

CSE 421: Algorithms

Winter 2014

Lecture 11: Divide & Conquer

Reading:
Sections 5.1-5.4

INEFFECTIVE SORTS

<pre> DEFINE: HALFHEAVYMERGESORT(LIST): IF LENGTH(LIST) < 2: RETURN LIST PIVOT = INT(LENGTH(LIST) / 2) A = HALFHEAVYMERGESORT(LIST[:PIVOT]) B = HALFHEAVYMERGESORT(LIST[PIVOT:]) //UPHEAVY RETURN [A, B] // HERE, SORRY. </pre>	<pre> DEFINE: FETTERBOGOSORT(LIST): // AN O(N!) SORT // READS IN O(N!LOGN) FOR N FROM 1 TO LOG(LENGTH(LIST)): SHUFFLE(LIST) IF ISORTED(LIST): RETURN LIST RETURN "KERNEL PAGE FAULT (ORDER GOOD 2)" </pre>
<pre> DEFINE: JOSEPHUSQUICKSORT(LIST): OK SO YOU CHOOSE A PIVOT THEN DIVIDE THE LIST IN HALF FOR EACH HALF: CHECK TO SEE IF IT'S SORTED NO? WAKE IT UP! SORT IT COMPARE EACH ELEMENT TO THE PIVOT THE BIGGER ONES GO IN A NEW LIST THE SMALLER ONES GO INTO AN THE SECOND LIST FROM BEFORE HING ON, LET THE NAME OF THE LISTS THIS IS LIST A THE NEW ONE IS LIST B PUT THE BIG ONES INTO LIST B NOW TAKE THE SECOND LIST OK, IF USE LHM, ARE WHICH ONE WAS THE PIVOT IN? SORTED? ALL THIS IT'S SORT RECURSIVELY CALLS ITSELF UNTIL BOTH LISTS ARE EMPTY RIGHT? NOT EMPTY, BUT YOU KNOW WHAT I MEAN AM I ALLOWED TO USE THE STINKING LIBRARIES? </pre>	<pre> DEFINE: FINNALSORT(LIST): IF ISORTED(LIST): RETURN LIST FOR N FROM 1 TO 30000: PIVOT = RANDOM(LENGTH(LIST)) LIST = LIST[PIVOT:] + LIST[:PIVOT] IF ISORTED(LIST): RETURN LIST IF ISORTED(LIST): //THIS CAN'T BE HAPPENING RETURN LIST IF ISORTED(LIST): //COME ON COME ON RETURN LIST // OH YEEZ // YOU GONNA BE IN SO MUCH TROUBLE LIST = [] SYSTEM("RANDOM -n 45") SYSTEM("cat -n -e") SYSTEM("cat -n -e") SYSTEM("cat -n -e") SYSTEM("cat -n -e") //PROBABLY RETURN [1, 2, 3, 4, 5] </pre>

divide & conquer q algorithm

Power(a,n):

if n=0 then

return(1)

else if n=1 then

return(a)

else

x ← Power(a, [n/2])

if n is even then

return(x•x)

else

return(a•x•x)

fast exponentiation

• Power(a,n)

- Input: integer n and number a
- Output: aⁿ

• Obvious algorithm

- n-1 multiplications

• Observation:

- if n is even, n=2m, then aⁿ=a^m•a^m

analysis

• Worst-case recurrence

$$- T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + 2 \text{ for } n \geq 1$$

$$- T(1) = 0$$

• Time:

• More precise analysis:

$$T(n) = \lceil \log_2 n \rceil + \# \text{ of } 1\text{'s in } n\text{'s binary representation}$$

practical application: RSA

- Instead of a^n want $a^n \bmod N$
 - $a^{i+j} \bmod N = ((a^i \bmod N) \cdot (a^j \bmod N)) \bmod N$
 - same algorithm applies with each $x \cdot y$ replaced by $((x \bmod N) \cdot (y \bmod N)) \bmod N$
- In RSA cryptosystem (widely used for security)
 - need $a^n \bmod N$ where a, n, N each typically have **1024** bits
 - Power: at most **2048** multiplies of **1024** bit numbers relatively easy for modern machines
 - Naive algorithm: 2^{1024} multiplies

bisection method

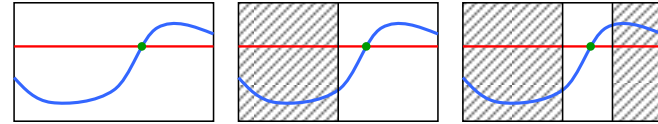
Bisection(a,b,ε):

```

if (a-b) < ε then
  return(a)
else
  c ← (a+b)/2
  if f(c) ≤ 0 then
    return(Bisection(c,b,ε))
  else
    return(Bisection(a,c,ε))

```

binary search for roots (bisection method)



- **Given:**
 - continuous function f and two points $a < b$ with $f(a) \leq 0$ and $f(b) > 0$
- **Find:**
 - approximation to c s.t. $f(c) = 0$ and $a < c < b$

analysis

- At each step we halved the size of the interval
- It started at size $b-a$
- It ended at size ϵ
- # of calls to f is $\log_2 \left(\frac{b-a}{\epsilon} \right)$

old favorites

- **Binary search**

- One subproblem of half size plus one comparison
- Recurrence $T(n) = T(\lceil n/2 \rceil) + 1$ for $n \geq 2$
 $T(1) = 0$
 So $T(n)$ is $\lceil \log_2 n \rceil + 1$

- **Mergesort**

- Two subproblems of half size plus merge cost of $n-1$ comparisons
- Recurrence $T(n) \leq 2T(\lceil n/2 \rceil) + n - 1$ for $n \geq 2$
 $T(1) = 0$
 Roughly n comparisons at each of $\log_2 n$ levels of recursion
 So $T(n)$ is roughly $2n \log_2 n$

closest pair in the plane



No single direction along which one can sort points to guarantee success!

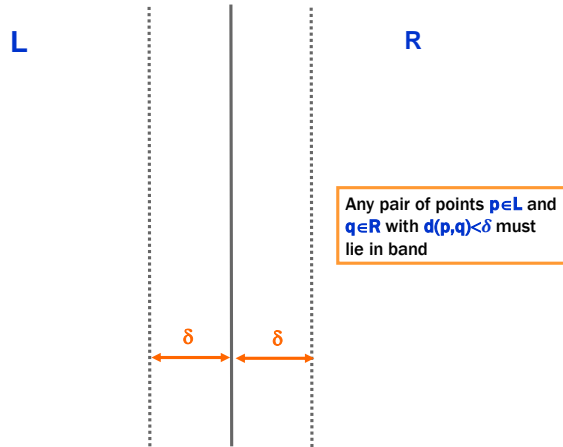
euclidean closest pair

- **Given** a set P of n points p_1, \dots, p_n with real-valued coordinates
- **Find** the pair of points $p_i, p_j \in P$ such that the Euclidean distance $d(p_i, p_j)$ is minimized
- $\Theta(n^2)$ possible pairs
- In one dimension?
- What about points in the plane?

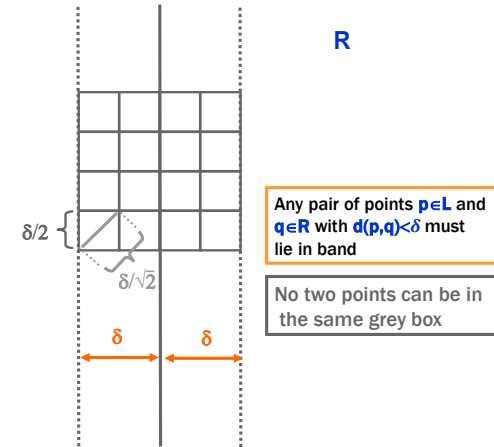
divide and conquer?

- Sort the points by their x coordinates
- Split the points into two sets of $n/2$ points L and R by x coordinate
- Recursively compute
 - closest pair of points in L , (p_L, q_L)
 - closest pair of points in R , (p_R, q_R)
- Let $\delta = \min\{d(p_L, q_L), d(p_R, q_R)\}$ and let (p, q) be the pair of points that has distance δ

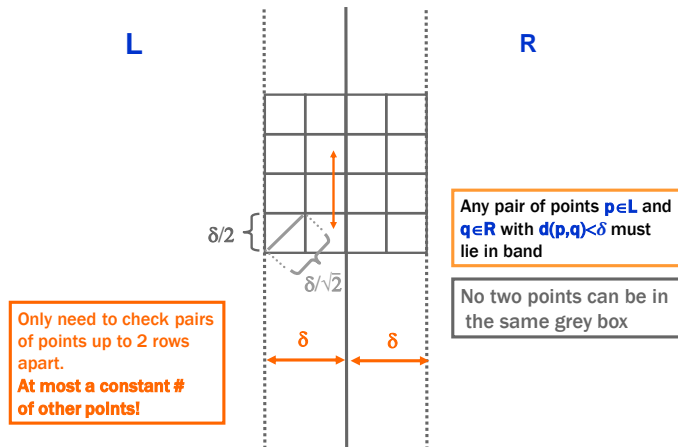
clever girl



clever girl



clever girl



closest pair recombining

- Sort points by y coordinate ahead of time
- On recombination only compare each point in δ -band of $L \cup R$ to the **11** points in δ -band of $L \cup R$ above it in the y sorted order
 - If any of those distances is better than δ replace (p,q) by the best of those pairs
- $O(n \log n)$ for x and y sorting at start
- Two recursive calls on problems on half size
- $O(n)$ recombination
- Total:

sometimes two sub-problems aren't enough

- More general divide and conquer
 - You've broken the problem into a different sub-problems
 - Each has size at most n/b
 - The cost of the break-up and recombining the sub-problem solutions is $O(n^k)$
- Recurrence: $T(n) \leq a \cdot T(n/b) + c \cdot n^k$

master divide and conquer recurrence

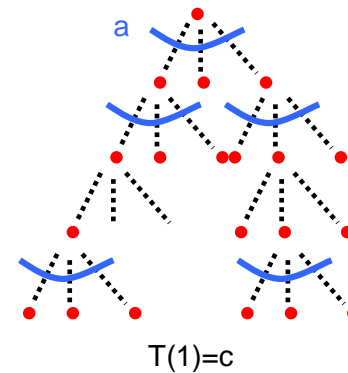
- If $T(n) \leq a \cdot T(n/b) + c \cdot n^k$ for $n > b$ then
 - if $a > b^k$ then $T(n)$ is $\Theta(n^{\log_b a})$
 - if $a < b^k$ then $T(n)$ is $\Theta(n^k)$
 - if $a = b^k$ then $T(n)$ is $\Theta(n^k \log n)$

master divide and conquer recurrence

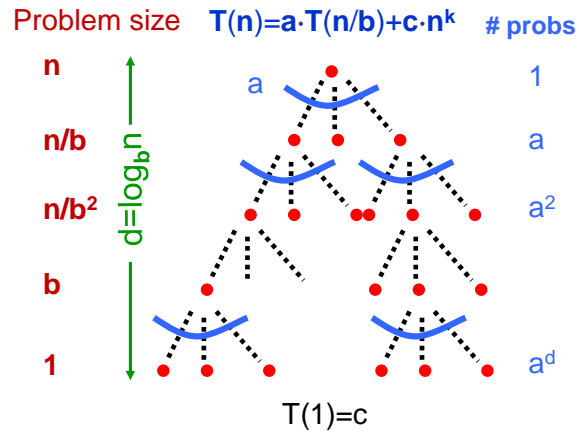
- If $T(n) \leq a \cdot T(n/b) + c \cdot n^k$ for $n > b$ then
 - if $a > b^k$ then $T(n)$ is $\Theta(n^{\log_b a})$
 - if $a < b^k$ then $T(n)$ is $\Theta(n^k)$
 - if $a = b^k$ then $T(n)$ is $\Theta(n^k \log n)$
- Works even if it is $\lceil \frac{n}{b} \rceil$ instead of $\frac{n}{b}$.

proving the master recurrence

Problem size $T(n) = a \cdot T(n/b) + c \cdot n^k$ # probs



proving the master recurrence



total cost

- **Geometric series**
 - ratio a/b^k
 - $d+1 = \log_b n + 1$ terms
 - first term cn^k , last term ca^d
- If $a/b^k = 1$
 - all terms are equal $T(n)$ is $\Theta(n^k \log n)$
- If $a/b^k < 1$
 - first term is largest $T(n)$ is $\Theta(n^k)$
- If $a/b^k > 1$
 - last term is largest $T(n)$ is
$$\Theta(a^d) = \Theta(a^{\log_b n}) = \Theta(n^{\log_b a})$$

geometric series

- $S = t + tr + tr^2 + \dots + tr^{n-1}$
- $r \cdot S = tr + tr^2 + \dots + tr^{n-1} + tr^n$
- $(r-1)S = tr^n - t$
- so $S = t(r^n - 1)/(r-1)$ if $r \neq 1$.
- **Simple rule**
 - If $r \neq 1$ then S is a constant times largest term in series

proving the master recurrence

