CSC236 fall 2014
more complexity: mergesort

Danny Heap
heap@cs.toronto.edu
BA4270 (behind elevators)
http://www.cdf.toronto.edu/~csc236h/fall/
416-978-5899

Using Introduction to the Theory of Computation,
Chapter 3
Outline

mergesort

divide and conquer (recombine)

using the Master Theorem

Notes
recurrence for MergeSort

MergeSort(A, b, e):
    if b == e: return
    m = \( \frac{b + e}{2} \)
    MergeSort(A, b, m)
    MergeSort(A, m+1, e)

# merge sorted A[b..m] and A[m+1..e] back into A[b..e]
for i = b, ..., e:
    B[c] = A[c]
    c = b
    d = m+1
    for i = b, ..., e:
        if d > e or (c <= m and B[c] < B[d]):
            A[i] = B[c]
            c = c + 1
        else: # d <= e and (c > m or B[c] >= B[d])
            A[i] = B[d]
            d = d + 1
Unwind (repeated substitution)

\[ T(n) = 2T(n/2) + n + 1 \quad \text{for } n = 2^k, \text{ some } k \in \mathbb{N} \]

\[ T(n) = n \log n + 2n - 1 \]
Prove that $T$ is non-decreasing

- see similar proof, 2 slides ahead
- you should read and work out

See Course Notes, Lemma 3.6 Exercise: Prove the recurrence for binary search is non-decreasing
Prove $T \in O(n \lg n)$ for general case

$T(n) = T([n/2]) + T([n/2]) + n + 1$

Claim: $\exists c \in \mathbb{R}^+, \exists B \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq B \Rightarrow T(n) \leq cn \lg n$

Proof:

Pick $c = \frac{8}{2}$, $B = \frac{2}{2}$. Then $c \in \mathbb{R}^+, B \in \mathbb{N}$, assume $n \in \mathbb{N}, n \geq 0$.

Then $T(n) \leq T(\hat{n}), \hat{n} = \lfloor \lg \hat{n} \rfloor$ since $T$ is monotonic increasing.

$\leq \hat{n} \lg \hat{n} + 2^\hat{n} - 1 \tag{\text{proven for power of 2}}$

$\leq 2n \lg (2n) + 4n - 1 \tag{\text{if } \frac{\hat{n}}{2} \leq n \leq \hat{n} = \frac{\hat{n}}{2}}$

$\leq 2n \lg n + 6n - 1 \tag{\text{if } \frac{\hat{n}}{2} \leq n \leq \frac{\hat{n}}{2}}$

$\leq 8n \lg n \tag{\text{if } n \geq 2}$

Conclude: $\exists c \in \mathbb{R}^+, B \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq B \Rightarrow T(n) \leq cn \lg n$. (So $T(n) \in \Theta(n \lg n)$.)
Prove \( T(n) \) is monotonic non-decreasing

\[
T(n) = \begin{cases} 
1 & \text{if } n = 1 \\
1 + \max\{T(\lceil n/2 \rceil), T(\lfloor n/2 \rfloor)\} & \text{if } n > 1
\end{cases}
\]

\( P(n) : \forall m \in \mathbb{N}, 1 \leq m \leq n \Rightarrow T(m) \leq T(n) \)

Claim \( \forall n \in \mathbb{N}, n \geq 1 \Rightarrow P(n) \)

Proof (complete induction)

Assume \( n \in \mathbb{N}, n \geq 1, \) and \( P(i), 1 \leq i < n. \) Must show \( P(n). \)

Case \( n \geq 3. \) It is enough to show \( T(n-1) \leq T(n), \) since by transitivity of \( \leq \) and \( P(n-1) \) we have \( T(k) \leq T(n-1) \) \( \leq T(n), \forall 1 \leq k \leq n-1. \) Note that \( 1 \leq \lceil n/2 \rceil, \lceil n/2 \rceil, n-1 < 1 \) since \( n \geq 2. \)

\[
T(n-1) = 1 + \max\left\{ T\left( \lceil n/2 \rceil \right), T\left( \lfloor n/2 \rfloor \right) \right\} \geq 1 + \max\left\{ T\left( \lceil n/2 \rceil \right), T\left( \lfloor n/2 \rfloor \right) \right\} \\
\leq 1 + \max\left\{ T\left( \lceil n/2 \rceil \right), T\left( \lfloor n/2 \rfloor \right) \right\} \\
\]

# By \( P(\lceil n/2 \rceil) \) and \( P(\lfloor n/2 \rfloor), T(\lceil n/2 \rceil) \geq T(\lceil n/2 \rceil), T(\lfloor n/2 \rfloor) \geq T(\lfloor n/2 \rfloor) \)

\[
= T(n)
\]

(Base cases \( P(1) + P(2) \) easy).
Prove $T(n)$ is monotonic non-decreasing

$$T(n) = \begin{cases} 
1 & \text{if } n = 1 \\
1 + \max\{T(\lceil n/2 \rceil), T(\lfloor n/2 \rfloor)\} & \text{if } n > 1
\end{cases}$$

If $1 \leq m \leq n$ \Rightarrow $T(m) \leq T(n)$

Prove $P(n)$, $\forall n \in \mathbb{N}$, $n > 0$, by complete induction.

**Induction Hypothesis**, IH: Assume $n \in \mathbb{N}$ is $> 0$, and assume $P(1)$ for $1 \leq i < n$. I must now show $P(n)$

- **Base case**, $n = 1$: In this case we must have $m = 1 = n$, since $1 \leq m \leq n$. That means $T(m) = 1 \leq 1 = T(n)$, So $P(1)$ holds.

**Case** $n > 1$: It’s convenient to consider subcases:

- **Subcase** $m = 1$: In this case we have:

  $$T(m) = T(1) = 1 \leq 1 + 1 = 2 \leq 1 + \max\{T(\lceil n/2 \rceil), T(\lfloor n/2 \rfloor)\}$$

  By IH for $1 \leq \lceil n/2 \rceil \leq \lfloor n/2 \rfloor < n$, $T(\lceil n/2 \rceil) \geq T(\lfloor n/2 \rfloor) \geq T(i) = 1$

  So $P(n)$ holds

  \[\text{continued}\]
Prove $T(n)$ is monotonic non-decreasing

$$T(n) = \begin{cases} 
1 & \text{if } n = 1 \\
1 + \max\{T(\lfloor n/2 \rfloor), T(\lceil n/2 \rceil)\} & \text{if } n > 1 
\end{cases}$$

**Case $n > 1$**

Subcase 1: In this case

\[
T(m) = 1 + \max(T(\lceil m/2 \rceil), T(\lfloor m/2 \rfloor)) \quad \text{# defn, } m > 1
\]

\[
\leq 1 + \max(T(\lceil n/2 \rceil), T(\lfloor n/2 \rfloor)) \quad \text{# By IH for } 1 \leq \lfloor m/2 \rfloor, \lceil m/2 \rceil, \lfloor n/2 \rfloor, \lceil n/2 \rceil < n
\]

\[
\leq T(\lfloor m/2 \rfloor) + T(\lceil m/2 \rceil) \leq T(n)
\]

So, $T(n)$ holds in this case.

I conclude, by Complete Induction, $\forall n \in \mathbb{N}^+, P(n)$. 
General case revisit...

Class of algorithms: partition problem into roughly equal subproblems, solve, and recombine:

\[ T(n) = \begin{cases} 
  k & \text{if } n \leq B \\
  a_1 T(\lceil n/b \rceil) + a_2 T(\lfloor n/b \rfloor) + f(n) & \text{if } n > B 
\end{cases} \]

where \( B, k > 0, a_1, a_2 \geq 0, \) and \( a = a_1 + a_2 > 0. \) \( f(n) \) is the cost of splitting and recombining.
Master Theorem
(for divide-and-conquer recurrences)

If $f$ from the previous slide has $f \in \theta(n^d)$, then

$$T(n) = \begin{cases} 
\theta(n^d) & \text{if } a < b^d, \\
\theta(n^d \log n) & \text{if } a = b^d, \\
\theta(n^{\log_b a}) & \text{if } a > b^d
\end{cases}$$
Proof sketch

1. Unwind the recurrence, and prove a result for $n = b^k$
   
   Various terms, including $\sum_{i=0}^{K} \left( \frac{a}{bd} \right)^i \Rightarrow a < b^d$ geometric series $\Rightarrow a = b^d \Rightarrow a > b^d$ ugly

2. Prove that $T$ is non-decreasing
   
   Some idea as MergeSort

3. Extend to all $n$, similar to MergeSort
   
   Same idea as several slides
multiply lots of bits
what if they don’t fit into a machine instruction?

\[
\begin{array}{c}
1101 \\
\times 1011 \\
\end{array}
\]

of \(2^n\) bits
\(\sim n^2\)
\(n\) copies of \(n\) bits
\(\Theta(n^2)\)

Suppose 32 thousand-bit numbers, rather than 32-bit...
divide and recombine recursively...

\[
\begin{array}{c|c}
11 & 01 \\
\times 10 & \text{11}\end{array}
\]

\[
x y = 2^n x_1 y_1 + 2^{n/2} (x_1 y_0 + y_1 x_0) + x_0 y_0
\]

\(x_1\) stands for \(x_1 \times 2^{n/2}\)

\(y_1\) stands for \(y_1 \times 2^{n/2}\)

So...

\(x_1 y_1 \times 2^{n/2} \times 2^{n/2} = x_1 y_1 \cdot 2^n\)

\(x_1 y_0 \times 2^{n/2} + y_1 x_0 \times 2^{n/2} = (x_1 y_0 + y_1 x_0) 2^{n/2}\)
compare costs

\[ a = 4 > b = 2^1, \text{ so} \]
\[ \Theta(n^{\log_2 4}) = \Theta(n^2) \text{ [oh no!]} \]

\[ \Theta(n^2) \]

$n$ $n$-bit additions versus:

1. divide each factor (roughly) in half \[ b = 2 \]
2. multiply the halves (recursively, if they're too big)
3. combine the products with shifts and adds

4. recursive products: \( x_1 y_1, x_1 y_0, y_0 x_1, x_0 y_0 \)

So \( a = 4 \)

proporional to \( n \), So \( f(n) \in \Theta(n^d), d=1 \)

more than, say, 64-bit machine
Gauss's trick

Want to reduce a number of recursive multiplications

\[ xy = 2^n x_1 y_1 + x_0 y_0 + 2^{n/2} ((x_1 + x_0)(y_1 + y_0) - x_1 y_1 - x_0 y_0) \]

2 mult (save value)

3 mult

1 mult. (save value)
Gauss’s payoff
lose one multiplication

1. divide each factor (roughly) in half
2. sum the halves
3. multiply the sum and the halves Gauss-wise
4. combine the products with shifts and adds
closest point pairs
see Wikipedia
divide-and-conquer v0.1
how many close points fit?
an \( n \lg n \) algorithm

\( P \) is a set of points

1. Construct (sort) \( P_x \) and \( P_y \)
2. For each recursive call, construct \( L_x, L_y, R_x, R_y \)
3. Recursively find closest pairs \((l_0, l_1)\) and \((r_0, r_1)\), with minimum distance \( \delta \)
4. \( V \) is the vertical line splitting \( L \) and \( R \), \( D \) is the \( \delta \)-neighbourhood of \( V \), and \( D_y \) is \( D \) ordered by \( y \)-ordinate
5. Traverse \( D_y \) looking for minimum pairs 15 places apart
6. Choose the minimum pair from \( D_y \) versus \((l_0, l_1)\) and \((r_0, r_1)\).