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...

```
b: number of pieces you divide problem into
a: number of recursive calls
f: cost of splitting and combining, and hope  f \in \Theta(n^d)
```

Class of algorithms: partition problem into b *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 $a_1 + a_2 > 0$. $f(n)$ 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?

$$
\begin{array}{r}
1101 \\
\times 1011 \\
\hline
\end{array}
$$

```
   1101
  1101
 0000
1101
------------
 10001111
```

```
n rowise multiplication
n column-wise adds

\Theta(n^2)
```

```
32-bit or 64-bit
```

divide and recombine

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

```
x = (1100 + 01) = [(11)(2^2) + 01]
y = (1000 + 11) = [(10)(2^2) + 11]
```

11	01
×10	11

$$
\begin{aligned}
xy &= (2^{n/2}x_1 + x_0)(2^{n/2}y_1 + y_0) \\
&= 2^n x_1 y_1 + 2^{n/2}(x_1 y_0 + y_1 x_0) + x_0 y_0
\end{aligned}
$$

compare costs

n n-bit additions versus:

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

f \in \Theta(n^1)

4 > 2^1

we're still in \Theta(n^2)

Gauss's trick

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

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

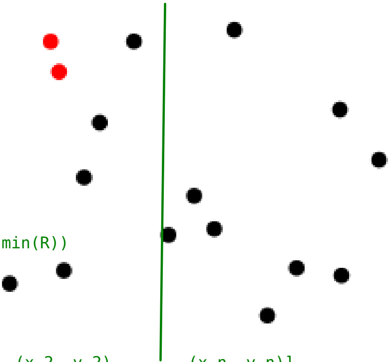
```
b = 2
a = 3
d = 1
```

```
3   >   2^1
```

```
now in \Theta(n^{log_2 3})
improve with FFT
```

closest point pairs

see Wikipedia



```
Brute force:
\in \Theta(C(n,2))
.... \Theta(n^2)

n(n-1)/2
```

```
min(min(L), min(R))
```

```
P = [(x_0, y_0), (x_1, y_1), (x_2, y_2), ...., (x_n, y_n)]
```

divide-and-conquer v0.1

$$T(n) = \begin{cases} k & \text{when } n \le 3 \\ T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + f(n) \sim n^d & \end{cases}$$

2   2^??

b = 2
a = 2
d = ??

after spark of insight, d = 1
and algorithm in \Theta(n lg n)

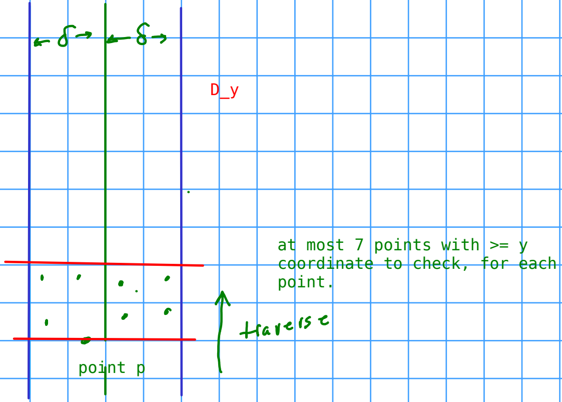
an $n \lg n$ algorithm

P is a set of points

1. Construct (sort) P_x and P_y do this once, before the recursive algo costs n lg n
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 δ
4. V is the vertical line splitting L and R, D is the δ-neighbourhood of V, and D_y is D ordered by y-ordinate
5. Traverse D_y looking for mininum pairs 7 places apart
6. Choose the minimum pair from D_y versus (l_0, l_1) and (r_0, r_1).

b = 2
a = 2
d = 1

2 = 2^1



δ
δ
D_y
at most 7 points with >= y coordinate to check, for each point.
traverse
point p

recursive binary search


```
A: nondecreasing array
b: beginning index
e: ending index
x: value to search for
n = len(A) = e-b+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)
```

```
recBinSearch find position p
useful for where to insert x
if not already there
```

```
1. 0 <= 0 <= e+1
2. b<p ==> A[p-1] < x
3. p <= e => A[p] >= x
```

conditions, pre- and post-

- x and elements of A are comparable
- e and b are valid indices, $0 \leq b \leq e < len(A)$
- $A[b..e]$ 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 1 <= m - b + 1 < n = e - b + 1

terminates, returns p

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

e

```
1. b <= p <= m+1
2. b < p => A[p-1] < x
3  p <= m => A[p] >= x
```

These are from Inductive Hypothesis

- Show that RBS(x,A,b,e) satisfies postcondition

```
1. b <= p <= m + 1 <= e+1   # by IH and m < e
2. b < p => A[p-1] < x
3. p <= e => p <= m+1, this breaks into p <= m  OR p = m+1
   so p <= m, by IH A[p] >= x
```

NEVER HAPP

```
if p = m+1, then p-1 = m
so by 2. A[m] < x
```

Case 2: $x > A[m]$

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

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

what could possibly go wrong?

- $m = \lceil \frac{e+b}{2.0} \rceil$
- $x < A[m]$
- $\ldots$
- Either prove correct, or find a counter-example

Notes