◉ 11-C 計算 Linear Convolution

We know that when $y[n] = x[n] * h[n] = \sum_k x[n-k]h[k]$

then $y[n] = IFFT\left(FFT\{x[n]\}\,FFT\{h[n]\}\right)$ $\boxed{P \geq M + N - 1}$

(*N* points) (*M* points)

(*P* points)

But how do we implement it correctly?

How do we choose *P* ?

Note: When $y_1[n] = IFFT_P\left(FFT_P\{x_1[n]\}\,FFT_P\{h_1[n]\}\right)$

then $y_1[n] = \sum_{k=0}^{P-1} x_1[((n-k))_P]h_1[k]$

FFT_P: the *P*-point FFT

$IFFT_P$: the *P*-point inverse FFT

$((a))_P$: a 除以 P 的餘數

$$y[n] = x[n] * h[n] = \sum_k x[n-k]h[k]$$

Convolution 有幾種 Cases

Case 1: Both $x[n]$ and $h[n]$ have infinite lengths.

Case 2: Both $x[n]$ and $h[n]$ have finite lengths.

Case 3: $x[n]$ has infinite length but $h[n]$ has finite length.

Case 4: $x[n]$ has finite length but $h[n]$ has infinite length.

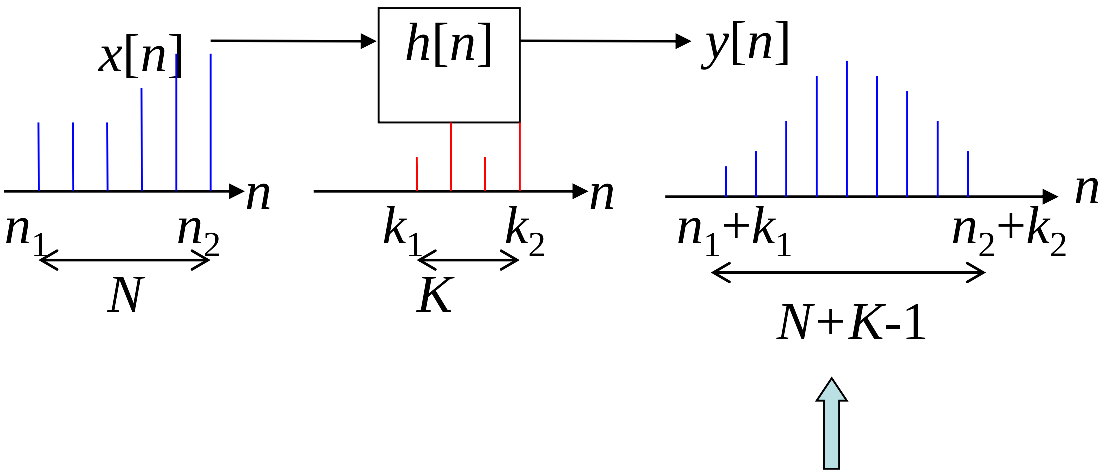
Case 2: Both $x[n]$ and $h[n]$ have finite lengths.

$x[n]$ 的範圍為 $n \in [n_1, n_2]$，大小為 $N = n_2 - n_1 + 1$

$h[n]$ 的範圍為 $n \in [k_1, k_2]$，大小為 $K = k_2 - k_1 + 1$

$$y[n] = x[n] * h[n] = \sum_{k=k_1}^{k_2} x[n-k]h[k]$$

$y[n]$ 的範圍?



Convolution output 的範圍以及點數，
是學信號處理的人必需了解的常識

$$y[n] = x[n] * h[n] = \sum_{k=k_1}^{k_2} x[n-k]h[k]$$

$x[n]$ 的範圍為 $n \in [n_1, n_2]$，範圍大小為 $N = n_2 - n_1 + 1$

$h[n]$ 的範圍為 $n \in [k_1, k_2]$，範圍大小為 $K = k_2 - k_1 + 1$

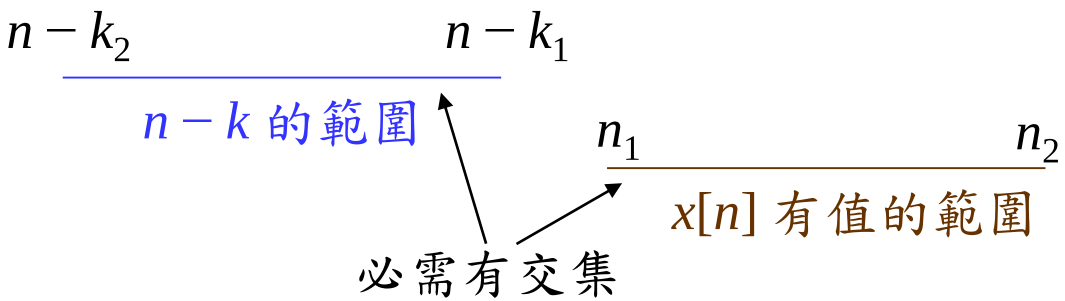
當 n 固定時

$$y[n] = x[n-k_1]h[k_1] + x[n-k_1-1]h[k_1+1] + x[n-k_1-2]h[k_1+2] + \cdots\cdots + x[n-k_2]h[k_2]$$

什麼情況下$y[n]$有值？

$x[n-k_1]$, $x[n-k_1-1]$, $x[n-k_1-2]$, $\cdots\cdots$, $x[n-k_2]$

其中至少有一個落在 $[n_1, n_2]$ 的範圍內



n 的下限為 $n-k_1$ 與 n_1 相重合

$$n-k_1=n_1, \quad n=k_1+n_1$$

n 的上限為 $n-k_2$ 與 n_2 相重合

$$n-k_2=n_2, \quad n=k_2+n_2$$

所以 $y[n]$ 的範圍是 $n\in[k_1+n_1\,,\,k_2+n_2]$

範圍大小為 $k_2+n_2-k_1-n_1+1=N+K-1$

FFT implementation for Case 2

$x_1[n] = x[n+n_1]$ for $n = 0, 1, 2, \ldots, N-1$

$x_1[n] = 0$ for $n = N, N+1, \ldots, P-1$ $P \geq N + K - 1$

$h_1[n] = h[n+k_1]$ for $n = 0, 1, 2, \ldots, K-1$

$h_1[n] = 0$ for $n = K, K+1, \ldots, P-1$

$$y_1[n] = IFFT_P\left(FFT_P\{x_1[n]\}FFT_P\{h_1[n]\}\right)$$

$y[n] = y_1[n-n_1-k_1]$ for $n = n_1+k_1, n_1+k_1+1, n_1+k_1+2, \ldots\ldots, k_2+n_2$

i.e., $n-n_1-k_1 = 0, 1, \cdots\cdots, N+K-2$

取 output 的前面 N+K–1 個點

證明：
$$y_1[n'] = \sum_{k'=0}^{P-1} x_1[((n'-k'))_P]h_1[k'] \qquad \text{(from page 384)}$$
$$= \sum_{k'=0}^{K-1} x_1[((n'-k'))_P]h_1[k'] \qquad (h_1[k'] = 0 \text{ for } k' \geq K)$$

可以簡化為 $y_1[n'] = \sum_{k'=0}^{K-1} x_1[n'-k']h_1[k']$

這是因為只有當 $n'-k'<0$ 時 $n'-k' \neq ((n'-k'))_P$

由於 $\min(n'-k') = \min(n') - \max(k') = 0-(K-1) = -K+1$

當 $n'-k'<0$ 時， $-K+1 \leq n'-k' \leq -1$

$$P-K+1 \leq ((n'-k'))_P \leq P-1$$

因為 $P \geq N+K-1$, $P-K+1 \geq N$ 又 $x_1[n] = 0$ for $n \geq N$ and $n < 0$

所以當 $n'-k'<0$ 時 $x\left[((n'-k'))_P\right] = x\left[n'-k'\right] = 0$

$$y_1[n'] = \sum_{k'=0}^{K-1} x_1[n'-k']h_1[k']$$

$$y[n'+n_1+k_1] = \sum_{k'=0}^{K-1} x[n'-k'+n_1]h[k'+k_1]$$

令 $k = k' + k_1,\ \ n = n' + n_1 + k_1$

$$y[n] = \sum_{k=k_1}^{k_1+K-1} x[n-n_1-k_1-k+k_1+n_1]h[k]$$

$$y[n] = \sum_{k=k_1}^{k_2} x[n-k]h[k] \qquad \text{得證}$$

Case 3: $x[n]$ has finite length but $h[n]$ has infinite length

$x[n]$ 的範圍為 $n \in [n_1, n_2]$，範圍大小為 $N = n_2 - n_1 + 1$

$h[n]$ 無限長

$y[n] = \sum_k x[n-k]h[k]$ $y[n]$ 每一點都有值 (範圍無限大)

但我們只想求出 $y[n]$ 的其中一段

希望算出的 $y[n]$ 的範圍為 $n \in [m_1, m_2]$，範圍大小為 $M = m_2 - m_1 + 1$

$h[n]$ 的範圍？

要用多少點的 FFT？

$$y[n]=x[n]*h[n]=\sum_{k=-\infty}^{\infty}x[n-k]h[k]$$

改寫成 $$y[n]=x[n]*h[n]=\sum_{s=n_1}^{n_2}x[s]h[n-s]$$

$$y[n]=x[n_1]h[n-n_1]+x[n_1+1]h[n-n_1-1]+x[n_1+2]h[n-n_1-2]$$
$$+\cdots\cdots+x[n_2]h[n-n_2]$$

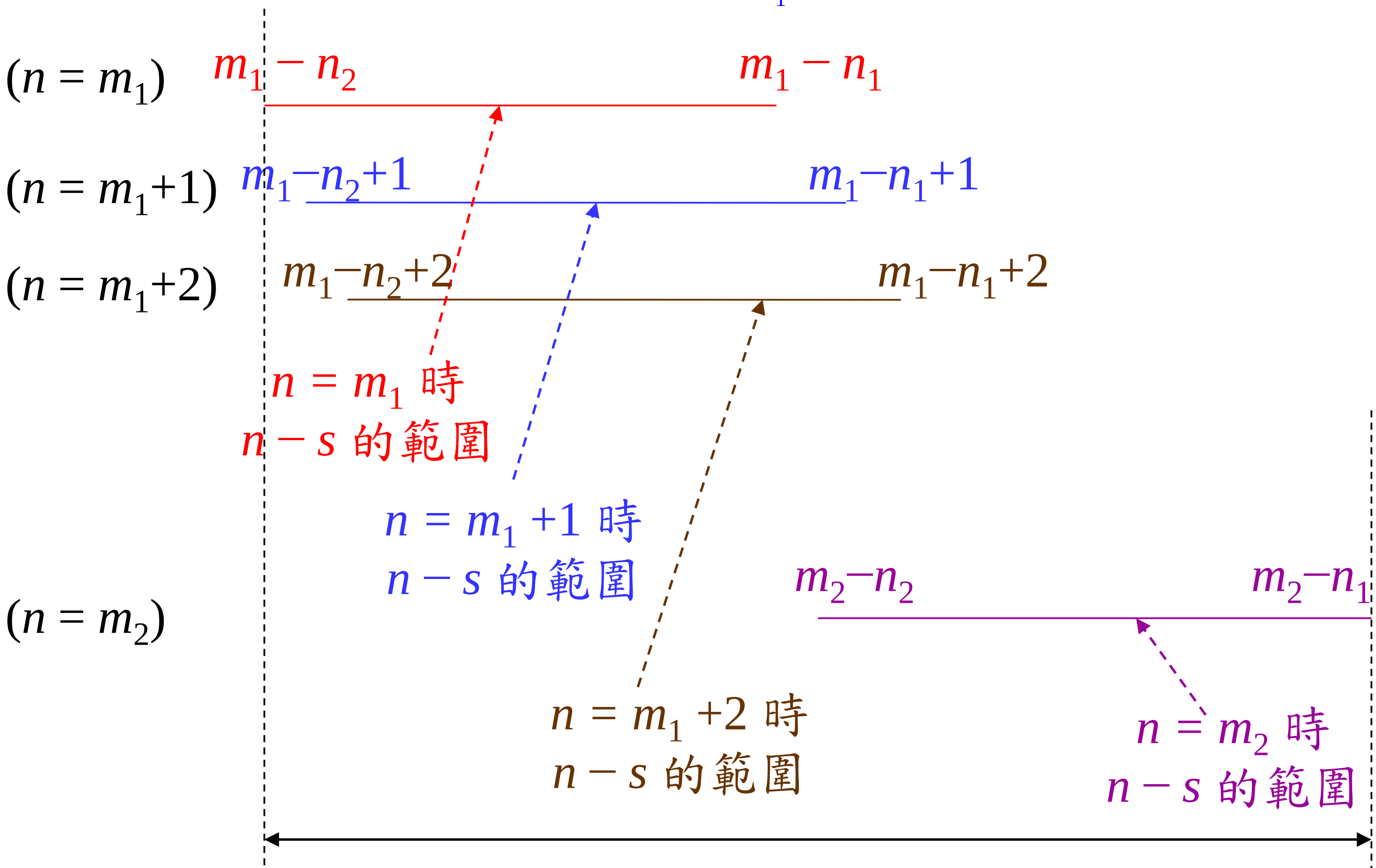
當$n=m_1$

$$y[m_1]=x[n_1]h[m_1-n_1]+x[n_1+1]h[m_1-n_1-1]+x[n_1+2]h[m_1-n_1-2]$$
$$+\cdots\cdots+x[n_2]h[m_1-n_2]$$

當$n=m_2$

$$y[m_2]=x[n_1]h[m_2-n_1]+x[n_1+1]h[m_2-n_1-1]+x[n_1+2]h[m_2-n_1-2]$$
$$+\cdots\cdots+x[n_2]h[m_2-n_2]$$

此圖為 $n-s$ 範圍示意圖　　$y[n]=\sum_{s=n_1}^{n_2} x[s]h[n-s]$



有用到的 $h[k]$ 的範圍：$k \in [m_1-n_2, m_2-n_1]$

所以有用到的 $h[k]$ 的範圍是 $k \in [m_1 - n_2\,, m_2 - n_1\,]$

範圍大小為 $m_2 - n_1 - m_1 + n_2 + 1 = N + M - 1$

FFT implementation for Case 3

$x_1[n] = x[n + n_1]$ for $n = 0, 1, 2, \ldots, N-1$

$x_1[n] = 0$ for $n = N, N+1, N+2, \ldots\ldots, P-1$ $P \geq N + M - 1$

$h_1[n] = h[n + m_1 - n_2]$ for $n = 0, 1, 2, \ldots, L-1$

$$y_1[n] = IFFT_P\left(FFT_P\{x_1[n]\}\, FFT_P\{h_1[n]\}\right)$$

$y[n] = y_1[n - m_1 + N - 1]$ for $n = m_1, m_1+1, m_1+2, \ldots, m_2$

$$n - m_1 + N - 1 = N - 1, N, \cdots\cdots, N + M - 2$$

注意：$y[n]$ 只選 $y_1[n]$ 的第 N 個點到第 $N+M-1$ 個點

⦿ 11-D Relations between the Signal Length and the Convolution Algorithm

Suppose that

$x[n]$: input, $h[n]$: the impulse response of the filter

length($x[n]$) = N, length($h[n]$) = M (Both of them have finite lengths)

We want to compute

$$y[n] = \sum_{m=0}^{M-1} x[\overset{\text{N}}{n-m}]\, \overset{\text{M}}{h[m]} \quad , \qquad y[n] = x[n] * h[n].$$

The above convolution needs the P-point DFT, $P \geq M + N - 1$.

complexity: $O(P \log_2 P)$

Case 1: When M is a very small integer:

Directly computing

Number of multiplications for directly computing:

$N \times M$

When $3N \times M \leq 9/2\times(N + M - 1)\log_2(N + M - 1)$, i.e.,

$\boldsymbol{M \leq (3/2)\log_2 N}$, (粗略估計)

it is proper to do directly computing instead of applying the DFT.

Example: $N = 126$, $M = 3$, (difference, edge detection)

$(3/2)\log_2 N = 10.4659$

When compute the number of real multiplications explicitly,

using direct implementation: $3N \times M = 1134$,

using the 128-point DFT:

using the 144-point DFT:

Although in usual "directly computing" is not a good idea for convolution implementation, in the cases where

(a) M is small

(b) The filter has some symmetric relation

using the directly computing method may be efficient for convolution implementation.

Example: edge detection

$h[n]$ = [-0.1, - 0.3, -0.6, 0, 0.6, 0.3, 0.1] for n = -3 ~ 3

Example: smooth filter

$h[n] = [0.1,\ 0.2,\ 0.4,\ 0.2,\ 0.1]$ for $n = -2 \sim 2$

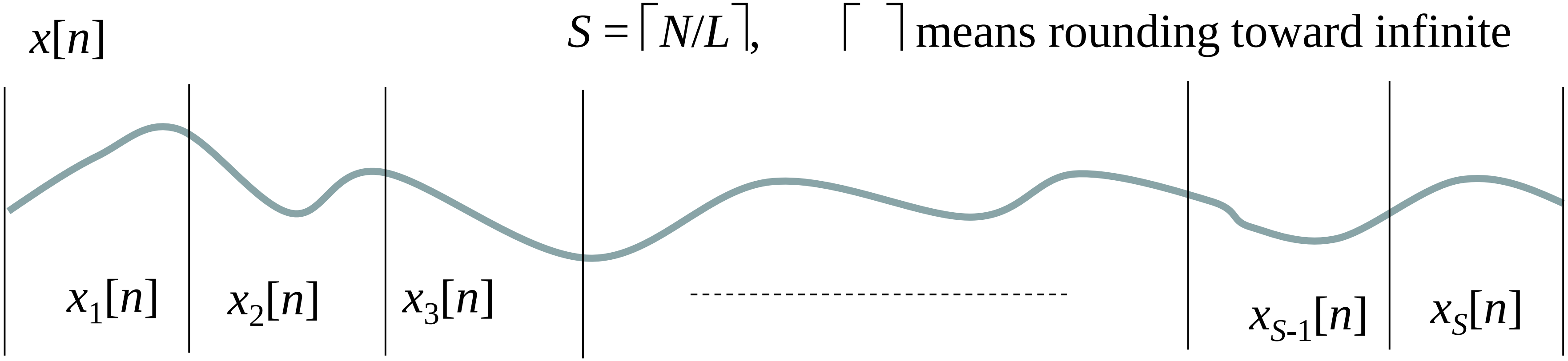
Case 2: When M is not a very small integer but much less than N ($N >> M$):

It is proper to divide the input $x[n]$ into several parts:

Each part has the size of L ($L > M$).

$$x[n]\ (n = 0, 1, \ldots, N-1) \rightarrow x_1[n],\ x_2[n],\ x_3[n], \ldots\ldots, x_S[n]$$

$S = \lceil N/L \rceil$, $\lceil\ \rceil$ means rounding toward infinite



Section 1 $\quad x_1[n] = x[n]$ $\quad$ for $n = 0, 1, 2, \ldots, L-1$,

Section 2 $\quad x_2[n] = x[n + L]$ $\quad$ for $n = 0, 1, 2, \ldots, L-1$,

:

Section s $\quad x_s[n] = x[n + (s-1)L]$ $\quad$ for $n = 0, 1, 2, \ldots, L-1$,

$s = 1, 2, 3, \ldots, S$

$$x[n] = \sum_{s=1}^{S} x_s[n-(s-1)L]$$

$$y[n] = x[n] * h[n] = \sum_{s=1}^{S} x_s[n-(s-1)L] * h[n]$$

$$= \sum_{s=1}^{S} \sum_{m=0}^{M-1} \underbrace{x_s[n-(s-1)L-m]}_{\text{length} = L} \underbrace{h[m]}_{\text{length} = M}$$

It should perform the P-point FFTs $2S+1$ times Why?

or $2S$ times

$$P \geq L+M-1$$

Detail of Implementation

Suppose that the P-point DFT is applied for each section

$$x[n] = 0 \quad \text{when } n < 0 \text{ and } n \geq N$$

(1) First, determine $L = P - M + 1$

(2) $x_s[n] = x\left[(s-1)L + n\right]$ for $n = 0, 1, 2, \ldots, L-1,$ $s = 1, 2, 3, \ldots, S$

$x_s[n] = 0$ for $n = L, L+1, \ldots, N-1$ $S = \lceil N/L \rceil$

$h_1[n] = h\left[n\right]$ for $n = 0, 1, 2, \ldots, M-1,$

$h_1[n] = 0$ for $n = M, M+1, \ldots, N-1$

(3) Then calculate

$$y_s[n] = IDFT_P\left\{DFT_P\left(x_s[n]\right)DFT_P\left(h_1[n]\right)\right\}$$

(4) Then, apply "overlapped addition"

$$y[n] = \sum_{s=1}^{S} y_s[n-(s-1)L]$$

replaced by the number of multiplications for the P-point DFT

運算量：$2S \times \boxed{[(3P/2)\log_2 P]} + 3S \times P$ $\quad S \approx N/L, \quad P \approx L+M-1$

(理論值)

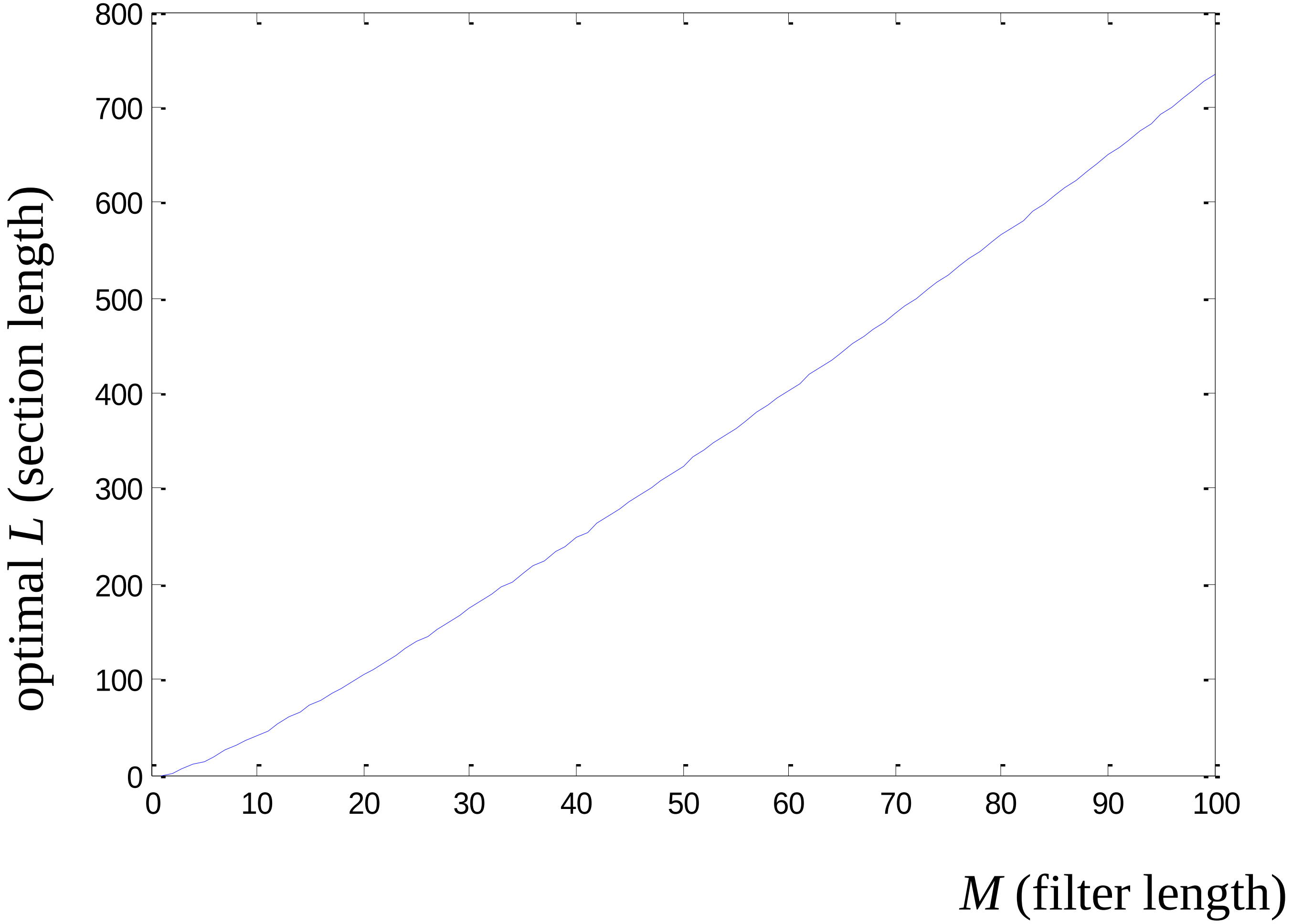
$$\approx \frac{N}{L} 3(L+M-1)\left[\log_2(L+M-1)+1\right]$$ (linear with N)

何時為 optimal? $\rightarrow \dfrac{\partial 運算量}{\partial L} = 0$

$$N\frac{L-(L+M-1)}{L^2}\left[\log_2(L+M-1)+1\right] + N\frac{L+M-1}{L}\frac{1}{(L+M-1)\log 2} = 0$$

$$L = (M-1)\left[\log(L+M-1)+\log 2\right]$$

In practice, a computer program is applied to determine the optimal L.


optimal L (section length)
800
700
600
500
400
300
200
100
0
0
10
20
30
40
50
60
70
80
90
100
M (filter length)

注意：

(1) Optimal section length is independent to N

(2) If M is a fixed constant, then the complexity is linear with N, i.e.,

$$O(N)$$

比較：使用原本方法時， complexity = $O((N+M\text{-}1)\log_2(N+M\text{-}1))$

(3) 實際上，需要考量 P-point FFT 的乘法量必需不多

$$P = L+M-1$$

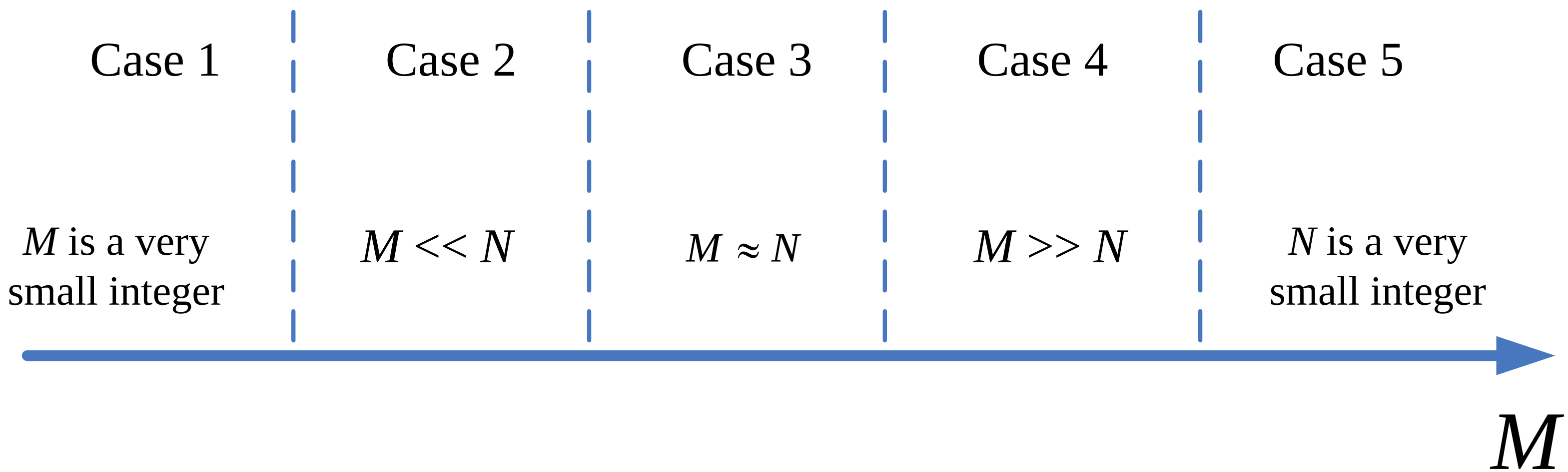
例如，根據 page 404 的方法，算出當 $M = 10$ 時， $L = 41.5439$ 為 optimal

但實際上，應該選 $L = 39$，因為此時 $P = L+M-1 = 48$ 點的 DFT 有較少的乘法量

Case 3 When M has the same order as N

Case 4 When M is much larger than N

Case 5 When N is a very small integer



- **Sectioned Convolution for the Condition where One Sequence is Finite and the Other One is Infinite**

$$y[m] = \sum_n x[n]h[m-n]$$

$x[n] \neq 0$ for $n_1 \leq n \leq n_2$, length of $x[n] = N = n_2 - n_1 + 1$,

length of $h[n]$ is infinite,
and we want to calculate $y[m]$ for $m_1 \leq m \leq m_2$, $M = m_2 - m_1 + 1$.

Suppose that $M << N$.

In this case, we can try to partition $x[n]$ into several sections.

section 1: $x_1[n] = x[n]$ for $n = n_1 \sim n_1 + L - 1$, $x_1[n] = 0$ otherwise,
section 2: $x_2[n] = x[n]$ for $n = n_1 + L \sim n_1 + 2L - 1$, $x_2[n] = 0$ otherwise,

$\vdots$

section q: $x_q[n] = x[n]$ for $n = n_1 + (q-1)L \sim n_1 + qL - 1$, $x_q[n] = 0$ otherwise,

$\vdots$

Then we perform the convolution of $x_q[n] * h[n]$ for each of the sections by the method on pages 394 and 395.

(Since the length of $x_q[n]$ is L, it requires the P-point DFT,

$$P \geq L+M-1.$$

Its complexity and the optimal section length can also be determined by the formulas on page 404.

◉ 11-E Recursive Method for Convolution Implementation

$$y[n] = \sum_{m=0}^{N-1} x[n-m] a \cdot b^m = x[n] * a \cdot b^n u[n]$$

$u[n]$: unit step function

$$Y(z) = X(z) \frac{a}{1 - bz^{-1}}$$

$$(1 - bz^{-1}) Y(z) = aX(z)$$

$$y[n] = by[n-1] + ax[n]$$

Only two multiplications required for calculating each output.

$$y[n] = x[n] * h[n]$$

$$h[n] = 0.25 \cdot 0.6^{|n|}$$

$$H(z) = \frac{0.25}{1-0.6z^{-1}} + \frac{0.25}{1-0.6z} - 0.25$$

$$= 0.25 \frac{1}{\frac{1.36}{0.64} - \frac{0.6}{0.64}\left(z^{-1} + z\right)}$$

12. Fast Algorithm 的補充

◉ 12-A Discrete Fourier Transform for Real Inputs

DFT: $$F[m] = \sum_{n=0}^{N-1} f[n] e^{-j\frac{2\pi}{N}mn}$$

當 $f[n]$ 為 real 時， $F[m] = F^*[N-m]$ *: conjugation

若我們要對兩個 real sequences $f_1[n]$ ，$f_2[n]$ 做 DFTs

Step 1: $f_3[n] = f_1[n] + j\,f_2[n]$

Step 2: $F_3[m] = \text{DFT}\{f_3[n]\}$

Step 3: $F_1[m] = \dfrac{F_3[m] + F_3^*[N-m]}{2}$ $\qquad F_2[m] = \dfrac{F_3[m] - F_3^*[N-m]}{2j}$

只需一個 DFT

證明：由於 DFT 是一個 linear operation

$$F_3[m] = F_1[m] + jF_2[m]$$

又 $F_1[m] = F_1{}^*[N-m]$ $\qquad F_2[m] = F_2{}^*[N-m]$

$$\begin{aligned} F_3[m] + F_3^*[N-m] &= F_1[m] + jF_2[m] + F_1^*[N-m] - jF_2^*[N-m] \\ &= 2F_1[m] \end{aligned}$$

$$F_3[m] - F_3^*[N-m] = j2F_2[m]$$

同理，當兩個 inputs 為

(1) pure imaginary

(2) one is real and another one is pure imaginary

時，也可以用同樣的方法將運算量減半

- 若 input sequence 為 even $f[n] = f[N\text{-}n]$，

  則 DFT output 也為 even $F[n] = F[N\text{-}n]$
- 若 input sequence 為 odd $f[n] = -f[N\text{-}n]$，

  則 DFT output 也為 odd $F[n] = -F[N\text{-}n]$

若 input sequence 為 odd and real，

則乘法量可減為 1/4

[Corollary 1]

If it is known that the IDFTs of $F_1[m]$ and $F_2[m]$ are real, then the IDFTs of $F_1[m]$ and $F_2[m]$ can be implemented using only one IDFT:

(Step 1) $F_3[m] = F_1[m] + j\,F_2[m]$

(Step 2) $f_3[n] = \mathrm{IDFT}\{F_3[m]\}$

(Step 3) $f_1[n] = \mathcal{Re}\{f_3[n]\}$, $f_2[n] = \mathcal{Im}\{f_2[n]\}$,

[Corollary 2]

When $x[n]$ and $h[n]$ are both real, the computation loading of the convolution (or the sectioned convolution) of $x[n]$ and $h[n]$ can be halved.

◎ 12-B Converting into Convolution

一般的 linear operation:

$$z[m]=\sum_{n=0}^{N-1}x[n]k[m,n]$$ (習慣上，把 $k[m, n]$ 稱作 “kernel”)

$n = 0, 1, \ldots, N-1, \quad m = 0, 1, \ldots, M-1$

可以用矩陣 (matrix) 來表示

運算量為 MN

若為 linear time-invariant operation:

$$z[m]=\sum_{n=0}^{N-1}x[n]h[m-n]$$ $k[m, n] = h[m-n]$ (dependent on m, n 之間的差)

$n = 0, 1, \ldots, N-1, \quad m = 0, 1, \ldots, M-1$

$m-n$ 的範圍： 從 $1-N$ 到 $M-1$， 全長 $M+N-1$

運算量為 $L\log_2 L$, $\quad L \geq M+2N-2$

大致上，變成 convolution 後 總是可以節省運算量

例子A $z[m]=\sum_{n=0}^{N-1}x[n]2^{mn}$

可以改寫為

$$z[m]=2^{\frac{m^2}{2}}\sum_{n=0}^{N-1}\left\{x[n]2^{\frac{n^2}{2}}\right\}2^{\frac{-(n-m)^2}{2}}$$

convolution

運算量為 $M+N+L\log_2 L$

例子B Linear Canonical Transform

$$y(k)=A\int_{-\infty}^{\infty}e^{j\alpha k^2+j\beta kt+j\gamma t^2}x(t)\,dt$$

$$\downarrow$$

$$y(k)=A\int_{-\infty}^{\infty}e^{j(\alpha+\beta/2)k^2-j\frac{\beta}{2}(k-t)^2+j(\gamma+\beta/2)t^2}x(t)\,dt$$

通則：

當$k[m, n]$ 可以拆解成 $A[m] \times B[m-n] \times C[n]$

或 $$k[m,n] = \sum_s A_s[m] B_s[m-n] C_s[n]$$

即可以使用 convolution

⦿ 12-C LUT

LUT (lookup table)

道理和背九九乘法表一樣

記憶體容量夠大時可用的方法

Problem: memory requirement

wasting energy

九九乘法表的例子

附錄十二　創意思考

New ideas 聽起來偉大，但大多是由既有的 ideas 變化而產生

(1) Combination

(2) Analogous

(3) Connection

(4) Generalization

(5) Simplification

(6) Reverse

註：感謝已過逝的李茂輝教授，他開的課「創造發明工程」，
讓我一生受用無窮

(7) Key Factor Analysis

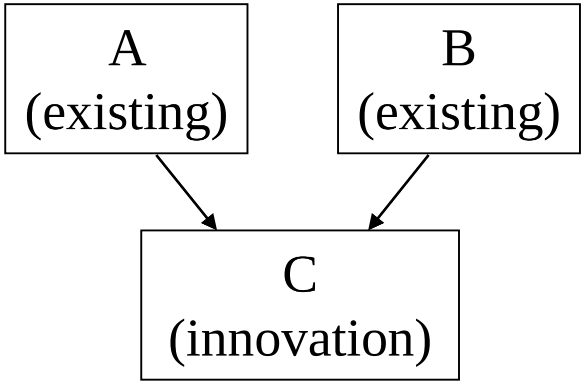
(8) 胡思亂想，純粹意外

(7) Key Factor

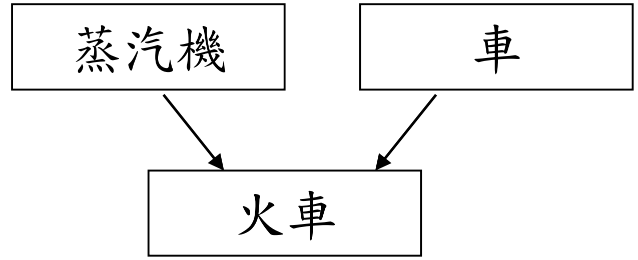
(8) Imagination

(9) 純粹意外

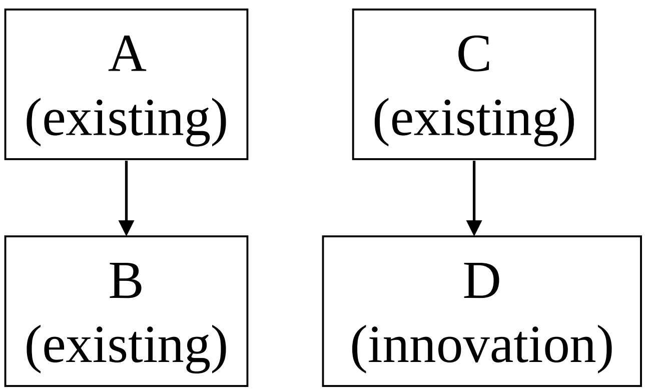
(1) Combination



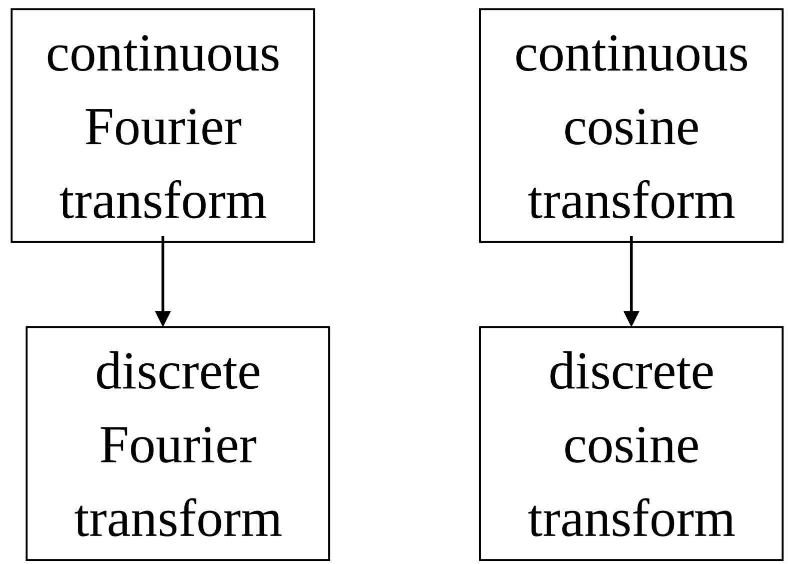
例：



(2) Analog



例：



(3) Connection

A (existing) → C (innovation) ← B (existing)

例：

古典物理學 → 相對論 ← 未能解釋的光學現象

(4) Generalization

A (existing)

↙ ↘

保留某些特性 | 將其他特性推廣化

↘ ↙

B (innovation)

例：

Walsh transform

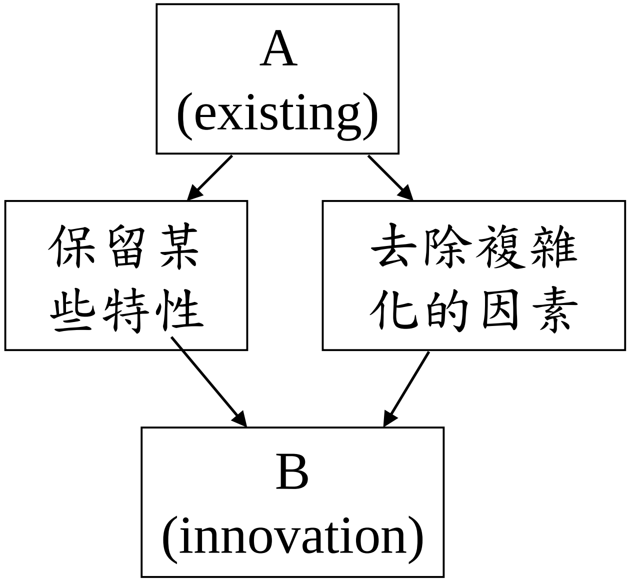
↙ ↘

保留不需乘法 | 將 ±1 推廣為 $\pm 2^k$

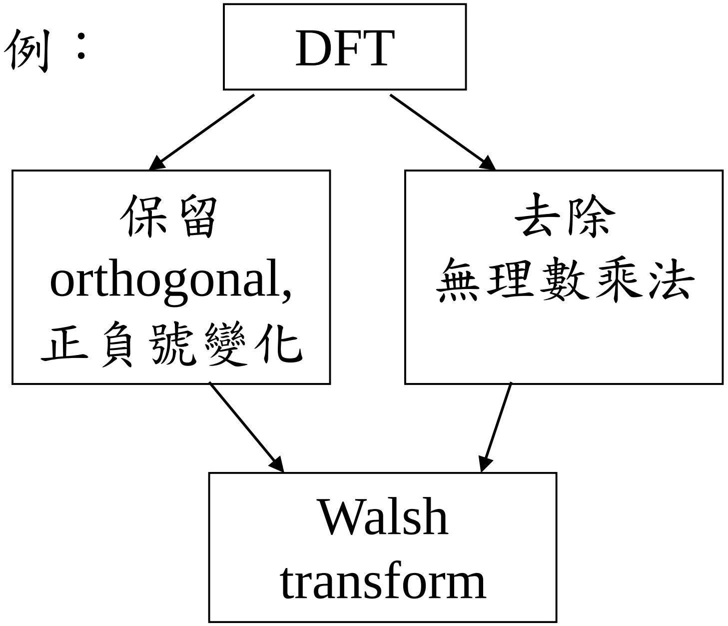
↘ ↙

Jacket transform

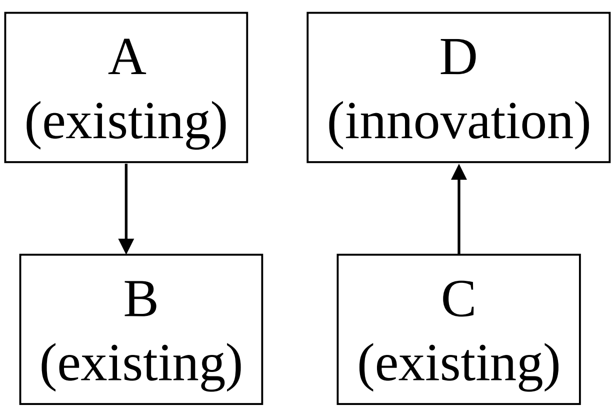
(5) Simplification

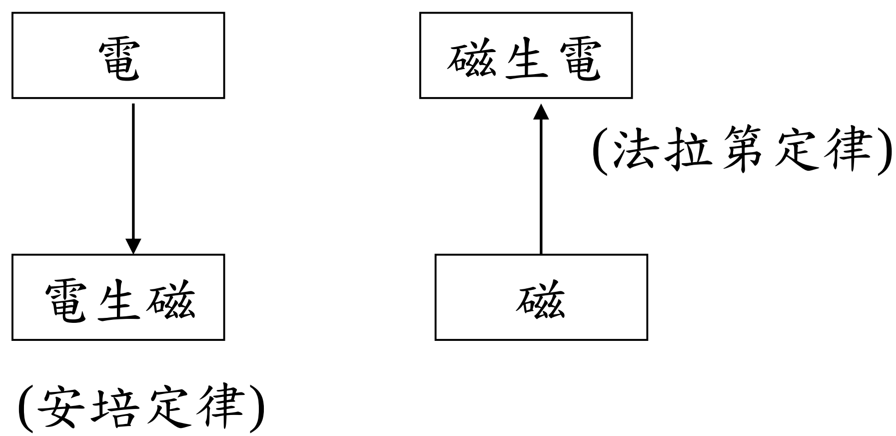


例：



(6) Reverse





個人做研究時，關於發明創造的心得

(1) 研究問題的第一步，往往是先想辦法把問題簡化

如果不將複雜的經濟問題簡化成二維供需圖，經濟學也就無從發展起來

如果不將電子學的問題簡化常小訊號模型，電子電路的許多問題都將難以解決

個人在研究影像處理時，也常常先針對 size 很小，且較不複雜的影像來做處理，成功之後再處理 size 較大且較複雜的影像

問題簡化之後，才比較容易對問題做分析，並提出改良之道

(2) 如果有好點子，趕快用筆記下來，

好點子是很容易稍縱即逝而忘記的。

(3) 練習多畫系統圖

系統圖畫得越多，越容易發現新的點子

(4) 其實，對台大的同學而言，提出 new ideas 並不難，但是要把 ideas 變成有用的、成功的 ideas，不可以缺少分析和解決問題的能力

很少有一個 new idea 一開始就 works well for any case，任何一個成功的創意，都是經由問題的分析，解決一連串的技術上的問題，才產生出來的

(5) 當心情放鬆時，想像力特別強，有助於發現意外的點子。

(6) 就短期而言，技術性的問題固然重要

但是就長期而言，不要因為技術上的困難，而否定了一個偉大的構想

大學以前的教育，是學習前人的智慧結晶

研究所的教育，是訓練創造發明和解決問題的能力