of the 2^x-long "block" of entries in N that precedes $N(n)$'s block—that is, $S(x-1) = \Sigma_{x=n/2}^{n-1}(N(x))$.

So, because $T(n)$ is defined as the sum of all entries $N(x)$ where $0 \leq x \leq n$, $T(n-1)$ must be the sum of all "blocks" that precede n's block (since the blocks are contiguous). And since n is the first entry in its block, because it is assumed to be a power of 2,

$$T(n-1) = \Sigma_{i=0}^{n-1} N(i)$$

Then, because $N(n) = n$ for $n = 2^x$ (this can be seen from the original table),

$$T(n) = T(n-1) + N(n) = T(n-1) + n$$

Thus we have the following formula for T:

$$T(n) = n + \Sigma_{x=0}^{\log_2(n)-1}(S(x))$$

So, combining this with the fact that $S(x) = 3^x$, we have

$$T(n) = n + \Sigma_{x=0}^{\log_2(n)-1}(3^x)$$

which simplifies to

$$T(n) = n + \frac{1}{2}(3^{\log_2(n)} - 1)$$

all for $n = 2^x$.

To see that this is correct, construct another table of values, where $T(n)$ is the value generated iteratively from the table, and $T?(n)$ is the value given by the previously derived formula for $T(n)$ (which we only showed works for $n = 2^x$):

n	$T(n)$	$T?(n)$
0	0	-0.5
1	1	1
2	3	3
3	4	5.35
4	8	8
5	10	10
6	12	14.06
7	13	17.42
8	21	21
9	25	24.77
10	29	28.73
11	31	32.86
12	35	37.17
13	37	41.64
14	39	46.27
15	40	51.06
16	56	56
$\vdots$	$\vdots$	$\vdots$

Although $T?(n) = T(n)$ only for $n = 2^x$, both functions must have the same asymptotic complexity, because $T?(n) \geq T(n)$ for all $n > 0$, but $T?(n)$ never gets too far ahead of $T(n)$ since they are equal for powers of 2.
Thus, $T(n) \in O(n + \frac{1}{2}(3^{\log_2(n)} - 1))$.

Then, finally, because

$$\lim_{n\to\infty} \frac{n + \frac{1}{2}(3^{\log_2(n)} - 1)}{3^{\log_2(n)}} = \lim_{n\to\infty} \frac{3^{\log_2(n)}}{3^{\log_2(n)}} = 1$$

exists and is finite, $T(n) \in O(3^{\log_2 n})$, which is between and $O(n \log n)$ and $O(n^2)$.

So the answer is $O(3^{\log_2 n})$.