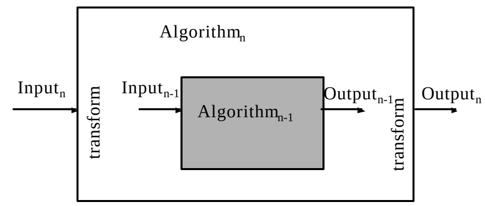
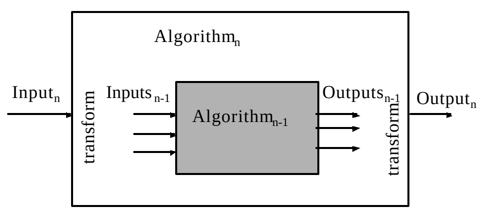
Algorithm Design Techniques

- Induction
- **Divide and Conquer**
- Dynamic Programming
- Greedy
- Reduction

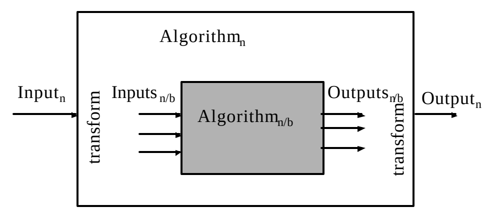
Induction: Algorithm_n



Induction: Algorithm_n



Divide and Conquer: Algorithm_n



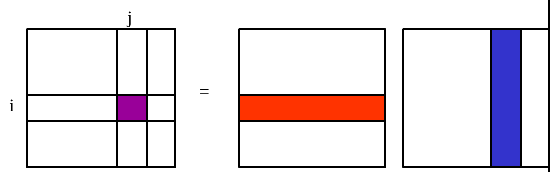
Sorting

- 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

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

Is there hope?

Rephrase

Call this matrix A — Call this vector v

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

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, 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, 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, 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+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

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

Bv: 1 multiplication

B

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

v

e
f
g
h

=

Cv: 1 multiplication
(2 so far)

C

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

v

e
f
g
h

=

Dv: 1 multiplication
(3 so far)

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

=

Ev: 2 multiplication

(5 so far)

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

=

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

Fv: 2 multiplication

(7total)

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

=

Strassen's Divide and Conquer

- Input: $\begin{matrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{matrix}$ $\begin{matrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{matrix}$
- Output: $\begin{matrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{matrix}$ $C_{ij} = A_{i1}B_{1j} + A_{i2}B_{2j}$

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