CS140: Algorithms

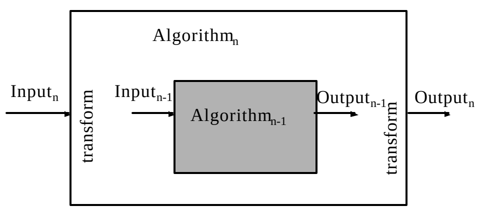
Z Sweedyk

Lecture 10

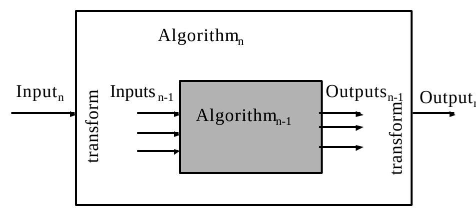
Algorithm Design Techniques

- Induction
- Reduction
- **Divide and Conquer**

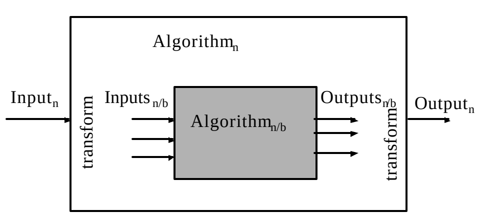
Induction: Algorithm_n



Induction: Algorithm_n



Divide and Conquer: Algorithm_n



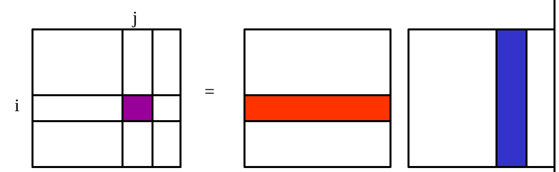
Sorting

- Insertion Sort: $T(n)=T(n-1)+cn=O(n^2)$
  - Sort first n-1 elements
  - Insert element n into sorted list
- Merge Sort: $T(n)=2T(n/b)+cn=O(n \lg(n))$
  - Sort first n/2 elements
  - Sort last n/2 elements
  - Merge sorted lists into one sorted list

Matrix Multiplication

- Input: Two n x n matrices A, B
- Output: The product AB

Reminder



Example

- Input:

-1	3
2	4

2	-3
2	1

- Output:

4	6
12	-2

Algorithm

```
For i = 1 to n
    For j = 1 to n
        C[i,j]=0
        For k = 1 to n
            C[i,j] = C[i,j] +  A[i,k]B[k,j]
```


Operation Count

```
For i = 1 to n
    For j = 1 to n
        C[i,j]=0
        For k = 1 to n
            C[i,j] = C[i,j] +  A[i,k]B[k,j]
```

Requires n^3 multiplications and $n^2(n-1)$ additions.

(Note the input size is $2n^2$.)

Divide and Conquer: $n=2^m$

- Input:

A_{11}	A_{12}
A_{21}	A_{22}

B_{11}	B_{12}
B_{21}	B_{22}

- Output:

C_{11}	C_{12}
C_{21}	C_{22}

$C_{ij} = A_{i1}B_{1j} + A_{i2}B_{2j}$

$T(2n^2)=8T(2(n/2)^2) + cn^2 = O(n^3)$

Can we do better?

- $T(2n^2)=aT(2(n/2)^2)+cn^2 = o(n^3)$ if $a<8$
- Strassen's algorithm:
  $a=7$ and $T(n)=O(n^{\log_2 7})$

Strassen's Algorithm
Base Case: 2x2

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \times \begin{pmatrix} e & g \\ f & h \end{pmatrix} = \begin{pmatrix} ae+bf & ag+bh \\ ce+df & cg+dh \end{pmatrix}$$

Is there hope?

Rephrase

Call this matrix A — Call this vector v

$$\begin{pmatrix} a & b & 0 & 0 \\ c & d & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & c & d \end{pmatrix} \begin{pmatrix} e \\ f \\ g \\ h \end{pmatrix} = \begin{pmatrix} ae+bf \\ ce+df \\ ag+bh \\ cg+dh \end{pmatrix}$$

A little magic …

We can compute Av with 7 scalar multiplications!

B and C

B

b	b	0	0
b	b	0	0
0	0	0	0
0	0	0	0

C

0	0	0	0
0	0	0	0
0	0	c	c
0	0	c	c

B+C

B and C

B

b	b	0	0
b	b	0	0
0	0	0	0
0	0	0	0

C

0	0	0	0
0	0	0	0
0	0	c	c
0	0	c	c

B+C

b	b	0	0
b	b	0	0
0	0	c	c
0	0	c	c

B, C and D

B+C

b	b	0	0
b	b	0	0
0	0	c	c
0	0	c	c

D

0	0	0	0
c-b	0	0	c-b
b-c	0	0	b-c
0	0	0	0

B+C+D

B, C and D

B+C

b	b	0	0
b	b	0	0
0	0	c	c
0	0	c	c

D

0	0	0	0
c-b	0	0	c-b
b-c	0	0	b-c
0	0	0	0

B+C+D

b	b	0	0
c	b	0	c-b
b-c	0	c	b
0	0	c	c

B, C, D and E

B+C+D

b	b	0	0
c	b	0	c-b
b-c	0	c	b
0	0	c	c

E

a-b	0	0	0
0	0	0	0
c-b	0	a-c	0
0	0	0	0

B+C+D+E

B, C, D and E

B+C+D

b	b	0	0
c	b	0	c-b
b-c	0	c	b
0	0	c	c

E

a-b	0	0	0
0	0	0	0
c-b	0	a-c	0
0	0	0	0

B+C+D+E

a	b	0	0
c	b	0	c-b
0	0	a	b
0	0	c	c

B, C, D, E and F

B+C+D+E

a	b	0	0
c	b	0	c-b
0	0	a	b
0	0	c	c

F

0	0	0	0
0	d-b	0	b-c
0	0	0	0
0	0	0	d-c

B+C+D+E+F

B, C, D, E and F

B+C+D+E

a	b	0	0
c	b	0	c-b
0	0	a	b
0	0	c	c

F

0	0	0	0
0	d-b	0	b-c
0	0	0	0
0	0	0	d-c

B+C+D+E+F

a	b	0	0
c	d	0	0
0	0	a	b
0	0	c	d

A=B+C+D+E+F

a	b	0	0	e		ae+bf
c	d	0	0	f		ce+df
					=	
0	0	a	b	g		ag+bh
0	0	c	d	h		cg+dh

10/17/01 CS140 Lec 10 25

A little magic…

Claim: We can compute the products Bv, Cv, Dv, Ev, and Fv with 7 scalar multiplications

So we can compute Av with 7 scalar multiplications

10/17/01 CS140 Lec 10 26

Bv: 1 multiplication

B v

b	b	0	0	e		
b	b	0	0	f	=	
0	0	0	0	g		
0	0	0	0	h		

10/17/01 CS140 Lec 10 27

Bv: 1 multiplication

B v

b	b	0	0	e		b(e+f)
b	b	0	0	f	=	b(e+f)
0	0	0	0	g		0
0	0	0	0	h		0

10/17/01 CS140 Lec 10 28

Cv: 1 multiplication

C v

0	0	0	0	e		
0	0	0	0	f	=	
0	0	c	c	g		
0	0	c	c	h		

10/17/01 CS140 Lec 10 29

Cv: 1 multiplication

C v

0	0	0	0	e		0
0	0	0	0	f	=	0
0	0	c	c	g		c(g+h)
0	0	c	c	h		c(g+h)

10/17/01 CS140 Lec 10 30

Dv: 1 multiplication

D

0	0	0	0
c-b	0	0	c-b
b-c	0	0	b-c
0	0	0	0

v

e
f
g
h

=

0
(c-b)(e+h)
(b-c)(e+h)
0

Ev: 2 multiplication

E

a-b	0	0	0
0	0	0	0
c-b	0	a-c	0
0	0	0	0

v

e
f
g
h

=

Ev: 2 multiplication

E

a-b	0	0	0
0	0	0	0
c-b	0	a-c	0
0	0	0	0

v

e
f
g
h

=

(a-b)e
0
(c-b)e+(a-c)g
0

Note: (c-b)e+(a-c)g =(a-c)(g-e)+(a-b)e

Fv: 2 multiplication

F

0	0	0	0
0	d-b	0	b-c
0	0	0	0
0	0	0	d-c

v

e
f
g
h

=

Fv: 2 multiplication

F

0	0	0	0
0	d-b	0	b-c
0	0	0	0
0	0	0	d-c

v

e
f
g
h

=

0
(d-b)f+(b-c)h
0
(d-c)h

Note: (d-b)(f-h)=(d-b)f+(b-c)h-(d-c)h

Strassen's Divide and Conquer

- Input: $\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$ $\begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$
- Output: $\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$

$C_{ij} = A_{i1}B_{1j} + A_{i2}B_{2j}$

$T(2n^2)=7T(2(n/2)^2) + cn^2 = O(n^{\log_3 7})$

Closest Pair

- Input: Set of points $n \geq 2$ on the plane $\{(x_i, y_i) \mid 1 \leq i \leq n\}$
- Output: Pair of closest points (under Euclidian distance)

Closest Pair Example

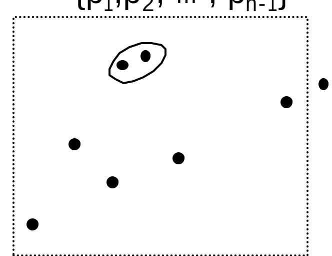
- Input: {(1,1),(2,2),(10,1),(10,2)}
- Output: (10,1) and (10,2)

Closest Pair – Algorithm Naïve Approach

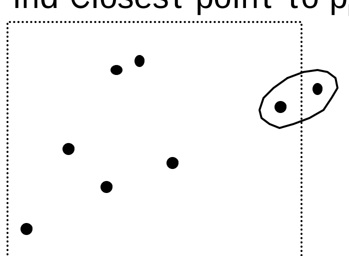
- Compute pair-wise distances
- Choose smallest

Requires $O(n^2)$ time

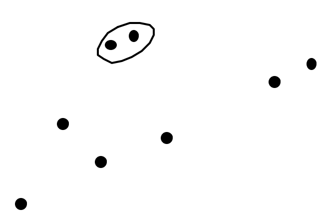
Simple Inductive Approach Find Closest Pair in $\{p_1, p_2, \ldots, p_{n-1}\}$



Simple Inductive Approach cont. Find Closest point to p_n



Simple Inductive Approach cont. Choose best of best

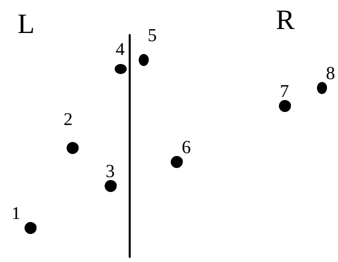


Simple Inductive Approach

- $T(n) = T(n-1) + O(n) = O(n^2)$

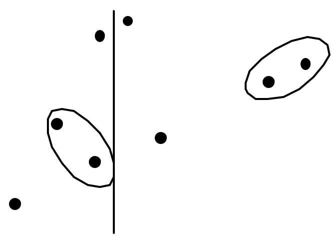
10/17/01 CS140 Lec 10 43

Divide and Conquer
Partition by x-coordinate



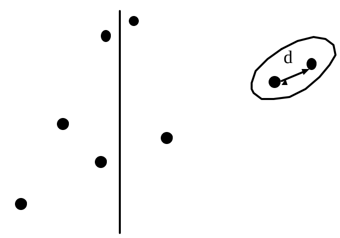
10/17/01 CS140 Lec 10 44

Divide and Conquer cont.
Solve sub-problems



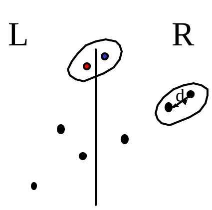
10/17/01 CS140 Lec 10 45

Divide and Conquer cont.
Choose best of best



10/17/01 CS140 Lec 10 46

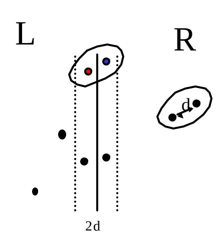
What have we missed?
Suppose p and q are closer than d.



1. Then p ε L and q ε R

10/17/01 CS140 Lec 10 47

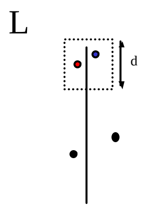
What have we missed?
Suppose p and q are closer than d.



1. Then p ε L and q ε R
2. Each is within d of the L-R boundary

10/17/01 CS140 Lec 10 48

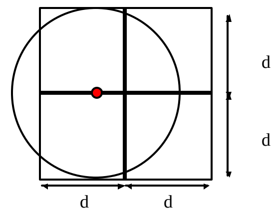
What have we missed? Suppose p and q are closer than d.



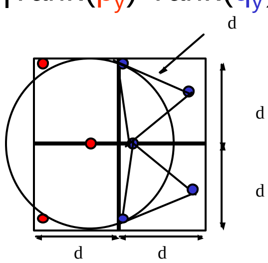
1. Then p ε L and q ε R
2. They are within d of the L-R boundary
3. Their y-coordinates differ by less than d

Objective: n lg n vs. n^2

- Sort (n lg n) then search (constant)



If d(p,q)<d then |rank(p_y)-rank(q_y)|<4



Closest-pair(P)

- Let M be the median p_x for p ε P
- P_1={p ε P | $p_x \leq$ M}, P_2={p ε P | p_x > M}
- Let (r,s) be the closest of the pairs Closest-pair(P1) and Closest-pair(P2)
- Let Q={p ε P | d(r,s)>|p_x-M|}
- Sort Q by y-coordinate.
- Find closest pair p_i and p_j ε Q, where 0<|i-j|<4.
- Return best pair found

Running time of Closest Pair

- T(n) = 2T(n/2) + n lg(n) = O(n lg(n))