Recurrence and Asymptotic Complexity Analysis

Recurrence점화식

Recurrence

– A function represented by the same function(s) of smaller size(s)

~ related to divide-and-conquer algorithms

Examples

– $a_n = a_{n\text{-}1} + 2$

– $f(n) = n\, f(n-1)$

– $f(n) = f(n-1) + f(n-2)$

– $f(n) = f(n/2) + n$

Running Time of Mergesort

mergeSort(A[], p, r): ▷ Sort A[p … r]

if ($p < r$)

$q \leftarrow \lfloor (p + r)/2 \rfloor$ ------ ❶ ▷ center location of p and q

mergeSort(A, p, q) ----- ❷ ▷ sorting the former half

mergeSort(A, q+1, r) --- ❸ ▷ sorting the latter half

merge(A, p, q, r) ------- ❹ ▷ merge

merge(A[], p, q, r):

Merge two sorted arrays A[p … q] and A[q+1 … r]

to a sorted array A[p … r]

Recurrence of running time: $T(n) = 2T(n/2)$ + overhead

✓a mergesort of size n

= two mergesorts of size $n/2$ + overhead

Methods of Asymptotic Analyses

1. Iteration반복대치
   – Iterative substitution by smaller functions
2. Guess & Verification추정후증명
   – Guess the conclusion, then prove by mathematical induction
3. Master Theorem마스터정리
   – Determine the complexity when a function is in some particular forms

Assumption

1. For all T(n), n is a positive integer

2. All functions are monotonically nondecreasing
   - $T(n) \le T(m) \quad \forall n < m$

3. If needed, we can assume WLOG $n = a^k$
   for any polynomial asymptotic function

a: positive integer

1. Iteration

$$T(n) = T(n-1) + n$$
$$T(1) = 1$$

$$\begin{aligned}
T(n) &= T(n-1) + n \\
&= (T(n-2) + (n-1)) + n \\
&= (T(n-3) + (n-2)) + (n-1) + n \\
&\cdots \\
&= T(1) + 2 + 3 + \ldots + n \\
&= 1 + 2 + \ldots + n \\
&= n(n+1)/2 \\
&= \Theta(n^2)
\end{aligned}$$

Iteration

$$T(n) = 2T(n/2) + n$$
$$T(1) = 1$$

Assume $n = 2^k$

$$\begin{aligned} T(n) &= 2T(n/2) + n \\ &= 2(2T(n/2^2) + n/2) + n = 2^2T(n/2^2) + 2n \\ &= 2^2(2T(n/2^3) + n/2^2) + 2n = 2^3T(n/2^3) + 3n \\ &\dots \\ &= 2^kT(n/2^k) + kn \\ &= n + n\log n \\ &= \Theta(n\log n) \end{aligned}$$

Another Example of Iteration

$$\boxed{T(n) = n + 3T(\frac{n}{4})}$$

Assume $n = 4^k$

$$
\begin{aligned}
T(n) &= n + 3T(\frac{n}{4}) \\
&= n + 3(\frac{n}{4} + 3\mathrm{T}(\frac{n}{4^2})) = n + \frac{3}{4}n + 3^2\mathrm{T}(\frac{n}{4^2}) \\
&= n + \frac{3}{4}n + 3^2(\frac{n}{4^2} + 3\mathrm{T}(\frac{n}{4^3})) = n + \frac{3}{4}n + (\frac{3}{4})^2 n + 3^3\mathrm{T}(\frac{n}{4^3}) \\
&\cdots \\
&= n + \frac{3}{4}n + (\frac{3}{4})^2 n + \quad \cdots \quad + 3^{\log_4 n}\underbrace{\mathrm{T}(\overbrace{\frac{n}{4^{\log_4 n}}}^{1})}_{\Theta(1)} \\
&\le n\sum_{i=0}^{\infty}(\frac{3}{4})^i + n^{\log_4 3}\Theta(1) \\
&= 4n + o(n) \\
&= \Theta(n)
\end{aligned}
$$

Here, $o(n)$ is not a set but a function in $o(n)$

This is a set

$$T(n) = O(n)$$

2. Guess and Verify

$$T(n) = 2T\left(\frac{n}{2}\right) + n$$

Guess: $T(n) = O(n\log n)$, i.e., $T(n) \le cn\log n$

Assume $T(k) \le ck\log k \quad \forall k < n$

inductive substitution귀납적 대치

<Proof>

$$\begin{aligned} T(n) &= 2\underline{T(n/2)} + n \\ &\le 2\underline{c(n/2)\log(n/2)} + n \\ &= cn\log n - cn\log 2 + n \\ &= cn\log n + (-c\log 2 + 1)n \\ &\le cn\log n \end{aligned}$$

$\exists c$ satisfying this

Choose e.g., $c = 2$, $n_0 = 4$(moderately)

Reminder: $O(n\log n) = \{f(n) \mid \exists c > 0, n_0 \ge 0$ s.t. $\forall n \ge n_0, \mathbf{f}(n) \le cn\log n \}$

Another Example of Guess & Verify

$$\boxed{T(n) = 2T\left(\frac{n}{2} + 17\right) + n}$$

Guess: $T(n) = O(n \log n)$, 즉 $T(n) \le cn \log n$

<Proof>

Assume $T(k) \le ck \log k \quad \forall k < n$

$$\begin{aligned}
T(n) &= \underline{2T(\tfrac{n}{2} + 17)} + n \\
&\le \underline{2c(\tfrac{n}{2} + 17)\log(\tfrac{n}{2} + 17)} + n \quad \longleftarrow \quad \frac{n}{2} + 17 < n,\ 34 < n \\
&= c(n+34)\log(\tfrac{n}{2} + 17) + n \\
&\le c(n+34)\log\frac{3n}{4} + n \quad \longleftarrow \quad \frac{n}{2} + 17 \le \frac{3n}{4},\ 68 \le n \\
&= cn\log n + cn\log\frac{3}{4} + 34c\log\frac{3n}{4} + n \\
&= cn\log n + n\left(c\log\frac{3}{4} + 1\right) + 34c\log\frac{3n}{4} \\
&\le cn\log n \quad \text{for sufficiently large } n
\end{aligned}$$

≤ 0

< 0

Choose $c = 5$

The Constant *c* Should be Consistent!

앞에서

$$\cdots$$

$$= c(n+34)\log(\frac{n}{2}+17) + n$$

$$\leq c(n+34)\underline{\log n} + n$$

$$= cn\log n + 34c\log n + n$$

$$\cancel{\leq} \; dn\log n \quad (\mathrm{X})$$

New constant *d*

Counterintuitive Example of Guess & Verify

$$\boxed{T(n) = 2T\left(\frac{n}{2}\right) + 1}$$

Guess: $T(n) = O(n)$, i.e., $T(n) \le cn$

<Proof>

$$\begin{aligned} T(n) &= 2T(n/2) + 1 \\ &\le \underline{2c(n/2)} + \boxed{1} && \longleftarrow \text{Inductive substitution} \\ &= cn + 1 \\ &\cancel{\le} cn && \longleftarrow \text{Can't proceed anymore!} \end{aligned}$$

Though Counterintuitive...

$$\boxed{T(n) = 2T\left(\frac{n}{2}\right) + 1}$$

Guess: $T(n) \le cn - 2$ ← Why not $T(n) \le cn + 2$?

<Proof>

$$\begin{aligned} T(n) &= 2T(n/2) + 1 \\ &\le 2(\underline{c(n/2) - 2}) + 1 \\ &= cn - 3 \\ &\le cn - 2 \end{aligned}$$

← Inductive substitution

If We Follow Intuition…

$$T(n) = 2T\left(\frac{n}{2}\right) + 1$$

Guess: $T(n) \leq cn+2$

<Proof>

$$\begin{aligned} T(n) &= 2T(n/2) + 1 \\ &\leq 2(\underline{c(n/2) + 2}) + 1 && \leftarrow \text{Inductive substitution} \\ &= cn + 5 \\ &\not\leq cn + 2 && \leftarrow \text{Can't proceed anymore!} \end{aligned}$$

Usually It is Straightforward

to verify a claim for boundary cases

e.g. $T(n) = 10T(n/10) + n,\ T(1) = 1$

Guess $T(n) \le cn\log n \longleftarrow O(\,)$

$\ge cn\log n \longleftarrow \Omega(\,)$

$\rightarrow \mathrm{T}(10) = 10\mathrm{T}(1) + 10 = 20 \le c10\log 10$

$\ge c10\log 10$

The common practice usually doesn't explicitly prove the boundary cases in Guess & Verify

3. Master Theorem

Recurrences of the form $T(n) = aT\left(\frac{n}{b}\right) + f(n)$

can be directly determined by Master Theorem.

Backgraound

Given a recurrence,

$$T(1) = 1$$

$$T(n) = a\,T(\frac{n}{b}) + f(n) \text{ for } n > 1,$$

where

a, b are positive constants

$f(n) = O(g(n))$ for some polynomial ft $g(n)$

Assume $n = b^k$

$$
\begin{aligned}
T(n) &= f(n) + aT(\tfrac{n}{b}) \\
&= f(n) + a(f(\tfrac{n}{b}) + aT(\tfrac{n}{b^2})) \\
&= f(n) + a(f(\tfrac{n}{b}) + a(f(\tfrac{n}{b^2}) + aT(\tfrac{n}{b^3}))) \\
&\cdots \\
&= \textstyle\sum_{i=0}^{k-1} a^i f(\frac{n}{b^i}) + a^k \overbrace{T(\tfrac{n}{b^k})}^{1} \\
&= \textstyle\sum_{i=0}^{k-1} a^i f(\frac{n}{b^i}) + n^{\log_b a}
\end{aligned}
$$

$a^k = a^{\log_b n} = n^{\log_b a}$

- Particular solution
- Cost of all overheads

- Homogeneous solution
- Cost of solving the boundary subproblems of size 1
- Criterion for time complexity

Intuitive Understanding of Master Thm

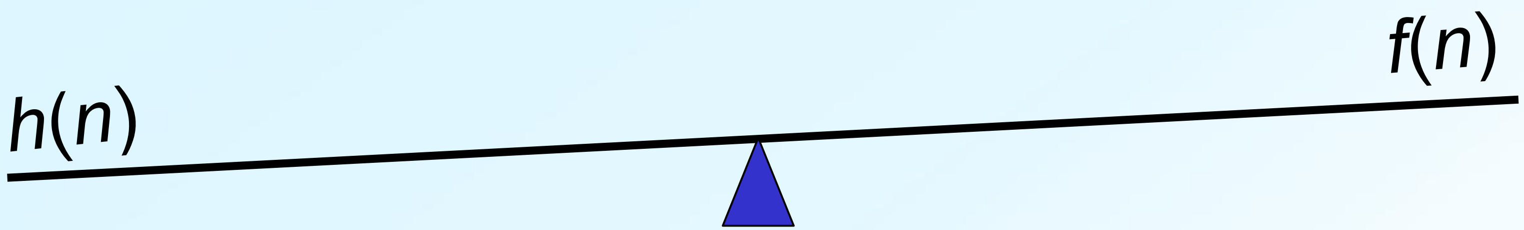
$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

Let $n^{\log_b a} = h(n)$

① $h(n)$ is heavier → $h(n)$ determines the running time

② $f(n)$ is heavier → $f(n)$ determines the running time

③ $h(n)$ and $f(n)$ draw → multiply $h(n)$ by $\log n$



Master Theorem

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

Let $n^{\log_b a} = h(n)$

1. $\frac{f(n)}{h(n)} = O(\frac{1}{n^{\varepsilon}})$ for some positive constant ε

   $\rightarrow T(n) = \Theta(h(n))$

2. $\frac{f(n)}{h(n)} = \Omega(n^{\varepsilon})$ for some positive constant ε

   and $af\left(\frac{n}{b}\right) < f(n)$ for all large enough n

   $\rightarrow T(n) = \Theta(f(n))$

3. $\frac{f(n)}{h(n)} = \Theta(1)$

   $\rightarrow T(n) = \Theta(h(n) \log n)$

Examples of Using Master Thm

- $T(n) = 2T(\frac{n}{3}) + c$
  - a=2, b=3, $h(n) = n^{\log_3 2}$, $f(n) = c$
  - $T(n) = \Theta(h(n)) = \Theta(n^{\log_3 2})$
- $T(n) = 2T(\frac{n}{4}) + n$
  - a=2, b=4, $h(n) = n^{\log_4 2}$, $f(n) = n$ and $2f\left(\frac{n}{4}\right) = \frac{n}{2} < n = f(n)$
  - $T(n) = \Theta(f(n)) = \Theta(n)$
- $T(n) = 2T(\frac{n}{2}) + n$
  - a=2, b=2, $h(n) = n^{\log_2 2} = n$, $f(n) = n$
  - $T(n) = \Theta(h(n)\log n) = \Theta(n \log n)$

Complexity of Matrix Multiplication

For interest

$$\underset{n*n}{\mathbf{A}}\cdot\underset{n*n}{\mathbf{B}}$$

✓ Running time: $\Theta(n^3)$

$$\mathbf{A}\cdot\mathbf{B} = \begin{pmatrix} \mathbf{A_1} & \mathbf{A_2} \\ \mathbf{A_3} & \mathbf{A_4} \end{pmatrix} \begin{pmatrix} \mathbf{B_1} & \mathbf{B_3} \\ \mathbf{B_2} & \mathbf{B_4} \end{pmatrix} = \begin{pmatrix} \mathbf{A_1B_1+A_2B_2} & \mathbf{A_1B_3+A_2B_4} \\ \mathbf{A_3B_1+A_4B_2} & \mathbf{A_3B_3+A_4B_4} \end{pmatrix}$$

Asymptotically, $T(n) = 8T(n/2) + \Theta(n^2)$, $T(1) = \Theta(1)$

✓ Running time: still $\Theta(n^3)$ by Master Thm

Strassen Algorithm

- Strassen, a young German mathematician, devised a clever method (1968)

$$\mathbf{A \cdot B} = \begin{pmatrix} A_1 & A_2 \\ A_3 & A_4 \end{pmatrix} \begin{pmatrix} B_1 & B_3 \\ B_2 & B_4 \end{pmatrix} = \begin{pmatrix} -P_2+P_4+P_5+P_6 & P_1+P_2 \\ P_3+P_4 & P_1-P_3+P_5-P_7 \end{pmatrix}$$

$$\| $$

$$\begin{pmatrix} A_1B_1+A_2B_2 & A_1B_3+A_2B_4 \\ A_3B_1+A_4B_2 & A_3B_3+A_4B_4 \end{pmatrix}$$

Instead

$P_1 = A_1(B_3-B_4)$

$P_2 = (A_1+A_2)B_4$

$P_3 = (A_3+A_4)B_1$

$P_4 = A_4(-B_1+B_2)$

$P_5 = (A_1+A_4)(B_1+B_4)$

$P_6 = (A_2-A_4)(B_2+B_4)$

$P_7 = (A_1-A_3)(B_1+B_3)$

Asymptotically, $T(n) = 7T(n/2) + \Theta(n^2)$, $T(1) = \Theta(1)$

✓ Running time: $\Theta(n^{\log_2 7}) = \Theta(n^{2.81})$

- After Strassen, many, many improvements … down to $\Theta(n^{2.376})$
- But, … proved that
  - Strassen algorithm is optimal

    in bilinear combination of $n/2$ * $n/2$ matrices

- We were curious
  - ✓ How many algorithms other than Strassen's algorithm exist?
  - ✓ Can a search algorithm achieve the efficiency of finding the same or equivalent algorithms?

We Found at Least 608 Such Algorithms

Representative Solutions in Each Group

Group 1 (Strassen's Solution)	Group 2	Group 3
$P_1 = A_1(B_3 - B_4)$	$P_1 = A_1(B_3 - B_4)$	$P_1 = A_1(B_3 - B_4)$
$P_2 = (A_1 + A_2)B_4$	$P_2 = (A_1 + A_2)B_4$	$P_2 = (A_1 + A_2)B_4$
$P_3 = (A_3 + A_4)B_1$	$P_3 = A_4(B_2 + B_4)$	$P_3 = (A_3 - A_4)B_2$
$P_4 = A_4(-B_1 + B_2)$	$P_4 = A_3(B_1 + B_3)$	$P_4 = A_3(B_1 + B_2)$
$P_5 = (A_1 + A_4)(B_1 + B_4)$	$P_5 = (A_2 + A_4)(B_1 - B_2)$	$P_5 = (A_1 + A_2 + A_3 + A_4)(B_1 + B_2 + B_3 + B_4)$
$P_6 = (A_2 - A_4)(B_2 + B_4)$	$P_6 = (A_1 + A_2 + A_4)(B_1 + B_4)$	$P_6 = (A_1 + A_2 - A_3 + A_4)(B_1 + B_2 + B_3 - B_4)$
$P_7 = (-A_1 + A_3)(B_1 + B_3)$	$P_7 = (A_1 + A_2 + A_3 + A_4)B_1$	$P_7 = (A_1 - A_2 + A_3 - A_4)(B_1 - B_2 + B_3 - B_4)$
$C_1 = -P_2 + P_4 + P_5 + P_6$	$C_1 = -P_2 - P_3 - P_5 + P_6$	$C_1 = -P_1 - P_3 + 0.5P_6 + 0.5P_7$
$C_2 = P_1 + P_2$	$C_2 = P_1 + P_2$	$C_2 = P_1 + P_2$
$C_3 = P_3 + P_4$	$C_3 = P_2 + P_3 - P_6 + P_7$	$C_3 = -P_3 + P_4$
$C_4 = P_1 - P_3 + P_5 + P_7$	$C_4 = -P_2 + P_4 + P_6 - P_7$	$C_4 = -P_2 - P_4 + 0.5P_5 - 0.5P_6$
Group 4	Group 5	Group 6
$P_1 = A_4(-B_1 + B_2 - B_3 + B_4)$	$P_1 = (A_1 + A_2)(B_1 + B_3)$	$P_1 = (A_1 + A_2)(B_3 + B_4)$
$P_2 = A_1(B_1 - B_2 - B_3 + B_4)$	$P_2 = (A_1 + A_2 - A_3 + A_4)(B_2 - B_3)$	$P_2 = (A_1 - A_2)(B_3 - B_4)$
$P_3 = (A_1 + A_4)(B_1 - B_2 + B_3 + B_4)$	$P_3 = (-A_3 + A_4)(B_1 - B_3)$	$P_3 = (A_2 - A_4)(B_1 - B_2 - B_3 + B_4)$
$P_4 = (A_1 - A_3)B_3$	$P_4 = (A_1 + A_2 - A_3 - A_4)(B_1 + B_2)$	$P_4 = (A_2 + A_4)(B_1 + B_2 + B_3 + B_4)$
$P_5 = (A_3 + A_4)(B_1 + B_3)$	$P_5 = (A_1 - A_2 - A_3 + A_4)(B_1 - B_2)$	$P_5 = (A_1 - A_4)(B_1 + B_4)$
$P_6 = (A_1 + A_2)(B_2 - B_4)$	$P_6 = A_2(B_1 - B_2 + B_3 - B_4)$	$P_6 = (A_1 + A_2 - A_3 - A_4)(B_1 - B_3)$
$P_7 = (A_2 - A_4)B_4$	$P_7 = A_4(B_1 + B_2 - B_3 - B_4)$	$P_7 = (A_1 - A_2 + A_3 - A_4)(B_1 + B_3)$
$C_1 = 0.5P_1 + 0.5P_2 + 0.5P_3 + P_6 + P_7$	$C_1 = 0.5P_1 + 0.5P_2 - 0.5P_3 + 0.5P_5$	$C_1 = -0.5P_1 + 0.5P_2 - 0.5P_3 + 0.5P_4 + P_5$
$C_2 = 0.5P_1 - 0.5P_2 + 0.5P_3 + P_7$	$C_2 = 0.5P_1 - 0.5P_2 + 0.5P_3 - 0.5P_5 - P_6$	$C_2 = 0.5P_1 + 0.5P_2$
$C_3 = 0.5P_1 + 0.5P_2 - 0.5P_3 + P_4 + P_5$	$C_3 = 0.5P_1 + 0.5P_2 - 0.5P_3 - 0.5P_4$	$C_3 = -0.5P_1 - 0.5P_2 + 0.5P_3 + 0.5P_4 - 0.5P_6 + 0.5P_7$
$C_4 = 0.5P_1 - 0.5P_2 + 0.5P_3 - P_4$	$C_4 = 0.5P_1 + 0.5P_2 + 0.5P_3 - 0.5P_4 - P_7$	$C_4 = 0.5P_1 - 0.5P_2 - P_5 + 0.5P_6 + 0.5P_7$
Group 7	Group 8	Group 9 (Winograd's Solution)
$P_1 = A_1B_1$	$P_1 = A_1B_1$	$P_1 = A_1B_1$
$P_2 = A_2B_2$	$P_2 = A_2B_2$	$P_2 = A_2B_2$
$P_3 = A_3(B_1 + B_2 + B_3 + B_4)$	$P_3 = A_3(B_3 + B_4)$	$P_3 = A_3(B_1 + B_2 + B_3 + B_4)$
$P_4 = (A_2 + A_4)(B_1 + B_2 - B_3 - B_4)$	$P_4 = (A_2 + A_4)(B_1 + B_2 + B_3 + B_4)$	$P_4 = (A_2 + A_4)(B_3 + B_4)$
$P_5 = (A_3 - A_4)(B_2 + B_4)$	$P_5 = (A_3 - A_4)B_4$	$P_5 = (A_3 - A_4)(B_2 + B_4)$
$P_6 = (A_2 - A_3 + A_4)(B_1 - B_2 - B_3 - B_4)$	$P_6 = (A_2 - A_3 + A_4)(B_1 + B_3 + B_4)$	$P_6 = (A_2 - A_3 + A_4)(B_2 - B_2 - B_3 - B_4)$
$P_7 = (A_1 - A_2 + A_3 - A_4)(B_1 - B_4)$	$P_7 = (A_1 - A_2 + A_3 - A_4)(B_1 + B_3)$	$P_7 = (A_1 - A_2 + A_3 - A_4)B_3$
$C_1 = P_1 + P_2$	$C_1 = P_1 + P_2$	$C_1 = P_1 + P_2$
$C_2 = P_1 - P_2 + P_3 - P_6 - P_7$	$C_2 = -P_1 + P_5 + P_6 + P_7$	$C_2 = -P_2 + P_5 + P_6 + P_7$
$C_3 = -P_2 + 0.5P_3 + 0.5P_4 + 0.5P_6$	$C_3 = -P_2 - P_3 + P_4 - P_6$	$C_3 = -P_2 + P_3 - P_4 + P_6$
$C_4 = P_2 + 0.5P_3 - 0.5P_4 - P_5 + 0.5P_6$	$C_4 = P_3 - P_5$	$C_4 = P_2 + P_4 - P_5 - P_6$

Group 1 (Strassen's Solution)	
$P_1 = A_1(B_3 - B_4)$	3
$P_2 = (A_1 + A_2)B_4$	3
$P_3 = (A_3 + A_4)B_1$	3
$P_4 = A_4(-B_1 + B_2)$	3
$P_5 = (A_1 + A_4)(B_1 + B_4)$	4
$P_6 = (A_2 - A_4)(B_2 + B_4)$	4
$P_7 = (-A_1 + A_3)(B_1 + B_3)$	4
$C_1 = -P_2 + P_4 + P_5 + P_6$	
$C_2 = P_1 + P_2$	
$C_3 = P_3 + P_4$	
$C_4 = P_1 - P_3 + P_5 + P_7$	

Group 2	
$P_1 = A_1(B_3 - B_4)$	3
$P_2 = (A_1 + A_2)B_4$	3
$P_3 = A_4(B_2 + B_4)$	3
$P_4 = A_3(B_1 + B_3)$	3
$P_5 = (A_2 + A_4)(B_1 - B_2)$	4
$P_6 = (A_1 + A_2 + A_4)(B_1 + B_4)$	5
$P_7 = (A_1 + A_2 + A_3 + A_4)B_1$	5
$C_1 = -P_2 - P_3 - P_5 + P_6$	
$C_2 = P_1 + P_2$	
$C_3 = P_2 + P_3 - P_6 + P_7$	
$C_4 = -P_2 + P_4 + P_6 - P_7$	

Group 3	
$P_1 = A_1(B_3 - B_4)$	3
$P_2 = (A_1 + A_2)B_4$	3
$P_3 = (A_3 - A_4)B_2$	3
$P_4 = A_3(B_1 + B_2)$	3
$P_5 = (A_1 + A_2 + A_3 + A_4)(B_1 + B_2 + B_3 + B_4)$	8
$P_6 = (A_1 + A_2 - A_3 + A_4)(B_1 + B_2 + B_3 - B_4)$	8
$P_7 = (A_1 - A_2 + A_3 - A_4)(B_1 - B_2 + B_3 - B_4)$	8
$C_1 = -P_1 - P_3 + 0.5P_6 + 0.5P_7$	
$C_2 = P_1 + P_2$	
$C_3 = -P_3 + P_4$	
$C_4 = -P_2 - P_4 + 0.5P_5 - 0.5P_6$	

Group 4	
$P_1 = A_4(-B_1 + B_2 - B_3 + B_4)$	5
$P_2 = A_1(B_1 - B_2 - B_3 + B_4)$	5
$P_3 = (A_1 + A_4)(B_1 - B_2 + B_3 + B_4)$	6
$P_4 = (A_1 - A_3)B_3$	3
$P_5 = (A_3 + A_4)(B_1 + B_3)$	4
$P_6 = (A_1 + A_2)(B_2 - B_4)$	4
$P_7 = (A_2 - A_4)B_4$	3
$C_1 = 0.5P_1 + 0.5P_2 + 0.5P_3 + P_6 + P_7$	
$C_2 = 0.5P_1 - 0.5P_2 + 0.5P_3 + P_7$	
$C_3 = 0.5P_1 + 0.5P_2 - 0.5P_3 + P_4 + P_5$	
$C_4 = 0.5P_1 - 0.5P_2 + 0.5P_3 - P_4$	

Group 5
$P_1 = (A_1 + A_2)(B_1 + B_3)$
$P_2 = (A_1 + A_2 - A_3 + A_4)(B_2 - B_3)$
$P_3 = (-A_3 + A_4)(B_1 - B_3)$
$P_4 = (A_1 + A_2 - A_3 - A_4)(B_1 + B_2)$
$P_5 = (A_1 - A_2 - A_3 + A_4)(B_1 - B_2)$
$P_6 = A_2(B_1 - B_2 + B_3 - B_4)$
$P_7 = A_4(B_1 + B_2 - B_3 - B_4)$
$C_1 = 0.5P_1 + 0.5P_2 - 0.5P_3 + 0.5P_5$
$C_2 = 0.5P_1 - 0.5P_2 + 0.5P_3 - 0.5P_5 - P_6$
$C_3 = 0.5P_1 + 0.5P_2 - 0.5P_3 - 0.5P_4$
$C_4 = 0.5P_1 + 0.5P_2 + 0.5P_3 - 0.5P_4 - P_7$

Group 6
$P_1 = (A_1 + A_2)(B_3 + B_4)$
$P_2 = (A_1 - A_2)(B_3 - B_4)$
$P_3 = (A_2 - A_4)(B_1 - B_2 - B_3 + B_4)$
$P_4 = (A_2 + A_4)(B_1 + B_2 + B_3 + B_4)$
$P_5 = (A_1 - A_4)(B_1 + B_4)$
$P_6 = (A_1 + A_2 - A_3 - A_4)(B_1 - B_3)$
$P_7 = (A_1 - A_2 + A_3 - A_4)(B_1 + B_3)$
$C_1 = -0.5P_1 + 0.5P_2 - 0.5P_3 + 0.5P_4 + P_5$
$C_2 = 0.5P_1 + 0.5P_2$
$C_3 = -0.5P_1 - 0.5P_2 + 0.5P_3 + 0.5P_4 - 0.5P_6 + 0.5P_7$
$C_4 = 0.5P_1 - 0.5P_2 - P_5 + 0.5P_6 + 0.5P_7$

Group 7
$P_1 = A_1 B_1$
$P_2 = A_2 B_2$
$P_3 = A_3(B_1 + B_2 + B_3 + B_4)$
$P_4 = (A_2 + A_4)(B_1 + B_2 - B_3 - B_4)$
$P_5 = (A_3 - A_4)(B_2 + B_4)$
$P_6 = (A_2 - A_3 + A_4)(B_1 - B_2 - B_3 - B_4)$
$P_7 = (A_1 - A_2 + A_3 - A_4)(B_1 - B_4)$
$C_1 = P_1 + P_2$
$C_2 = P_1 - P_2 + P_3 - P_6 - P_7$
$C_3 = -P_2 + 0.5P_3 + 0.5P_4 + 0.5P_6$
$C_4 = P_2 + 0.5P_3 - 0.5P_4 - P_5 + 0.5P_6$

Group 8
$P_1 = A_1 B_1$
$P_2 = A_2 B_2$
$P_3 = A_3(B_3 + B_4)$
$P_4 = (A_2 + A_4)(B_1 + B_2 + B_3 + B_4)$
$P_5 = (A_3 - A_4)B_4$
$P_6 = (A_2 - A_3 + A_4)(B_1 + B_3 + B_4)$
$P_7 = (A_1 - A_2 + A_3 - A_4)(B_1 + B_3)$
$C_1 = P_1 + P_2$
$C_2 = -P_1 + P_5 + P_6 + P_7$
$C_3 = -P_2 - P_3 + P_4 - P_6$
$C_4 = P_3 - P_5$

Group 9 (Winograd's Solution)
$P_1 = A_1 B_1$
$P_2 = A_2 B_2$
$P_3 = A_3(B_1 + B_2 + B_3 + B_4)$
$P_4 = (A_2 + A_4)(B_3 + B_4)$
$P_5 = (A_3 - A_4)(B_2 + B_4)$
$P_6 = (A_2 - A_3 + A_4)(B_2 - B_2 - B_3 - B_4)$
$P_7 = (A_1 - A_2 + A_3 - A_4)B_3$
$C_1 = P_1 + P_2$
$C_2 = -P_2 + P_5 + P_6 + P_7$
$C_3 = -P_2 + P_3 - P_4 + P_6$
$C_4 = P_2 + P_4 - P_5 - P_6$