CSC236 fall 2018

divide and conquer
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...

Class of algorithms: partition problem into $b$ \emph{roughly} equal subproblems, solve, and recombine:

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

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?

1101
×1011

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

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

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

$n$ $n$-bit additions versus:

1. divide each factor (roughly) in half
2. multiply the halves (recursively, if they’re too big)
3. combine the products with shifts and adds
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
2. subtract the halves...
3. multiply the difference and the halves Gauss-wise
4. combine the products with shifts and adds
closest point pairs
see Wikipedia
divide-and-conquer v0.1
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 ordered $L_x, L_y, R_x, R_y$
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 7 places apart
6. Choose the minimum pair from $D_y$ versus $(l_0, l_1)$ and $(r_0, r_1)$. 
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)
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 $\Rightarrow$ termination and postcondition

Proof: induction on $n = e - b + 1$

Base case, $n = 1$: Terminates because there are no loops or further calls, returns $p = b = e \Leftrightarrow x \leq A[b = p]$ or $p = b + 1 = e + 1 \Leftrightarrow x > A[b = p - 1]$, 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 $x > A[m]$. 
Case 1: $x \leq A[m]$

- Show that IH applies to RBS(x,A,b,m)
- Translate the postcondition to RBS(x,A,b,m)

- Show that RBS(x,A,b,e) satisfies postcondition
Case 2: $x > A[m]$

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

- Show that $\text{RBS}(x, A, b, e)$
what could possibly go wrong?

- $m = \left[ \frac{e+b}{2.0} \right]$

- $x < A[m]$

- ...

- Either prove correct, or find a counter-example