Foundations of Computer Science HW 4

Greg Stewart

February 24, 2018

Q1) 6.4

Problem. Prove $P(n): \quad n^7 < 2^n$ for $n \geq 37$.
(a) with induction.

Proof. Take the base case, $n = 37$. $37^7 = 94931877133 < 2^37 = 137438953472$.
Assume that for some n, $P(n)$ is true. So for $P(n+1)$ we have

$$\begin{aligned}(n+1)^7 &= n^7 + 7n^6 + 21n^5 + 35n^4 + 35n^3 + 21n^2 + 7n + 1 \\ &< 2^n + 7n^6 + 21n^5 + 35n^4 + 35n^3 + 21n^2 + 7n + 1 \qquad < 2^n + 2^n = 2^{n+1}\end{aligned}$$

In order for this to be true, we must show $7n^6 + 21n^5 + 35n^4 + 35n^3 + 21n^2 + 7n + 1 < 2^n$ in essentially the same manner. The base case $n = 37$ is true, so now for $n + 1$:

$$\begin{aligned}&7(n+1)^6 + 21(n+1)^5 + 35(n+1)^4 + 35(n+1)^3 + 21(n+1)^2 + 7(n+1) + 1 < \\ &< 2^n + 42n^5 + 210n^4 + 490n^3 + 630n^2 + 434n + 126 \\ &< 2^{n+1}\end{aligned}$$

But now we have to prove it again for the 5-th degree polynomial. We have to do this over and over down to the 1st degree, so I'll just show how this goes:

$$\begin{aligned}&42(n+1)^5 + 210(n+1)^4 + 490(n+1)^3 + 630(n+1)^2 + 434(n+1) + 126 < \\ &< 2^n + 210n^4 + 1260n^3 + 3150n^2 + 3780n + 1806 \\ &< 2^{n+1}\end{aligned}$$

$$\begin{aligned}210(n+1)^4 + 1260(n+1)^3 + 3150(n+1)^2 + 3780(n+1) + 1806 &< 2^n + 840n^3 + 5040n^2 + 10920n + 8400 \\ &< 2^{n+1}\end{aligned}$$

$$\begin{aligned}840(n+1)^3 + 5040(n+1)^2 + 10920(n+1) + 8400 &< 2^n + 2520n^2 + 12600n + 16800 \\ &< 2^{n+1}\end{aligned}$$

$$\begin{aligned}2520(n+1)^2 + 12600(n+1) + 16800 &< 2^n + 5040n + 15120 \\ &< 2^{n+1}\end{aligned}$$

$$
\begin{aligned}
5040n + 15120 &< 2^n + 5040 \\
&< 2^{n+1}
\end{aligned}
$$

where $5040 < 2^n$ for $n \geq 37$. Since we have shown all these in sequence, we see that it's true that $n^7 < 2^n$ for $n \geq 37$. $P(n)$ is confirmed.

(b) with leaping induction.

Proof. Take the base case, $n = 37$. $37^7 = 94931877133 < 2^37 = 137438953472$.

Assume that for some n, $n^7 < 2^n$. Then for $n+3$ we have

$$
\begin{aligned}
(n+3)^7 &= n^7 + 21n^6 + 189n^5 + 945n^4 + 2835n^3 + 5103n^2 + 5103n + 2187 \\
&< n^7 + n \cdot n^6 + n^2 \cdot n^5 + n^3 \cdot n^4 + n^4 \cdot n^3 + n^5 \cdot n^2 + n^6 \cdot n + n^7 \\
&< 8 \cdot n^7 \\
&< 8 \cdot 2^n = 2^{n+3}
\end{aligned}
$$

where the step made in the second line, where coefficients are substituted out for powers of n, is true in all cases for $n \geq 37$.

So for the predicate $P(n) = n^7 < 2^n$ for $n \geq 37$, we have that $P(n) \implies P(n+3)$. This is now just a leaping induction problem, and we can demonstrate a couple more base cases to ensure all n are covered:

$P(37) \implies P(40)$

$P(38)$: $38^7 = 114415582592 < 2^{38} = 274877906944$ so $P(38) \implies P(41)$.

$P(39)$: $39^7 = 137231006679 < 2^{39} = 549755813888$ so $P(39) \implies P(42)$

Since $P(n) \implies P(n+3)$ and we have shown the first three base cases are true, the claim is true for all $n \geq 37$.

Q2) 6.16

Problem. Prove that, for all $n \geq 1$, there is $k \geq 0$ and l odd such that $n = 2^k l$.

Consider all odd numbers greater than 1—all of that can be written as $2^k l$, where $k = 0$ and l is the odd number itself. This is trivial; we just need to show that all even numbers greater than 1 can be written in this form. Consider $n = 2$. This can be written as $2^1 \cdot 1 = 2$, which is the correct form. It is also a power of 2, and, in fact, all powers of 2 can easily be written in this form, as $1 * 2^k$, since 1 is odd. Now, the only case left is when the even n is not a power of 2. If $n/2$ is odd, then $n = 2^1 \cdot \frac{n}{2}$, which is the right form. We can prove the rest by induction, and we have already seen the base case.

Let's assume that all values up to n can be written in the form $n = 2^k l$. Then when $n/2$ is even, $n = \frac{n}{2} \cdot 2^1$, and by the induction hypothesis we know that $n/2$ can be written in this form, so the formula for $n/2$ need only be double, or multiplied by 2^1, to obtain n. Thus, n also satisfies this form.

So we have shown that $\exists k \geq 0$ s.t. $n = 2^k l \forall n \geq 1$.

Q3) 7.4(c)

Problem. Guess a formula for A_n and prove it by induction.

$$
A_0 = 1; A_1 = 2; A_n = 2A_{n-1} - A_{n-2} + 2 \qquad n \geq 2
$$

Let's look at some cases.

$$
\begin{aligned}
A_2 &= 2(2) - 1 + 2 = 5\\
A_3 &= 2(5) - 2 + 2 = 10\\
A_4 &= 2(10) - 5 + 2 = 17\\
A_5 &= 2(17) - 10 + 2 = 26\\
A_6 &= 2(26) - 17 + 2 = 37
\end{aligned}
$$

Based on these results, the formula for A_n appears to be

$$A_n = n^2 + 1$$

We prove this by induction.

The base case here is $2^2 + 1 = 5$, which we know to be true. Let's assume that for all k such that $2 \le k \le n$, $A_k = k^2 + 1 = 2A_{k-1} - A_{k-2} + 2$. Then we have

$$
\begin{aligned}
A_{k+1} &= 2A_k - A_{k-1} + 2\\
&= 2(k^2 + 1) - (k-1)^2 + 1\\
&= 2k^2 + 2 - k^2 + 2k - 1 + 1\\
&= k^2 + 2k + 2\\
&= (k+1)^2 + 1
\end{aligned}
$$

Which is exactly what we were looking for the the $k+1$ case! Thus this formula is correct:

$$A_n = n^2 + 1$$

Q4) 7.6

Problem. Use the function $f(n)$ as defined here, where $n \in \mathbb{N}$

$$f(n) = \begin{cases} 0 & n = 1 \\ f(\lfloor n/2 \rfloor) + f(\lceil n/2 \rceil) + 1 & n > 1 \end{cases}$$

(a) Is f a well-defined function?

f is well-defined as it is not ambiguously defined for any values of n. That is, it is single-valued for all $n \in \mathbb{N}$. This comes from the fact that the function is always increasing.

(b) Tinker and guess a formula for $f(n)$.

Let's take a look at some cases for $n > 1$:

$$
\begin{aligned}
f(2) &= 0 + 0 + 1 = 1\\
f(3) &= 0 + 1 + 2 = 2\\
f(4) &= 1 + 1 + 1 = 3\\
f(5) &= 1 + 2 + 1 = 4\\
f(6) &= 2 + 2 + 1 = 5
\end{aligned}
$$

From this is seems obvious that the formula is

$$f(n) = n - 1$$

(c) Prove your guess.

Take the base case, $n = 1$: $f(1) = 1 - 1 = 0$, which we know to be true.

For the n case, assume that $f(n) = n - 1$ is true, for all n from 0 to n. Then we have two possibilities for $n + 1$: either $n + 1$ is even or odd.

if $n + 1$ is even, then

$$\begin{aligned}
f(n+1) &= f(\frac{n+1}{2}) + f(\frac{n+1}{2}) + 1 \\
&= \frac{n+1}{2} - 1 + \frac{n+1}{2} - 1 + 1 \\
&= \frac{2n+2}{2} - 1 \\
&= n
\end{aligned}$$

which is exactly what we're looking for; and if $n + 1$ is odd, then

$$\begin{aligned}
f(n+1) &= f(\frac{n}{2}) + f(\frac{n+2}{2}) + 1 \\
&= \frac{n}{2} - 1 + \frac{n+2}{2} - 1 + 1 \\
&= \frac{2n+2}{2} - 1 \\
&= n
\end{aligned}$$

which is again exactly what we want for the $n+1$ case! Thus we have shown that this formula is correct:

$$f(n) = n - 1$$

Q5) 7.8(o)

Problem. Prove $P(n): \quad F_n^2 + F_{n+1}^2 = F_{2n+1}$.

Note that $F_n = F_{n-1} + F_{n-2}$

$$F_1^2 + F_2^2 = 1 + 1 = 2 = F_3$$

Let's assume that $P(1) \wedge P(2) \wedge \cdots \wedge P(n-1) \wedge P(n)$ are all true. For $P(n+1)$, we have

$$\begin{aligned}
F_{2n+3} &= F_{n+1}^2 + F_{n+2}^2 \\
&= (F_n + F_{n-1})^2 + (F_{n+1} + F_n)^2 \\
&= F_n^2 + F_{n-1}^2 + 2F_nF_{n-1} + F_{n+1}^2 + F_n^2 + 2F_{n+1}F_n \\
&= F_{n+1}^2 + 2F_n^2 + F_{n-1}^2 + 2F_n(F_{n+1} + F_{n-1}) \\
&= F_{2n+1} + F_n^2 + F_{n-1}^2 + 2F_n(F_{n+1} + F_{n-1})
\end{aligned}$$

$F_nF_{n-1} + F_nF_{n+1} = F_{2n}$, so

$$\begin{aligned}
F_{2n+3} &= F_{2n+1} + F_n^2 + F_{n-1}^2 + 2F_{2n} \\
&= F_{2n+1} + F_{2n-1} + 2F_{2n} \\
&= F_{2n+2} + F_{2n+1}
\end{aligned}$$

This is exactly the formula for the Fibonacci number F_{2n+3}, so we have shown $P(n)$ to be true!

Q6) 7.26

Problem. Give pseudocode for a recursive function that computes 3^{2^n} on input n.

```
function powers(sq, n):
  input: n, the number to which you raise 3^2
         sq, a running square of 3 (exponentiation by squaring)
          when first called, sq = 3
  output: 3^2^n

  if n = 0: return sq
  return powers(sq*sq, n-1)
```

(a) Prove that your function correctly computes 3^{2^n} for every $n \geq 0$.

This is a simplified exponentiation by squaring algorithm. We have to account for fewer cases because 3 is always raised to a power of 2.

Consider what the algorithm is doing for each recurrence: squaring the value of sq. Starting with 3, this means we have the current value being squared each time in this sequence:

$$3 \implies 3^2 \implies (3^2)^2 = 3^4 \implies (3^4)^2 = 3^8 \implies \cdots \implies 3^{2^n}$$

Every time the function recurses, the number is squared again, which means the exponent is doubled. The exponent is thus doubled n times, making the exponent by definition 2^n. This is exactly what we want:

$$3^{2^n}$$

(b) Obtain a recurrence for the runtime T_n. Guess and prove a formula for T_n.

Each recurrence does one multiplication, which we will call $O(1)$, and this occurs n times, so the recurrence in the runtime is

$$T_n = T_{n-1} + O(1)$$

Obviously, this makes the formula for T_n simply $T_n = n$, as each recurrence reduces n by 1, down to the $n = 0$ case, and only one computation is done per recurrence.

Q7) 7.31(c)

Problem. Give recursive definition for the set S in each of the following cases.

$$S = \{\text{all strings with the same number of 0's and 1's}\}$$

Base case: $x = \epsilon$, where ϵ is just an empty string.
Rest of the set:
$x \in S \implies (0x1 \wedge 1x0) \in S$

Q8) 8.7(a)-(c)

Problem. The set P of parenthesis strings has defintion

$$\begin{aligned} \epsilon &\in P \\ x \in P &\implies [x] \in P \\ x, y \in P &\implies xy \in P \end{aligned}$$

(a) Which of [[[]]][, [][[]][[]] and [[][][] are in P? Give derivations.

Only [][[]][[]] is in P. This is derived in this order:

$$x = \epsilon \implies [] \implies [[]]$$

$$y = \epsilon \implies [] \implies [[]]$$

$$x, y \implies [[]][[]]$$

$$z = \epsilon \implies []$$

$$z, [[]][[]] \implies [][[]][[]]$$

(b) Give two different derivations of [][][[]]
First:

$$x = \epsilon \implies [] \implies [[]]$$

$$y = \epsilon \implies []$$

$$x, y \implies [][[]]$$

$$x, [][[]] \implies [][][[]]$$

Second:

$$x = \epsilon \implies []$$

$$y = \epsilon \implies []$$

$$x, y \implies [][]$$

$$z = \epsilon \implies [] \implies [[]]$$

$$[][], z \implies [][][[]]$$

(c) Prove by structural induction that every string in P has even length.

For the empty string ϵ the length $l_\epsilon = 0$, which is even. For the next case, where we get $x = []$, the length is $l_1 = 2$, which is also even. Let's assume that for each parent $x, y \in P$, l_x and l_y are even. There are two constructor rules.

Rule 1: $x \in P \implies [x] \in P$. Since l_x is even, and the child in this case has two characters added, that means l_{child} is even. So it is true for this case.

Rule 2: $x, y \in P \implies xy \in P$. Since l_x and l_y are even, when they are concatenated the length is $l_x + l_y$, and two even numbers added together give an even number, so l_{xy} is even too.

Since both constructor rules give an even string length, every string in P has even length.