Computer Science 2405
April 20, 2020

Happy Monday, April 20.

This is the last lecture that I will send you. It contains examples of finding general solutions to nonhomogeneous recurrences and basics of solving divide-and-conquer recurrences.

Exam 4 is on Monday, April 27. Spend your remaining time for this course working exercises and studying for the exam. I am behind in grading, but I will try to catch up.

The final exam is scheduled for Friday, May 1 at 11:00-1:30. I will make the exam available at 10:30 on May 1 and expect it to be submitted by 2:00, unless you make separate arrangements with me. I will try to make it not too long.

Examples of getting a general solution to a nonhomogeneous recurrence

The general solution of a recurrence works for all initial conditions. It has unknown constants that you need to adjust to the initial conditions. You find those constants by solving a set of linear equations as was done in the previous lecture.

Today, we will only find the general solutions. We will not try to find constants for particular initial conditions.

Example. Find the general solution the following recurrence.

$$a_n = 3a_{n-1} - 2a_{n-2} + n^2$$

Answer.

1. The associated homogeneous recurrence (lecture 2020-04-15: Associated homogeneous recurrences) is

   $$a_n = 3a_{n-1} - 2a_{n-2}$$

   and its characteristic equation (lecture 2020-04-08: Characteristic equations) is

   $$r^2 - 3r + 2 = 0.$$

   Since $r^2 - 3r + 2 = (r-1)(r-2)$, the solutions are $r = 1$ and $r = 2$.

2. Since the characteristic equations has two different solutions, the general solution of the associated homogeneous recurrence (2020-04-08: Degree 2 homogeneous recurrences without repeated roots) is

$$a_n = c_1(1^n) + c_2(2^n)$$

where constants c_1 and c_2 depend on the initial values.

3. $n^2 = n^2(1^n)$, so $s = 1$. (See 2020-04-17: Solving nonhomogeneous linear recurrences of a restricted form.) Notice that s is a solution of the characteristic equation with multiplicity 1. (That is because, in the factored form $(r-1)(r-2)$, factor $(r-1)$ appears one time.) According to the rule for solving nonhomogeneous recurrences, the general solution is

$$\begin{aligned} a_n &= n^1(p_2n^2 + p_1n + p_0)(1^n) + c_1(1^n) + c_2(2^n) \\ &= p_2n^3 + p_1n^2 + p_0n + c_1 + c_2(2^n) \end{aligned}$$

where p_0, p_1 and p_2 are additional constants that depend on the initial values.

Since there are 5 unknown constants p_0, p_1, p_2, c_0 and c_1, you would need to use the values of a_0, a_1, a_2, a_3 and a_4 to get 5 linear equations. The solution of those linear equations gives the values of the constants. But we won't do that. We will stop at the general solution.

Example. Find the general solution the following recurrence.

$$a_n = 4a_{n-1} - 4a_{n-2} + 3^n.$$

Answer.

1. The associated homogenous recurrence is

$$a_n = 4a_{n-1} - 4a_{n-2}.$$

The characteristic equation is

$$r^2 - 4r + 4 = 0.$$

Since $r^2 - 4r + 4 = (r-2)^2$, there is just one solution, $r = 2$, and it has multiplicity 2.

2. Since the characteristic equation has only one solution, the general solution of the homogeneous recurrence (2020-04-08: Degree 2 homogeneous recurrences with repeated roots) is

$$a_n = c_1 2^n + c_2 n 2^n.$$

3. The $F(n)$ part in the nonhomogenenous recurrence is $F(n) = 3^n$. Since 3 is not a solution of the characteristic equation, the general solution of recurrence

$$a_n = 4a_{n-1} - 4a_{n-2} + 3^n$$

is

$$a_n = p_1 3^n + c_1 2^n + c_2 n 2^n$$

(2020-04-17: Solving nonhomogeneous linear recurrences of a restricted form).

Divide-and-conquer recurrences

This material is from section 8.3 of Rosen.

Some algorithms work by cutting the input into two or more pieces, using the same algorithm on each piece, then combining the solutions of the pieces into a solution for the original input.

A simple example is ***Merge Sort***, which sorts a sequence L of values into nondescending order as follows.

1. If there are 0 or 1 numbers in L, then the answer is L. (A list of 0 or 1 number is already sorted, trivially).

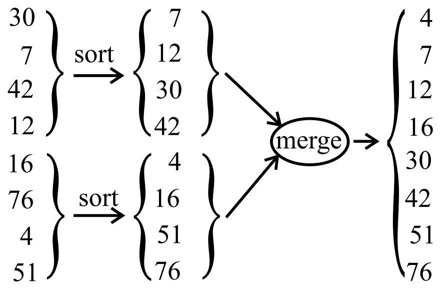
2. So suppose that L has at least 2 numbers in it.

   (a) Cut list L in half, giving lists L_1 and L_2. If the length of L is odd, make L_1 have length $\lfloor n/2 \rfloor$ and make L_2 have length $\lceil n/2 \rceil$.

   (b) Sort L_1 and L_2 using Merge Sort. Suppose that S_1 and S_2 are the two sorted lists.

   (c) Merge S_1 and S_2 into a single sorted list.

Here is an illustration.



Merging two sorted sequences is easy. We use head(L) to mean the first value in list L.

```
merge(L1, L2):
  output = empty list
  while L1 and L2 are both nonempty
    If head(L2) < head(L1)
      remove head(L2) from L2 and put it at the end of output
    else
      remove head(L1) from L1 and put it at the end of output
    end if
  end while
  If L1 is empty
    add all of L2 to the end of output
  else
    add all of L1 to the end of output
  end if
```

Suppose the sum of the lengths of $L1$ and $L2$ is N. The merge algorithm looks at each value just once. So it takes cN time to merge $L1$ and $L2$, for some constant c.

Let's define $T(N)$ to be the time that it takes to sort a list of length N using Merge Sort. Looking at the Merge Sort algorithm, it is clear that, for $N > 1$,

$$T(N) = T(\lfloor N/2 \rfloor) + T(\lceil N/2 \rceil) + cN.$$

That is a typical divide-and-conquer recurrence. Notice that, when N is a power of 2, $T(N)$ is defined in terms of $T(N/2)$ rather than in terms of $T(N-1)$.

We will now look at how to solve such recurrences.

Change of variable

Suppose that $N = 2^n$ is a power of 2. Then $\lfloor N/2 \rfloor = N/2$ and $\lceil N/2 \rceil = N/2$. The recurrence for Merge Sort (for $N > 1$) simplifies to

$$T(N) = 2T(N/2) + cN. \tag{1}$$

Now we do a *change of variable*. Instead of expressing equations in terms of N, we can express them in terms of n. Since $N = 2^n$, we can replace N by 2^n in Equation (1). That gives

$$T(2^n) = 2T(2^{n-1}) + c2^n \tag{2}$$

since $2^n/2 = 2^{n-1}$. Now let's define function A by

$$A(n) = T(2^n)$$

for all values of n. Then it must also be the case that

$$A(n-1) = T(2^{n-1}).$$

Replacing $T(2^n)$ by $A(n)$ and $T(2^{n-1})$ by $A(n-1)$ in Equation (2) gives

$$A(n) = 2A(n-1) + c2^n.$$

To put that in a more familiar form, lets define $a_n = A(n)$. Then the recurrence is

$$a_n = 2a_{n-1} + c2^n. \tag{3}$$

We have defined $T(N)$, $A(n)$ and a_n all in terms of the amount of time that Merge Sort takes to sort a list of N values. We would like to find a closed-form solution to each of those, so that we get an idea of how much time Merge Sort takes. But Equation (3) is a linear nonhomogeneous recurrence that we know how to solve!

1. The associated homogeneous recurrence is

   $$a_n = 2a_{n-1}$$

   and its characteristic equation is

   $$r - 2 = 0.$$

   The characteristic equation has one solution, $r = 2$.

2. The general solution of the associated homogeneous recurrence is

   $$a_n = d2^n$$

   for some constant d.

3. Using the formula for solving nonhomogeneous linear recurrences (2020-04-17), the $F(n)$ term in the nonhomogeneous recurrence if $c2^n$, so $s = 2$. Notice that s is a solution of the characteristic equation (of multiplicity 1). So the general solution of Equation (3) is

   $$\begin{aligned} a_n &= n^1(p_0)2^n + d2^n \\ &= d2^n + p_0 n2^n \end{aligned}$$

Now that we know what a_n is, let's find $T(N)$. Replacing a_n by $A(n)$ gives

$$A(n) = d2^n + p_0 n 2^n. \tag{4}$$

But $A(n) = T(2^n)$. We have said that $N = 2^n$. So $n = \log_2(N)$. Replacing $A(n)$ by $T(N)$ and n by $\log_2(N)$ in Equation (4) gives

$$T(N) = dN + p_0 N \log_2(N). \tag{5}$$

We haven't worked out the values of constants d and p_0. It turns out that $p_0 \neq 0$. Function $N\log_2(N)$ grows faster than N, so it is the dominant term. Throwing out the smaller term dN from Equation (5) gives

$$T(N) \approx p_0 N \log_2(N).$$

That is, it takes some constant times $N\log_2(N)$ to sort a list of N values using Merge Sort.

Big-Theta

We need a way of saying that two functions $f(n)$ and $g(n)$ "grow at the same rate." The following is standard mathematical notation for that. Θ is a capital Greek letter theta.

Suppose that $f(n)$ and $g(n)$ are two functions. We say that $f(n) = \Theta(g(n))$ if there exist two positive real constants u and v so that, for all sufficiently large values of n, $ug(n) \le f(n) \le vg(n)$. That is, $f(n)$ is between one constant times $g(n)$ and another constant times $g(n)$.

For example, $n^2 = \Theta(4n^2)$ because

$$0.25(4n^2) \le n^2 \le 1(4n^2).$$

Notice that $4n^2 = \Theta(n^2)$ because

$$1(n^2) \le 4n^2 \le 4(n^2).$$

For any two functions $f(n)$ and $g(n)$,

$$f(n) = \Theta(g(n)) \iff g(n) = \Theta(f(n)).$$

If $f(n)$ and $g(n)$ are polynomials, then $f(n) = \Theta(g(n))$ if and only if polynomials $f(n)$ and $g(n)$ have the same degree. So $n^3 + 2n = \Theta(n^3)$.

A rule for solving divide-and-conquer recurrences

The idea of change of variable can be used to derive a useful general rule for solving divide-and-conquer recurrences.

It turns out that floors and ceilings don't matter. You can replace $T(\lfloor n/k \rfloor)$ and $T(\lceil n/k \rceil)$ by $T(n/k)$ in your recurrence, ignoring the fact that n/k might not be an integer.

Master Theorem

Suppose that $T(n)$ is an increasing function that satisfies recurrence relation

$$T(n) = aT(n/b) + cn^d$$

whenever $n = b^k$ for some positive integer k. Also suppose that b is a positive integer and a, c and d are real numbers where

$$\begin{aligned} a &\ge 1 \\ c &> 0 \\ d &\ge 0 \end{aligned}$$

Then

$$T(n) = \begin{cases} \Theta(n^d) & if a < b^d \\ \Theta(n^d \log_2(n)) & if a = b^d \\ \Theta(n^{\log_b(a)}) & if a > b^d \end{cases}$$

Example. Use the Master Theorem to find a solution to

$$T(n) = 2T(n/2) + n.$$

Answer. Compare this recurrence with the general form

$$T(n) = aT(n/b) + cn^d$$

in the Master Theorem. Here, $a = 2$, $b = 2$, $c = 1$ and $d = 1$. Notice that $a = b^d$. So the solution is

$$T(n) = \Theta(n \log_2(n)).$$

Example. Use the Master Theorem to solve

$$T(n) = 3T(n/2) + n.$$

Answer. Compare this recurrence with the general form

$$T(n) = aT(n/b) + cn^d$$

in the Master Theorem. Here, $a = 3$, $b = 2$, $c = 1$ and $d = 1$. Notice that $a > b^d$. So the solution is

$$t(n) = \Theta(n^{log_2(3)}).$$

$log_2(3)$ is rougly 1.6. So $T(n)$ is roughly proportional to $n^1.6$.

Example. Use the Master Theorem to solve

$$T(n) = 2T(n/3) + n^2.$$

Answer. Compare this recurrence with the general form

$$T(n) = aT(n/b) + cn^d$$

in the Master Theorem. Here, $a = 2$, $b = 3$, $c = 1$ and $d = 2$. Notice that $a < b^d$. So the solution is

$$T(n) = \Theta(n^2).$$

Example. Use the Master Theorem to solve

$$T(n) = 4T(n/2) + n^2.$$

Answer. Compare this recurrence with the general form

$$T(n) = aT(n/b) + cn^d$$

in the Master Theorem. Here, $a = 4$, $b = 2$, $c = 1$ and $d = 2$. Notice that $a = b^d$. So the solution is

$$T(n) = \Theta(n^2 log_2(n)).$$

Example. Use the Master Theorem to solve

$$T(n) = 7T(n/2) + n^2.$$

Answer. Compare this recurrence with the general form

$$T(n) = aT(n/b) + cn^d$$

in the Master Theorem. Here, $a = 7$, $b = 2$, $c = 1$ and $d = 2$. Notice that $a > b^d$. So the solution is

$$T(n) = \Theta(n^{log_2(7)}).$$

$\log_2(7)$ is roughly 2.8.

Exercises

Do exercises 14 from homework set 6 and the following exercises.

1. Find a Θ estimate of $T(n)$, where $T(n)$ satisfies the following recurrence.
$$T(n) = 2T(n/4) + n.$$
2. Find a Θ estimate of $T(n)$, where $T(n)$ satisfies the following recurrence.
$$T(n) = 4T(n/4) + n.$$
3. Find a Θ estimate of $T(n)$, where $T(n)$ satisfies the following recurrence.
$$T(n) = 5T(n/4) + n.$$

4. Find a Θ estimate of $T(n)$, where $T(n)$ satisfies the following recurrence.
$$T(n) = 16T(n/4) + n.$$

5. Find a Θ estimate of $T(n)$, where $T(n)$ satisfies the following recurrence.
$$T(n) = 16T(n/4) + n^2.$$