CH6、NP-Complete

NP 完全

考題重點(目錄)：

1. 基本概念
2. Reduce 概念
3. P, NP, NP-hard, NP-Complete
4. 證明過程
5. 近似演算法

基本概念

一、目的：將問題依難度做分類

1. 如何定義"A 比 B 難"
2. 要分幾類？
3. 如何做出分類的動作？

二、Decision Problem

答案為 yes/no 的問題

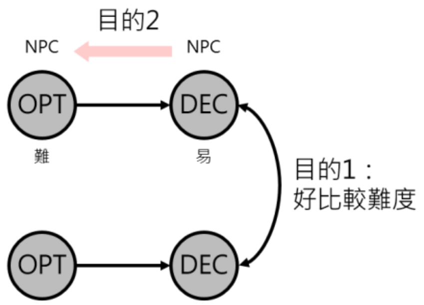
例："一個 Graph 中有無 Hamiltanian Cycle"為 Decision Problem

三、每一個 Optimization Problem 均可用參數化的方式，化成一對應的 Decision Problem

例：

KP-OPT：給定一背包負重 W 及 n 個 item，問最大 Profit 為何？

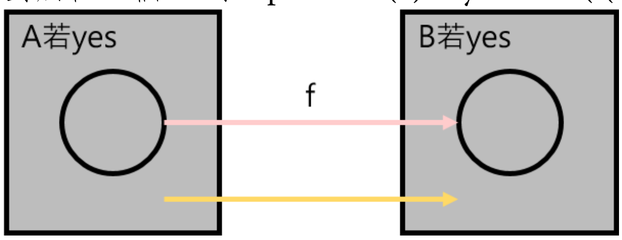
KP-DEC：給定一背包負重 W，n 個 item 及一個整數 k，問：Profit 是否可大於 k？



Reduce 概念

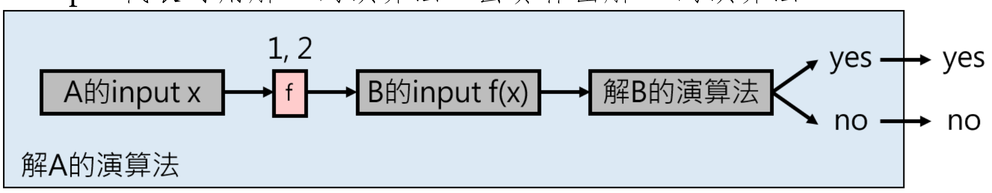
一、Def：設 A, B 是 2 個問題，A 可以 Polynomial-Time Reduce 到 B，則滿足：

1. 存在一個函數 f：A 的 input -> B 的 input (transformation)
2. f 為 Polynomial-Time Computable
3. 對於任一個 A 的 input x，A(x) ="yes" ⇔ B(f(x)) ="yes"

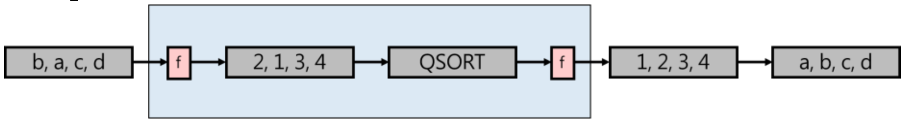


Note：A 可以 Polynomial-Time Reduce 到 B，記作：A <=p B

二、A <=p B 代表可用解 B 的演算法，去實作出解 A 的演算法



三、A <=p B 代表 A 的難度 <= B 的難度(B 至少比 A 難)[Q1 的答案]



排 a, b, c ,d 是一種排 1, 2, 3, 4 的問題

P, NP, NP-hard, NP-Complete[Q2 **的答案**]

一、Def：以下 4 者皆為 a set of problems

1. P：在其中的 Problem 皆有 Deterministic Algorithm 可在 Polynomial-Time 中解之
   例：”排序 n 個數”屬於 P*(因為存在一個演算法 QSORT，可在 Theta(n lg n)中解之)*

2. NP*(easy to verify)*：
   (1) 在其中的 Problem 皆有 Non-deterministic Algorithm 可在 Polynomial-Time 中解之*(注意：NP 不等於 Not-P)*
   (2) *(理解用)*給一個可能的解，若可在 Polynomical-Time 內，判斷其是否為該問題的解，則此問題屬於 NP

   小結：P 包含於 NP

3. NP-hard*(NP 難問題)*：對於所有 Q 屬於 NP，Q <=p A ⇔ A 屬於 NP-hard

比喻：
所有正妹的集合

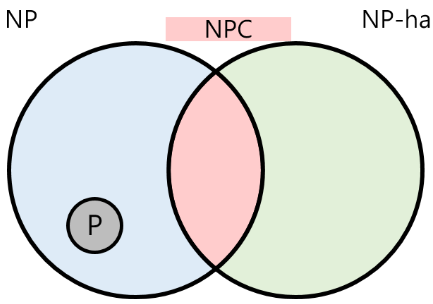
4. NP-Complete：A 屬於 NP-Complete ⇔ A 屬於 NP、且 A 屬於 NP-hard

比喻：
是班上所有女生的一員，而且也是正妹

   在 NP 中的問題，均有演算法可以解，所以 NP-Complete 中的問題即：有演算法可解的問題中，最難的那些

   Note：NPC 的問題：在『worst case』下，沒有『Polynomial-Time』的演算法可解(至少都要 Exponential Time)

二、一般相信四者關係如下*(在 P 不等於 NP 的假設下)*：



在無任何假設下，”P=NP”尚無定論

例：True/False

1. P 包含於 NP？
2. P 包含 NP？
3. P=NP？
4. P!=NP？

True：1；False(Unknown)：2, 3, 4

三、有一個在 NPC 的問題，可在 worst case 下被一個 Polynomial-Time 的演算法解決 ⇔ P=NP

證明一個問題為 NPC(Q3)

一、證明方式：

欲證 A 屬於 NPC，則：

1. 證 A 屬於 NP
2. 任找一個 B 屬於 NPC，證 B <=p A　　*//等價於證明：A 屬於 NP-hard*

Note：注意方向

例(94 交大)：(True/False)已知 CLIQUE 屬於 NPC，VERTEX-COVER 屬於 NP，欲證 VERTEX-COVER 屬於 NPC，則將 VERTEX-COVER <=p CLIQUE 即可

False：方向錯誤

例(99 政大)：(True/False)欲證 A 為 NPC，則將一個 B 屬於 NPC reduce 到 A 即可？

False：2.B <=p A 是對的，方向也正確，但沒有證明 1. A 屬於 NP

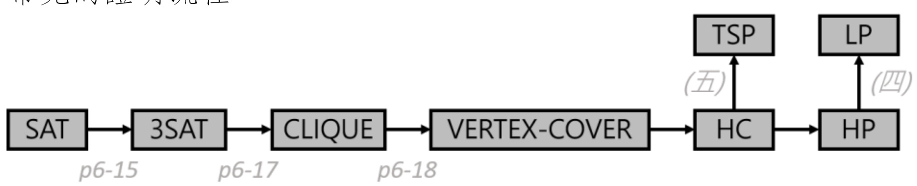
二、第一個 NP-Complete 的問題為 SAT problem

SAT problem：給一個 CNF F，問有無一組 assignment 可使 F 為真？

例：F=(x1 v !x2 v !x3) ^ (!x1 v x3) ^ (x2 v !x3)

有 assignment 可使 F 為真：{x1 = T, x2 = T, x3 = T}
因此，若將此 F 當做 SAT problem 的 input，則其 answer 為 True

三、常見的證明流程：



四、Longest Path Problem 屬於 NPC

證明：Longest Path Problem：給定一個 Graph G=(V, E)及 k，問 G 中有沒有長度 >= k 的 simple path？

1. claim LP 屬於 NP

   給定一個 LP：1.input: (G, k)及任一條 simple path P，則可在 Polynomial-Time 中，判斷 P 是否為 G 中長度 >=k 的 simple path

   ⇨ LP 屬於 NP

   P=<u1, u2, …, um>為給定的 path

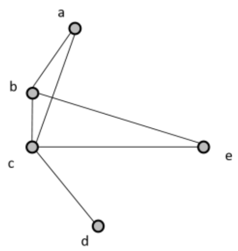
   (1) check m>=k+1？若 yes 則(2)；若 no 則錯誤 *//Theta(1)*

   (2) check (ui, ui+1)屬於 E？若 yes 則正確；若 no 則錯誤 *//Theta(m)：G 用 adj matrix 表示，判斷一邊是否屬於 E => Theta(1)*

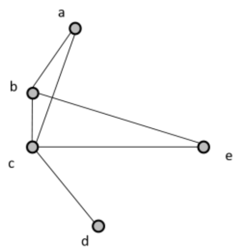
2. claim HP <=p LP

   (1) 給定一個 HP 的 input：G=(V, E)，定義一個函數 f: G -> (G, k=|V|-1)，則(G, |V|-1)為 LP 的 input

   例：



G



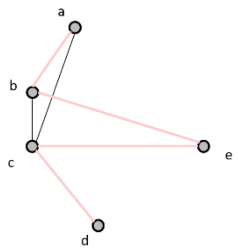
G, k=4

   (2) 顯然地，f 為 Polynomial-Time Computable

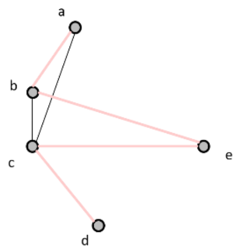
   (3) G 有 HP ⇔ G 有長度 >= |V|-1 的 simple path，G 有 HP P

   ⇨ P 必過 G 中每點恰一次，且 P 為 simple path

   ⇨ P 為 G 中長度 >= |V|-1 的 simple path



G

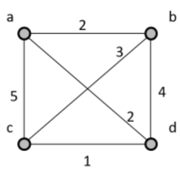


G, k=4

五、TSP 屬於 NPC：Travelling Salesman Problem

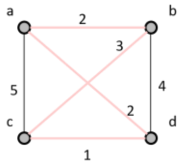
給定一個無向、有權重的完全圖 G=(V, E)及 k，問 G 中有沒有 weight 和 <= k 的 HC？

例：



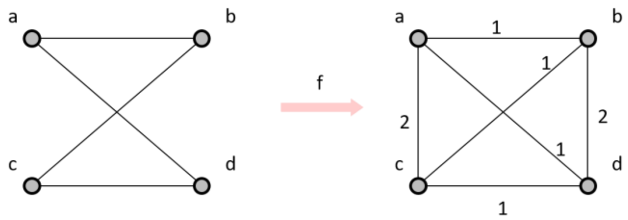
G, k=10

給定(G, 10)為 TSP 的 input，則 answer 為 yes(因為存在一個 HC：C 其 weight 和 <= 10)
證明：



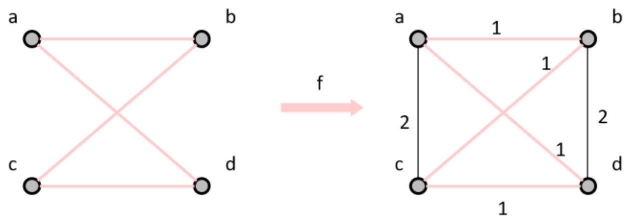
G, k=10

1. claim TSP 屬於 NP：給定一個無向、有權重的完全圖 G=(V, E)及 k，和一個其上的 HC：C，則可在 Polynomial-Time 中驗證 C 的 weight 和是否<=k
   ⇨ TSP 屬於 NP
2. Claim HC <=p TSP
   (1) 給定一個 HC 的 input：G=(V, E)，定義一個 f: G -> (G’, k=|V|)，其中 G’的定義如下：
   G’=(V, E’)，　若(u, v)屬於 E，則(u, v)屬於 E’，且 weight 為 1
   　　　　　　　若(u, v)不屬於 E，則(u, v)屬於 E’，且 weight 為 2
   例：



G　　　　G, k=4

   (2) 顯然地，f 為 Polynomial-Time Computable
   (3) [順向]
   G 有 HC ⬄ G’有 weight <= |V|的 HC
   G 有 HC：C
   ⇨ C 是一個 cycle 通過每點一次
   ⇨ C 上的每一個邊屬於 E
   ⇨ C 上的每一邊在 G’的 weight=1
   ⇨ C 在 G’中為 weight <= |V|的 HC



G　　　　G, k=4

   [逆向]
   G’中有 weight <= |V|的 HC：C
   ⇨ C 中邊的 weight 均為 1
   ⇨ C 中每邊屬於 E
   ⇨ C 為 G 的 HC

例(100 交大)(p6-52.22)：

1. (TSP)給定一個 n*n 的 Distance matrix，n 個 city B，有無一個 tour 經每個 city 恰一次，且 Distance 和 <= B
2. (HC 的變形)給定一個無向圖，2 個點 s, t，問可否自 s -> t 且經每點恰一次的 path？
3. (HC)給一無向圖，有無 cycle 可過每點恰一次？

1. *正確：因為 a 屬於 NP，且所有的 b 屬於 NP，b <=p a*
2. *正確*
3. *錯誤：必須連 s, t 才行*
4. *正確*
5. *正確：HP 無論 Graph 是有向或無向均為 NPC*

3. *例題*
   (1) *正確*
   (2) *正確*
   (3) *正確，問題 1 為原本最佳化 TSP 的決定性版本*
   (4) *錯誤：reduction 並無此含意*
   (5) *正確*

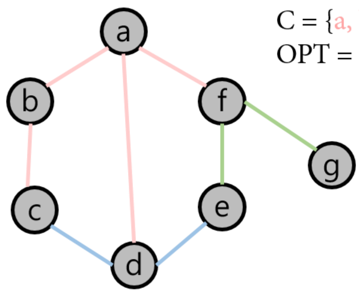
Approximation Algorithm

一、Approximation Ratio：設 A 是一個 Approximation Algorithm，OPT 是可解出最佳解的演算法。對於任一個 input x，|A(x)|*(以 x 為 input，A 產生的解的大小)*小於等於 alpha|OPT(x)|，則稱 A 的 Approximation Ratio 為 alpha。*(假設處理的問題為最小化問題)*

二、Minimum Vertex Cover 的 Approximation Algorithm

程式：

```
C <- 0;        //記 Vertex Cover 的球員
E' <- E;
while(E'!=0)
{
        (u, v) <- E'之任一邊;
        C <- C 連集{u, v};
        將 E'所有含 u 及 v 的邊去除
}
return C;
```



C = {a, b, c, d, e, f}
OPT = {b, d, e}

Time Complexity：Theta(|E|)，*因為在 while 中，每次至少會去掉 1 個邊，又在 E'=0 時，跳出 while，所以 Theta(|E|)*

Approximation Ratio：對於每個邊而言，至少有 1 個 Vertex 會屬於 Minimum Vertex Cover C*中，在此演算法中，每次挑一邊，並將 2 個點，均加入 C 中，因此，在 worst case 下，挑到的邊恰只有一個 Vertex 在 C*中因此，|C| <= 2|C*|

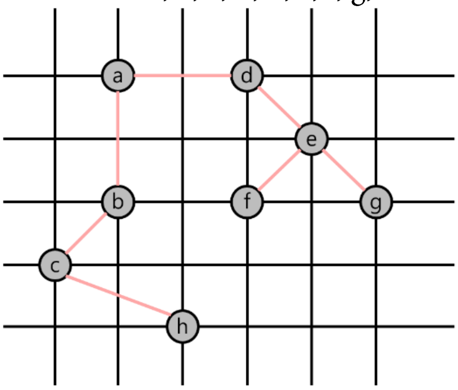
⇨ Approximation Ratio = 2

三、Euclidean TSP

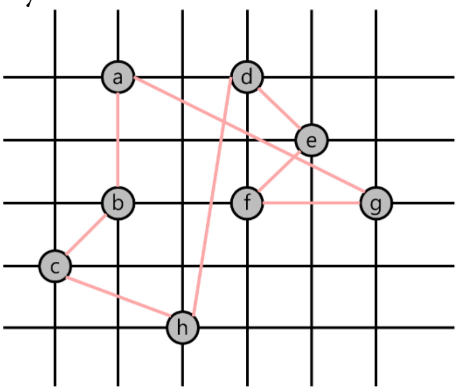
1. problem statement：給定平面上的 n 個點，求一個 cycle 經過每個點恰一次、且 Euclidean distance 和最小
2. 演算法：
   (1) 選一個點 v 當做 root
   (2) 以 v 為起點，用 Prim's Algorithm 算出 n 個點的 MST
   (3) 令 L 是 MST 上做 Preorder traversal 的順序
   (4) 依 L 的順序將點相連，則可得到欲求的 cycle C

   例(p6-30.4.4)：

   Preorder：a, b, c, h, d, e, f, g,



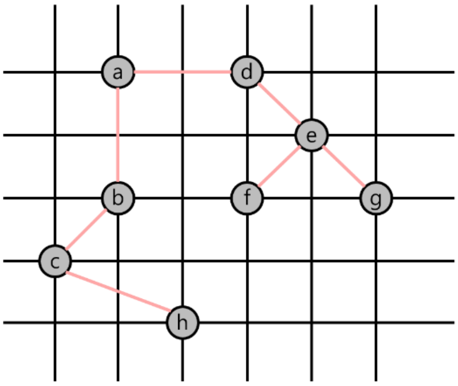
   Cycle：



3. Time Complexity：Theta(n^2)
   1. Theta(1)
   2. Theta(n^2), n 為點數
   3. Theta(n)
   4. Logical 上的步驟，依 3.結果 output 即可
   ⇨ Theta(n^2)

4. Approximation Ratio：設有最小 distance 和的 cycle 為 C*，令 T 為步驟所求出的 MST。將 C*去掉任一邊 => 形成一個 Spanning Tree T'
   因為 T 是 MST，所以 cost(T) <= cost(T') <= cost(C*)…(1)

   設 w 為找 preorder 時的 full walk(以上例：w=a, b, c, h, c, b, a, d, e, f, e, g, e, d)，又 MST 的邊被 full walk 走過 2 次，因此：cost(w)=2cost(T)…(2)



   又因為 Euclidean distance 符合三角不等式*(即 dist(a, b) <= dist(a, c)+dist(c, a))*，且用演算法求出的 cycle C，可由將 full walk 去掉若干點得到*(以上例：w=a, b, c, h, c, b, a, d, e, f, e, g, e, d, a、C=a, b, c, h, x, x, x, d, e, f, x, g, x, x, a)*，因此：cost(C) <= cost(w)…(3)

   By (2)與(3) => cost(C) <= 2cost(T)…(4)
   By (1)與(4) => cost(C) <= 2cost(C*)
   ⇨ Approximation Ratio=2