Lecture 16 Divide & Conquer Recurrences Michael Monagan

February 26, 2021 10:47 AM

Lecture 16: Divide and Conquer Algorithms and Recurrences

Please use the notes on Canvas not 10.6 Grimaldi.

Assignment # 4 is due on Monday @ 11pm.

What is the fastest algorithm for sorting an array of n numbers ?
What is the fastest algorithm to multiply two polynomials of degree n ?

Sorting Algorithms

Suppose we want to sort an A of n integers e.g.

$$A = \begin{array}{|c|c|c|c|c|c|c|}\hline 9 & 3 & 11 & 2 & 6 & 13 & 5 \\ \hline\end{array}$$

To compare sorting algorithms, by tradition, we count the number of comparisons they do. Bubblesort does exactly $n(n-1)/2$ comparisons. Mergesort does at most $n\log_2 n - n + 1$ comparisons. Below is a table for various values of n comparing the number of comparisons of these two algorithms.

	n	4	16	64	1024	10^6
Bubblesort	$n(n-1)/2$	6	120	2016	523776	approx 5×10^{11}
Mergesort	$n\log_2 n - n + 1$	5	49	321	9217	approx 20×10^6

50x

For $n = 10^6$ Mergesort does a factor of over 25,000 fewer comparisons!
Demo Mergesort

```
void Merge( int A[], int n1, int B[], int n2, int C[] ) {
// Merge the sorted arrays A of length n1 and B of length n2 into C
   int i,j,k;
   i = j = k = 0;
   while( i<n1 && j<n2 )
      if( A[i]<B[j] ) { C[k] = A[i]; i++; k++; }
      else { C[k] = B[j]; j++; k++; }
   while( i<n1 ) { C[k] = A[i]; i++; k++; }
   while( j<n2 ) { C[k] = B[j]; j++; k++; }
   return;
}
```

C Code.

Figure: C code for merging two sorted arrays A and B into the array C

Let $n = n_1 + n_2$. The number of comparisons that mergesort does is $\leq n_1 + n_2 - 1 = n - 1$.

The Mergesort Algorithm

```
 1: void Mergesort( int A[], int n, int C[] ) {
 2: // sort A[0],A[1],...,A[n] into ascending order
 3: // C is an array of length n for working storage
 4:    int n1,n2,*B;
 5:    if( n<=1 ) return;
 6:    n1 = n/2;
 7:    n2 = n-n1;
 8:    B = A + n1;
 9:    Mergesort(A,n1,C); // sort the first half of A
10:    Mergesort(B,n2,C); // sort the second half of A
11:    Merge(A,n1,B,n2,C); // merge A and B into C
12:    for( i=0; i<n; i++ ) A[i] = C[i]; // copy C into A
13:    return;
14: }
```

$\leftarrow C_1 = 0.$

Let C_n be the # comparisons that Mergesort does.

$$C_n \leq \underbrace{C_{n_1}}_{\text{line 9}} + \underbrace{C_{n_2}}_{\text{line 10}} + \underbrace{n-1}_{\text{merge at line 11.}} = 2C_{n/2} + n - 1$$

Assume $n = 2^k$ for some $k \geq 0$. to simplify the analysis. $n_1 = n_2 = \frac{n}{2}$

Solving $C(n) \leq 2C(n/2) + n - 1$ with $C(1) = 0$.

$$n = 2^k$$

$$\log_2 n = k \log_2 2 = k.$$

$$n/2 = 2^{k-1}$$

$$1 \quad C_n \leq 2C_{n/2} + n - 1$$

$$2 \quad C_{n/2} \leq 2(2C_{n/4} + n/2 - 1) = 2^2 C_{n/4} + n - 2$$

$$2^2 \quad C_{n/4} \leq 2^2(2C_{n/8} + n/4 - 1) = 2^3 C_{n/8} + n - 4$$

$$\vdots$$

$$\tfrac{n}{2} = 2^{k-1} \quad C_2 \leq 2^{k-1}(2C_1 + 2 - 1) = 2^k C_1 + n - 2^{k-1}$$

$$2^k \quad C_1 = 2^k \cdot 0 = 0$$

$$C_n \leq n-1 + n-2 + n-2^2 + \cdots + n - 2^{k-1}$$

$$= \sum_{i=0}^{k-1} n - 2^i = \sum_{i=0}^{k-1} n - \underbrace{(1+2+4+\cdots+2^{k-1})}_{=2^k-1} = n \cdot k - (2^k - 1)$$

$$= n\log_2 n - n + 1$$

Divide and Conquer Algorithms

Suppose we are given a problem of size n.

S1: Divide the problem into $a \geq 2$ subproblems of approximately the same size, say size b. Algorithm Mergesort divided A into $a = 2$ subproblems of size $n_1 = n/2$ and $n_2 = n - n_1$.

S2: Solve the subproblems recursively using the same "divide-and-conquer" approach.

S3: Combine the results from the subproblems to obtain the final solution. Algorithm Mergesort merges two sorted arrays of size n_1 and n_2 into one sorted array of size n.

Let $f(n)$ be the # operations to solve the problem of size n and let $h(n)$ " " " " for steps S_1 and S_3.

Assume $n = b^k$ for some $k \geq 0$ then

$$f(n) = \underbrace{a\, f(n/b)}_{\text{Step } S2} + \underbrace{h(n)}_{\text{steps } S_1 \& S_3} \quad \text{for } n > 1. \qquad \text{Use } f(1) = c \text{ for some constant } c.$$

Example. Adding an array of numbers.

```
1: double Add( double A[], int n ) {
2: // Add A[0]+A[1]+...+A[n-1]
3:     double s1,s2,*B;  int n1,n2;
4:     if( n==1 ) return A[0];
5:     n1 = n/2; n2 = n-n1;
6:     s1 = Add(A,n1); // s1 = A[0]+A[1]+...+A[n1-1]
7:     B = A + n1; // B is a subarray of A starting at n1
8:     s2 = Add(B,n2); // s2 = A[n1]+A[n2+1]+...+A[n-1]
9:     return s1+s2;
10: }
```

you | I

$A = [\,9\,|\,1\,|\,3\,|\,7\,\|\,2\,|\,2\,|\,6\,|\,1\,]$

$s_1 = 20 + s_2 = 11 = 33.$

Let a_n be the # of additions in step 9. Assume $n = 2^k$.

$a_n = \underset{\text{line 6}}{a_{n/2}} + \underset{\text{line 8}}{a_{n/2}} + \underset{s_1+s_2 \text{ in line 9.}}{1} = 2a_{n/2} + 1$

$2 \cdot a_{n/2} = 2(2a_{n/4} + 1) = 2^2 a_{n/4} + 2$

$2^2 a_{n/4} = 2^2(2a_{n/8} + 1) = 2^3 a_{n/8} + 4$

$\vdots$

$2^{k-1} a_2 = 2^{k-1}(2a_1 + 1) = 2^k a_1 + 2^{k-1}$

$2^k a_1 = 2^k\, 0 = 0$

$a_n = 1 + 2 + 4 + \cdots 2^{k-1}$

$= 2^k - 1$

$= n - 1.$

Maple examples using the rsolve command.

A second order recurrence

```
> re := a(n) = 5*a(n-1) - 6*a(n-2);
```

$$re := a(n) = 5\,a(n-1) - 6\,a(n-2)$$

```
> rsolve( {re,a(0)=1,a(1)=4}, a(n) );
```

$$2\,3^n - 2^n$$

The mergesort recurrence

```
> re := c(n) = 2*c(n/2) + n-1;
```

$$re := c(n) = 2\,c(n/2) + n - 1$$

```
> expand( rsolve( {re, c(1)=0}, c(n) ) );
```

$$-n + \frac{\ln(n)\,n}{\ln(2)} + 1$$

$\log_2 n = \frac{\ln n}{\ln 2}$