Unrolling a Recurrence

Remington Greko, Tyler Gutowski, and Spencer Hirsch

February 27, 2023

Work with your team to write a report showing your knowledge of the unrolling technique to solve a recurrence.

Submit the team's report of Canvas. Include a matrix indicating who did what.

Sums

Give a closed form expression for the sums below.

1. Sigma notation:

$$\sum_{i=0}^{i=n-1} 1 \tag{1}$$

   Closed form expression:

$$n \tag{2}$$

2. Sigma notation:

$$\sum_{i=0}^{i=n-1} 2^i \tag{3}$$

   Closed form expression:

$$2 - 2^{n-1} \tag{4}$$

3. Sigma notation:

$$\sum_{k=0}^{k=n} (n!/k!(n-k)!) \tag{5}$$

   Closed form expression:

$$2^n \tag{6}$$

4. Sigma notation:

$$\sum_{k=0}^{k=n-1} r^k \tag{7}$$

Closed form expression:

$$(r^n - 1/(r-1) \tag{8}$$

5. Sigma notation:

$$\sum_{i=0}^{i=n-1} kr^{k-1} \tag{9}$$

Closed form expression:

$$knr^{k-1} \tag{10}$$

Unrolling a Recurrence

I like this recurrence solving technique because it only involves simple algebraic substitutions and an ability to compute closed form expressions for common sums. Here is an example of unrolling that you can use as a template for other problems.

Consider the recurrence:

$$T(n) = T(n/2) + 1, T(0) = 1$$

The idea is simple: if

$$T(n) = T(n/2) + 1, T(1) = 0$$

then by substitution (unrolling)

$$T(n/2) = T(n/4) + 1, T(1) = 0$$

n can be divded by 2 a only a finite number of times before 1 is reached ending the unrolling. (It is olay to assume n = 2^p or p = logn and the unrolling ends after p stages.) In another step

$$T(n) = T(n/4) + 1 + 1, T(1) = 0$$

Convince yourself this results in adding 1 to itself p times computing the sume of values in the first column of Pascal's triangle.

Therefore,

$$T(n) = p = log(n)$$

Make the connection that the recurrence models the binary search algorithm.

Here are some additional problems to flex and sharpen your skills.

1. Solve the recurrence

$$T(n) = T(n-1) + n, T(0) = 1$$

$$T(n) = T(n-1)^2 + n$$

$$T(n) = n^2 - 2n + 1 + n$$

$$T(n) = n^2 - (2-1)n + 1$$

$$T(n) = n^2 - (2-1)n + 1$$

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

Therefore,

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

Mention the algorithm that is described by this recurrence.

Bubble sort is an algorithm that has a recurrence of $O(n^2)$.

2. Solve the recurrence

$$T(n) = 2T(n/2) + n, T(0) = 1$$

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

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

$$T(n) = nlog((n) - nlog(2) + n)$$

$$T(n) = nlog(n) - n + n$$

$$T(n) = nlog(n)$$

$$T(n) = nlog(n)$$

Therefore,

$$T(n) = nlog(n)$$

Mention the algorithm that is described by this recurrence.

Mergesort is a recursive algoirthm that has a time complexity of O(nlog(n)).

3. Solve the recurrence

$$T(n) = 2T(n-1) + 1, T(1) = 1$$

$$T(n) = 2(2T(n-2) + 1) + 1$$

$$T(n) = 2^2T(n-2) + 2 + 1$$

$$T(n) = 2^3T(n-3) + 2^2 + 2 + 1$$

$$T(n) = 2^kT(n-k) + 2^k - 1 + 2^k - 2 + ... + 2^2 + 2 + 1$$

$$T(n) = 2^nT(0) + 1 + 2 + 2^2 + ... + 2^k - 1$$

$$T(n) = 2^n * 1 + 2^k - 1$$

$$T(n) = 2^n + 2^n - 1$$

$$T(n) = 2^n + 1 - 1$$

Therefore,

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

What famous problem is described by this recurrence?

he Tower of Hanoi problem is a famous computer science problem which is solved in an O(2^n) time complexity.

Name	**Section**
Remington Greko	Unrolling recurrence part 2c
Tyler Gutowski	Closed form expression sums
Spencer Hirsch	Unrolling a recurrence part 2a and 2b