CSC236 fall 2018

divide and conquer: as a design tool
recursive correctness

Danny Heap
heap@cs.toronto.edu / BA4270 (behind elevators)
http://www.teach.cs.toronto.edu/~heap/236/F18/
416-978-5899

Using Introduction to the Theory of Computation,
Chapter 3
Outline

divide and conquer (recombine)

D&C: multiply quickly

D&C: closest points

binary search

Notes
general D&C case
revisit...

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

where \( b, k > 0, a_1, a_2 \geq 0, \) and \( a_1 + a_2 > 0. \) \( f(n) \) 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

$$T(n) \in \begin{cases} 
\theta(n^d) & \text{if } a < b^d \\
\theta(n^d \log_b n) & \text{if } a = b^d \\
\theta(n^{\log_b a}) & \text{if } a > b^d 
\end{cases}$$
multiply lots of bits
what if they don’t fit into a machine instruction?

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

\[
\begin{array}{c}
1101 \\
1101 \\
0000 \\
1101 \\
\hline
10001111
\end{array}
\]

n copies, Theta(n^2)
n column-wise additions, Theta(n^2)
divide and recombine

recursively... $2^n = n$ left-shifts, and addition/subtraction are $\Theta(n)$

$$1101 = (1100 + 01) = (11 \times 2^2 + 01)$$
$$1011 = (1000 + 11) = (10 \times 2^2 + 11)$$

$$\begin{array}{c|c}
11 & 01 \\
\hline
\times 10 & 11 \\
\end{array}$$

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

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

$n$ $n$-bit additions versus:

1. divide each factor (roughly) in half $\quad b = 2$
2. multiply the halves (recursively, if they’re too big) $\quad a = 4$
3. combine the products with shifts and adds $\quad d = 1$

$4 > 2^1$

what?!? back to $\Theta(n^2)$
Gauss’s trick

\[ xy = 2^n x_1 y_1 + 2^{n/2} x_1 y_1 + 2^{n/2} (x_1 - x_0)(y_0 - y_1) + x_0 y_0 + x_0 y_0 \]
Gauss’s payoff
lose one multiplication!

1. divide each factor (roughly) in half \( b = 2 \)
2. subtract the halves... \( d = 1 \)
3. multiply the difference and the halves Gauss-wise \( a = 3 \)
4. combine the products with shifts and adds \( d = 1 \)

\[ 3 > 2^1 \]

\[ \Theta(n^{\log_2 3}) \]

FFT
closest point pairs

see Wikipedia

\[ P = [(x_0, y_0), (x_1, y_1), ..., (x_n, y_n)] \]

brute force: \( \Theta(n^2) \)
before recursion, sort into \( P_x \) and \( P_y \): same points, ordered by \( x \), ordered by \( y \)
cost: \( n \lg n \)

\( L_x \)
\( L_y \)

\( R_x \)
\( R_y \)

\( \text{min}_x: \text{min pair distance on } L \)

\( \text{min}_y: \text{min pair distance on } R \)

\( \delta = \min(\text{min}_L, \text{min}_R) \)
b = 2
a = 2
d = ?? 1, it turns out!
an $n \lg n$ algorithm

$P$ is a set of points

1. Construct (sort) $P_x$ and $P_y$ before recursion, do it once: $n \lg n$
2. For each recursive call, construct ordered $L_x, L_y, R_x, R_y$ must be in linear time
3. Recursively find closest pairs $(l_0, l_1)$ and $(r_0, r_1)$, with minimum distance $\delta$ $a = 2$
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 7 places apart
6. Choose the minimum pair from $D_y$ versus $(l_0, l_1)$ and $(r_0, r_1)$. 
traverse Dy bottom to top...

\[ b = 2 \]
\[ a = 2 \]
\[ d = 1 \]

Theta(n log n)
def recBinSearch(x, A, b, e):
    if b == e:
        if x <= A[b]:
            return b
        else:
            return e + 1
    else:
        m = (b + e) // 2  # midpoint
        if x <= A[m]:
            return recBinSearch(x, A, b, m)
        else:
            return recBinSearch(x, A, m + 1, e)

A: list, nondecreasing, comparable elements
x: value to search for, must be comparable
b: beginning index of search
e: end index of search

return position p where x is, or should be inserted.

1. b <= p <= e + 1
2. b < p => A[p-1] < x
3. p < e + 1 => A[p] >= x
conditions, pre- and post-

- $x$ and elements of $A$ are comparable
- $e$ and $b$ are valid indices, $0 \leq b \leq e < \text{len}(A)$
- $A[b..e]$ is sorted non-decreasing

RecBinSearch($x, A, b, e$) terminates and returns index $p$

- $b \leq p \leq e + 1$
- $b < p \Rightarrow A[p - 1] < x$
- $p \leq e \Rightarrow x \leq A[p]$

(except for boundaries, returns $p$ so that $A[p - 1] < x \leq A[p]$)
precondition ⇒ termination and postcondition

Proof: induction on \( n = e - b + 1 \)

Base case, \( n = 1 \): Terminates because there are no loops or further calls, returns \( p = b = e \iff x \leq A[b = p] \) or 
\( p = b + 1 = e + 1 \iff x > A[b = p - 1] \), so postcondition satisfied. Notice that the choice forces if-and-only-if.

Induction step: Assume \( n > 1 \) and that the postcondition is satisfied for inputs of size \( 1 \leq k < n \) that satisfy the precondition, and the RecBinSearch terminates on such inputs. Call RecBinSearch\((A,x,b,e)\) when \( n = e - b + 1 > 1 \). Since \( b < e \) in this case, the test on line 1 fails, and line 7 executes.

Exercise: \( b \leq m < e \) in this case. There are two cases, according to whether \( x \leq A[m] \) or \( x > A[m] \).

recursive call’s postcondition becomes the IH
Case 1: $x \leq A[m]$

$0 \leq 0 + 1 \leq m - b + 1 < e - b + 1 = n$
# since $b \leq m < e$, by exercise

- Show that IH applies to $\text{RBS}(x,A,b,m)$
- Translate the postcondition to $\text{RBS}(x,A,b,m)$

These are now our I.H.
1. $b \leq p \leq m+1$          # by postcondition
2. $b < p \Rightarrow A[p-1] < x$
3. $p \leq m \Rightarrow A[p] \geq x$

- Show that $\text{RBS}(x,A,b,e)$ satisfies postcondition

1. $b \leq p$                  # by IH
   $p \leq m+1 \leq e+1$      # since $m < e$, by exercise
2. $b < p \Rightarrow A[p-1] < x$      # by IH
3. $p \leq e$                 # always true, since $p \leq m+1 \leq e$
   $\Rightarrow p = m+1$  $O\ p \leq m$  # first case NEVER happens
   $\Rightarrow A[p] \geq x$          # since $p = m+1 \Rightarrow A[p-1] = A[m] \Rightarrow A[m] < x$  (contradiction!)
   # by 3. in IH
Case 2: $x > A[m]$

must show that $1 \leq e - m < e - b + 1 = n$
# since $b \leq m < e$

- Show that IH applies to $RBS(x,A,m+1,e)$
- Translate postcondition to $RBS(x,A,m+1,e)$

terminates, and
1. $m+1 \leq p \leq e+1$
2. $m+1 < p \implies A[p-1] < x$
3. $p \leq e \implies A[p] \geq x$

- Show that $RBS(x,A,b,e)$

1. $p \leq e+1$
   # by IH
   $b \leq m+1 \leq p$
   # since $b \leq m$ (by exercise)
3. $p \leq e \implies A[p] \geq x$
   # by IH
2. $b < p$
   # always true, since $p \geq m+1 > b$
   either $p = m+1$ OR $p > m+1$
   case $p > m+1 \implies A[p-1] < x$
   # by 2. of IH
   case $p = m+1 \implies A[p-1] = A[m] < x$ (by assumption of this case)
what could possibly go wrong?

- \( m = \left\lceil \frac{e + b}{2.0} \right\rceil \)

- \( x < A[m] \)

- ... 

- Either prove correct, or find a counter-example