CS 2110: INDUCTION

DISCUSSION TOPICS

The following ideas are suggestions for how to handle your discussion classes. You can do as much or as little of this as you want. You can either present at the board, or you can have students work on the problems. You all know your discussion sections better than I do. The goal of this document is to give you some ideas of what to cover and how to present it, in order to be consistent with the lecture.

With that said, as a bare minimum, I suggest that you cover one of the weak induction proofs (such as Example **??**) and the MergeSort recurrence.

Weak Induction Proofs

The primary goal of discussion is to give students practice with induction. In order to be consistent with the lecture, I highly recommend that you use the following style:

- **At the start, identify the "inductive" strategy needed to solve the problem.** This is a recursive function or algorithm that will make induction easier. Sometimes it is given, but sometimes the student has to come up with it. This is covered in the lecture, but I have an example of it below.
- **Once the strategy is defined, write up in a highly structured format.** Clearly identify the base step, the inductive hypothesis and the inductive step. Again, I have an example of this below. The idea is that the inductive strategy is what they do on scrap paper, but a proof needs to be clearly written up.

The motivation of induction for this class is recurrence relations, so some of the best examples to use are the classic "summation identities". Here are some classic ones to use. Pick whichever ones you like best:

Example 1. Let

$$S(n) = 0^2 + 1^2 + 2^2 + \ldots + n^2 = \sum_{i=0}^{i} i^2 \tag{1}$$

Show that

$$S(n) = \frac{n(n+1)(2n+1)}{6} \tag{2}$$

for all $n \geq 0$.

Example 2. Let

$$S(n) = 1^2 + 3^2 + 5^2 + \ldots + (2n+1)^2 = \sum_{i=0}^{i} (2i+1)^2 \tag{3}$$

Show that

$$S(n) = \frac{(n+1)(2n+1)(2n+3)}{3} \tag{4}$$

for all $n \geq 0$.

Example 3. Let

$$S(n) = 0 \cdot 2^0 + 1 \cdot 2^1 + 2 \cdot 2^2 + \ldots + n2^n = \sum_{i=0}^{i} i2^i \tag{5}$$

Show that

$$S(n) = (n-1)2^{n+1} + 2 \tag{6}$$

As an demonstration, let's do Example 3. In order to do induction, we need a recursive function involved. We suspect that we can write the function in (5) as the recursive function

$$S_R(n) = \begin{cases} 0 & \text{if } n = 0 \\ S_R(n-1) + n2^n & \text{if } n > 0 \end{cases}$$

However, this is just our strategy. We have not proven this yet. That means that what we really want to prove now is

$$S(n) = S_R(n) = (n-1)2^{n+1} + 2 \tag{7}$$

That is, we now have three different things: the iterative function on the left, the recursive function in the middle, and the closed form expression on the right. We want to prove that all three are equal.

Write (7) up as claim, and then proceed with the induction proof. Clearly label the following three steps:

Base Step: We prove the claim for $n = 0$. Clearly $S(0) = 0$ and $S_R(0) = 0$. Furthermore,

$$(0-1)2^{0+1} + 2 = -2 + 2 = 0$$

Therefore, all three are equal and we are done with the base step.

Inductive Hypothesis: Assume that $S(k) = S_R(k) = (k-1)2^{k+1} + 2$ holds true for some arbitrary k.

Inductive Step: We need to prove that the claim is true for $k+1$. In other words, we need to show

$$S(k+1) = S_R(k+1) = (k)2^{k+2} + 2$$

First, we show $S(k+1) = S_R(k+1)$. But that is really easy since

$$\begin{aligned} S_R(k+1) &= S_R(k) + (k+1)2^{k+1} && \text{Definition of } S_R \\ &= S(k) + (k+1)2^{k+1} && \text{Induction Hypothesis} \\ &= 0 + \ldots + k2^k + (k+1)2^{k+1} && \text{Definition of } S \\ &= S(k+1) && \text{Definition of } S \end{aligned}$$

For the other half,

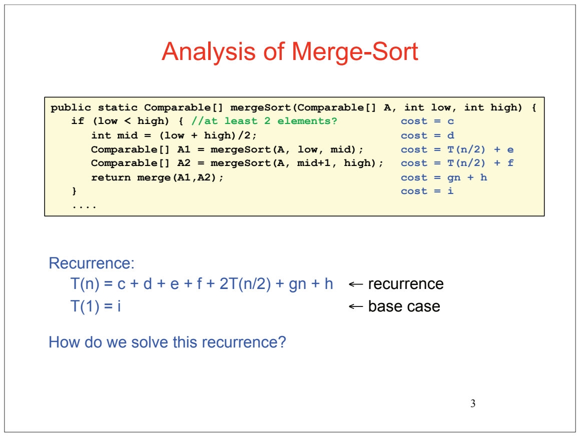
$$\begin{aligned} S_R(k+1) &= S_R(k) + (k+1)2^{k+1} && \text{Definition of } S_R \\ &(k-1)2^{k+1} + 2 + (k+1)2^{k+1} && \text{Induction Hypothesis} \\ &= 2k2^{k+1} + 2 = k2^{k+2} + 2 && \text{Algebra} \end{aligned}$$

Conclusion: Since our choice of k in the argument above was arbitrary, we know that our property in (7) holds for all $n \geq 0$.

Recurrences

The entire motivation of induction is so that students can prove the complexity of MergeSort. Unfortunately, because of scheduling, there is a one week gap between the induction lecture and the recurrence lecture. Therefore, we would like you to introduce the concept of a recurrence, and then use that with (strong) induction to show how to compute the complexity.

That means, you are going to have to motivate recurrence relations. I suggest that you use the following two lecture slides as a guide:



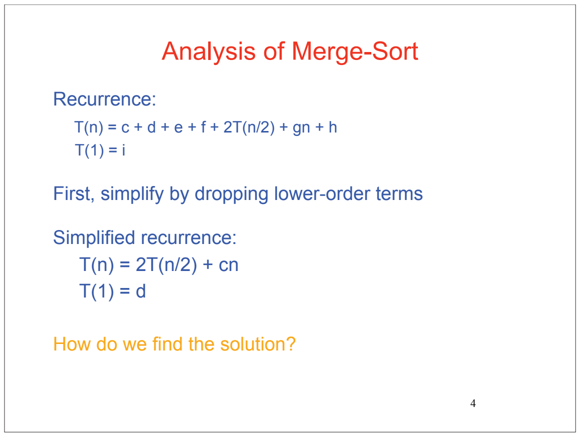
Analysis of Merge-Sort

```
public static Comparable[] mergeSort(Comparable[] A, int low, int high) {
   if (low < high) { //at least 2 elements?          cost = c
      int mid = (low + high)/2;                        cost = d
      Comparable[] A1 = mergeSort(A, low, mid);        cost = T(n/2) + e
      Comparable[] A2 = mergeSort(A, mid+1, high);     cost = T(n/2) + f
      return merge(A1,A2);                             cost = gn + h
   }                                                   cost = i
   ....
```

Recurrence:
T(n) = c + d + e + f + 2T(n/2) + gn + h ← recurrence
T(1) = i ← base case

How do we solve this recurrence?

3



Analysis of Merge-Sort

Recurrence:
T(n) = c + d + e + f + 2T(n/2) + gn + h
T(1) = i

First, simplify by dropping lower-order terms

Simplified recurrence:
T(n) = 2T(n/2) + cn
T(1) = d

How do we find the solution?

4

Because $T(0) = 0$, you can actually write the MergeSort recurrence as

$$T(n) = \begin{cases} 2c & \text{if } n = 2 \\ 2T(n/2) + cn & \text{if } n > 2 \end{cases}$$

What want to prove is that $T(n)$ is $O(n \log n)$. However, this is not a property of the form $P(n)$. You might want to point out to the students that a property $P(n)$ has to allow you to substitute values into it, like $P(k)$ or $P(2)$. But if you plug that in for $O(n \log n)$, you get something like $O(2 \log 2)$, which makes no sense.

Instead, we have to phrase what we want to prove as the following claim:

$$T(n) \leq cn \log n$$

This needs to be done by strong induction. Again, be very clear about the steps.

First of all, point out that there is no need for a recursive strategy. We are already given a recursive function. When you are given a recursive function, you can usually (but not always) do the inductive proof directly on the recursive function. It was only because we had no recursive function to start with in the earlier examples that we had to figure out what our strategy was before we started. With that observation out of the way, we proceed with the usual steps.

Base Step: We prove the claim for $n = 2$ (we are starting at 2, there is no need to start earlier). Then $2c \log 2 = 2c \cdot 1 = 2c$, so we are done.

Strong Inductive Hypothesis: Assume that, for some arbitrary k, $T(m) \leq cm \log m$ holds true for **all** $2 \leq m \leq k$.

Inductive Step: We need to prove that the claim is true for $k+1$.

$$\begin{aligned}
T(k+1) &= 2T((k+1)/2) + c(k+1) && \text{Definition of } T \\
&\leq 2\big[c\frac{k+1}{2}\log\frac{k+1}{2}\big] + c(k+1) && \text{Strong Induction Hypothesis} \\
&= c(k+1)\log\frac{k+1}{2} + c(k+1) && \text{Algebra} \\
&= c(k+1)(\log(k+1) - 1) + c(k+1) && \text{Algebra} \\
&= c(k+1)\log(k+1) && \text{Algebra}
\end{aligned}$$

In doing this proof, try to make very clear why we needed strong induction, and why weak induction will not work.

Once this is done, you might want to shake things up a bit to see if they are following. Take the following line in MergeSort:

```
int mid = (low + high)/2;
```

Change this line to

```
int mid = low+1;
```

Try to the exercise again to compute the new complexity. See if you can get them to guess the complexity before you go through the proof.

If the students are following, and you are having a good time, you can do a few other recurrences as well. In each case, set up the recurrence relation, and then use induction to prove the complexity.

- **Linear Search**: $T(n) = T(n1) + 1$ is the recurrence. Prove $T(n) = O(n)$.
- **Binary Search**: $T(n) = T(n/2) + 1$ is the recurrence. Prove $T(n) = O(\log n)$.
- **QuickSort Worst-Case**: $T(n) = T(n-1) + n$ is the recurrence. Prove $T(n) = O(n^2)$.

However, if the students are not following, you might want to cut your loses with MergeSort and move on to the next suggestion.

Flawed Induction

A fun exercise that helps students to build up their understanding of induction is to presented a flawed induction proof that shows something ridiculous. Then have them try to find the error. Here are two good examples:

Example 4. Claim: Let $S(n) = 1 + \ldots + n$. For every $n \geq 1$

$$S(n) = \frac{(n+\frac{1}{2})^2}{2} \tag{8}$$

Base Step: The equation (8) is true for $n = 1$.

Inductive Hypothesis: Assume that (8) is true for k.

Inductive Step: We prove (8) is true for $k+1$. Using algebra,

$$S(k+1) = S(k) + (k+1) = \frac{(k+\frac{1}{2})^2}{2} + (k+1) = \frac{(k^2+n+\frac{1}{4})}{2} + k + 1$$
$$= \frac{(k^2+3n+\frac{9}{4})}{2} = \frac{(k+\frac{3}{2})^2}{2} = \frac{[(k+1)+\frac{1}{2}]^2}{2}$$

Example 5. Claim: $5n = 0$ for every $n \geq 0$.

Base Step: $5 \cdot 0 = 0$.

Strong Inductive Hypothesis: Assume that $5 \cdot m = 0$ for every $m \leq k$.

Inductive Step: We prove the claim is true for $k+1$. We write $k+1 = m_0 + m_1$ where m_0, m_1 are both less than $k+1$. By the strong induction hypothesis, $5m_0 = 0$ and $5m_1 = 0$. Therefore,

$$5(k+1) = 5(m_0+m_1) = 5m_0 + 5m_1 = 0 + 0 = 0$$

Additional Problems

If you have any other induction problems that you really like, feel free to introduce them. However, I suggest that you stick to the following guidelines.

- Avoid the geometric induction puzzles. Stick to properties of functions, or clearly numeric properties, for now. Remember this is 2110, not 2800.
- Always remember to use k in the proof. Do not induct on n (e.g. assume true for $n-1$ and then prove for n), even though that may have appeared on class slides in the past. This is confusing to a new student.