

lecture 16

more recurrences

- matrix power (fast Fibonacci)
- merge sort
- fast multiplication

Matrix Multiplication

Let A, B be 2×2 matrices.

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

$$\equiv \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{bmatrix}$$

4 additions, 8 multiplications

$$\text{power}(A, n) \equiv \underbrace{A \cdot A \cdot \dots \cdot A}_n$$

$\text{power}(A, n)$ {

```

if n == 0
  return 1
else {
  tmp ← power(A, n/2)
  if n % 2 == 0
    return tmp * tmp
  else
    return tmp * tmp * A
}

```

} matrix multiplications

$$t(n) \leq t\left(\frac{n}{2}\right) + C \text{ is } O(\log n)$$

↑ worst case: two matrix multiplications

Fast Fibonacci (clever)

$$F(n+2) = F(n) + F(n+1)$$

$$\begin{bmatrix} F(n+2) \\ F(n+1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} F(n+1) \\ F(n) \end{bmatrix}$$

Claim:

$$\begin{bmatrix} F(n+2) \\ F(n+1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n \begin{bmatrix} F(2) \\ F(1) \end{bmatrix}$$

Proof (by induction)

Base case ($n=1$). Verify easily.

Induction Step. Assume for $n=k$
Prove for $n=k+1$

$$\begin{aligned} & \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{k+1} \begin{bmatrix} F(2) \\ F(1) \end{bmatrix} \\ &= \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^k \begin{bmatrix} F(2) \\ F(1) \end{bmatrix} \\ &= \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} F(k+2) \\ F(k+1) \end{bmatrix} = \begin{bmatrix} F(k+3) \\ F(k+2) \end{bmatrix} \end{aligned}$$

induction hypothesis

```

mergesort (list) {
  if (list.size == 1)
    return list
  else {
    partition list into two approximately
    equal size lists l1, l2
    return merge (mergesort(l1),
                  mergesort(l2))
  }
}

```

Let $t(n)$ be the number of operations required to mergesort n elements.

$$t(n) = cn + 2t\left(\frac{n}{2}\right)$$

↑
merge two sorted lists
of size $\frac{n}{2}$ elements

$$\begin{aligned}
 t(n) &= cn + 2t\left(\frac{n}{2}\right) \\
 &= cn + 2\left[c\frac{n}{2} + 2t\left(\frac{n}{4}\right) \right] \\
 &= c(n+n) + 4t\left(\frac{n}{4}\right) \\
 &= c(n+n) + 4\left(c\frac{n}{4} + 2t\left(\frac{n}{8}\right) \right) \\
 &= c(n+n+n) + 8t\left(\frac{n}{8}\right) \\
 &= c(n+n+n+\dots+n) + nt(1) \\
 &= cn \log n, \text{ if } n \text{ is a power of } 2 \\
 &\quad t(1) = 0
 \end{aligned}$$

What if we modify mergesort so that we use an $O(n^2)$ algorithm when $n \leq 4$?
Is the modified algorithm still $O(n \log n)$?

(See recurrences exercises.)

Fast Multiplication

$$\begin{array}{l}
 x = \begin{array}{|c|c|} \hline x_1 & x_0 \\ \hline \end{array} \\
 y = \begin{array}{|c|c|} \hline y_1 & y_0 \\ \hline \end{array} \\
 \longleftarrow \hspace{10em} \longrightarrow \\
 n \text{ digits}
 \end{array}$$

- $x * y$ has $2n$ digits.
- grade school algorithm is $O(n^2)$

$$\begin{array}{l}
 x = \begin{array}{|c|c|} \hline x_1 & x_0 \\ \hline \end{array} \\
 y = \begin{array}{|c|c|} \hline y_1 & y_0 \\ \hline \end{array}
 \end{array}$$

$$\begin{aligned}
 &(x_1 * 10^{\frac{n}{2}} + x_0) * (y_1 * 10^{\frac{n}{2}} + y_0) \\
 &= x_1 y_1 * 10^n + (x_0 y_1 + x_1 y_0) * 10^{\frac{n}{2}} + x_0 y_0
 \end{aligned}$$

$$\begin{aligned}
 t(n) &= t\left(\frac{n}{2}\right) + 2t\left(\frac{n}{2}\right) + t\left(\frac{n}{2}\right) + cn \\
 &= 4t\left(\frac{n}{2}\right) + cn
 \end{aligned}$$

$$\begin{aligned}
t(n) &= 4t\left(\frac{n}{2}\right) + cn \\
&= 4\left(4t\left(\frac{n}{4}\right) + c\frac{n}{2}\right) + cn \\
&= 4 \cdot 4 \cdot t\left(\frac{n}{4}\right) + 2cn + cn \\
&= 4 \cdot 4 \cdot \left(4t\left(\frac{n}{8}\right) + c\frac{n}{4}\right) + cn(2+1) \\
&= 4 \cdot 4 \cdot 4 \cdot t\left(\frac{n}{8}\right) + cn(4+2+1) \\
&\leq 4^{\log_2 n} t\left(\frac{n}{n}\right) + cn\left(\frac{n}{2} + \dots + 4+2+1\right) \\
&= (2^{\log_2 n})^2 t(1) + cn(n-1) \\
&= c n^2 \cdot t(1) + cn(n-1)
\end{aligned}$$

which is $O(n^2)$

Trick:

$$\begin{aligned}
&x_1 y_0 + x_0 y_1 \\
&= (x_1 + x_0) * (y_1 + y_0) - x_1 y_1 - x_0 y_0 \\
&\quad \underbrace{\hspace{10em}}_{t\left(\frac{n}{2}+1\right)^*} \quad \underbrace{\hspace{2em}}_{t\left(\frac{n}{2}\right)} \quad \underbrace{\hspace{2em}}_{t\left(\frac{n}{2}\right)}
\end{aligned}$$

* Note: $t\left(\frac{n}{2}+1\right) = t\left(\frac{n}{2}\right) + cn$

$$\begin{array}{l}
x = \\
y =
\end{array}
\begin{array}{|c|c|}
\hline
x_1 & x_0 \\
\hline
y_1 & y_0 \\
\hline
\end{array}$$

$$(x_1 + 10^{\frac{n}{2}} + x_0) * (y_1 * 10^{\frac{n}{2}} + y_0)$$

$$= x_1 y_1 * 10^n + (x_0 y_1 + x_1 y_0) * 10^{n/2} + x_0 y_0$$

$$\begin{aligned}
t(n) &= t\left(\frac{n}{2}\right) + t\left(\frac{n}{2}+1\right) + t\left(\frac{n}{2}\right) + cn \\
&= 3t\left(\frac{n}{2}\right) + c_1 n
\end{aligned}$$

$$t(n) = 3t\left(\frac{n}{2}\right) + n$$

One can show

$$t(n) \text{ is } O\left(n^{\log_2 3}\right)$$

$$n^{\log_2 3} \approx n^{1.6} < n^2$$

Similar tricks are used to solve many other problems.