姓名:胡劭 系級:資工碩一 學號:M11215075

1. By using the reduction technique, explain how would you solve each of the following questions:

   (a) Finding the median: Given a set of numbers, find the median value.

Sol:

如果我們能夠有效地解決排序問題，我們也能夠有效地找到中位數，

由老師提供的講義的 Lemma 10.4 可以推知，以下是其証明

Sorting problem: 給定一組 number S，我們需要將其排序

Finding the median: 給定一組 number S，我們需要找到其中的中位數

中位數是有序序列中的中間值。

如果序列的長度是奇數，中位數就是排序後序列的正中間的元素。如果序列的

長度是偶數，中位數是排序後序列中間兩個元素，或平均值(視定義而定)

Sorting:已知將集合 S 排序，用 quick sort 時間複雜度為 O(nlogn)

Median:可以根據下次的 key 給予來計算目前的中位數，如果有 n 個 keys，我

們須計算 n 個中位數。

令 P1 為 sorting problem，其 sequence of keys K = k1,k2,⋯,kn

令 P2 為 Finding the median，其中 The instance of P2 應該使得每一個 key

of K 是 on-line median.

所以 K 的最小 key 應該被 output 在次小 key 之前輸出，次小 key 在第三小

key 之前輸出，依此類推。

通過擴展序列 K 並加上-∞和+∞如下所創建這種 P2 的 instance:

-∞,-∞,...,-∞,k1,k2,...,kn,∞,∞,...,∞

輸入包含 n 個-∞，然後是 K，然後是 2n 個∞。

解決 P2 的方法就是在 O(n)時間內將其 reduction 為 P1，

該解決方案由 4n 個 Medians 組成，當第一個∞輸入時，給定 key 的 medians 就是最小 key；當第三個∞輸入時，給定 key 的 medians 就是第二大的 key⋯等，換句話說 K 的排序順序就是 2n+1st median, 2n+3rd median,⋯,4n-1st median，被任何用 P2 算法給輸出，因此 P1 可以解 P2，因此 P1 reduces to P2 在 O(n)時間，故 P1$\leq_d$P2

若 S(n)是排序 n 個 Key 所需要的時間，M(m)是 P2 的解決時間，其共有 m 個 keys，以下是其 reduction

S(n) ≤ M(4n) + O(n)，我們知道 S(n) ≥ cnlogn for some constant c

因此 M(4n) ≥ cnlogn -O(n) => M(n) $\geq c\frac{n}{4}log\frac{n}{4}$-O($\frac{n}{4}$)

所以 M(n) = Ω(nlogn)

以下是一個 Finding Median 的數值例子:

n = 7，first key = 7，所以 median 為 7

第二個 key 輸入為 15，所以 median 是 7 或 15，

然後依序輸入 5 個 keys，3,17,8,11,5

則 medians 的輸出依序為 7, 7 or 15, 8, 8 or 11,8

(b) Distinct values: Determine the number of distinct values in a set of numbers.

Sol:

如果我們能夠有效地解決排序問題，我們也能有效地將 Distinct values 也可以

reduct to sorting，

Sorting problem: 給定一組 number S，我們需要將其排序

Distinct values: 決定一組 number S，S 中不同數值的數量

1. 先將 S 中的數字做排序，使得同樣的值會在一塊。
2. 從排序後的數字中，用一個 count 做計算，用來記錄不同值的個數，將

   count 初始化為 1，因為至少存在一個不同的值。
3. 如果相同則不變，不同則 count+1
4. 最後 count 的值就是不同值的個數。

令 P1 為 Sorting，已知將集合 S 排序，用 quick sort 時間複雜度為 O(nlogn)

令 P2 為 Distinct values

下面是一個有效的演算法:

```
Quick.sort(a);

Int count = 1;// Assume a.length > 0

for (int i = 1; I < a.length; i++)

   if (a[i].compareTo(a[i-1]) != 0)

      count++;
```

代表說可以將 Distinct value reduct to sorting(這邊用 Quick sort)，且 count

的方式只要 O(n)，

因此 P1 reduces to P2 在 O(n)時間，故 P1$\leq_d$P2

以下是一個 Distinct value 的數值例子:

S = [3,1,2,3,4,2,1,5]

作排列=> S_sorted = [1,1,2,2,3,3,4,5]

初值 count = 1

S_sorted[1] = 1 與 S_sorted[0] = 1 相同 所以 count 不變

S_sorted[2] = 2 與 S_sorted[1] = 1 不同 count = 2

S_sorted[3] = 2 與 S_sorted[2] = 2 相同 所以 count 不變

S_sorted[4] = 3 與 S_sorted[3] = 2 不同 count = 3

S_sorted[5] = 3 與 S_sorted[4] = 3 相同 所以 count 不變

S_sorted[6] = 4 與 S_sorted[5] = 3 不同 count = 4

S_sorted[7] = 5 與 S_sorted[6] = 4 不同 count = 5

所以 Distinct value 為 5。

2. Show that the $\leq_P$ relation is a transitive relation on languages. That is, show that if $L_1 \leq_P L_2$ and $L_2 \leq_P L_3$ then $L_1 \leq_P L_3$.

Sol:

令 $L_1 \leq_p L_2$

f_1 是 polynomial time reduction function

such that $x \in L_1 \leftrightarrow f_1(x) \in L_2$

同理

令 $L_2 \leq_p L_3$

f_2 是 polynomial time reduction function

such that $x \in L_2 \leftrightarrow f_2(x) \in L_3$

$\because$ $\underset{\text{合成}}{f_2 \circ f_1}$ 會在 polynomial time

且 $x \in L_1 \leftrightarrow f_1(x) \in L_2 \leftrightarrow f_2(f_1(x)) \in L_3$

$\therefore$ $x \in L_1 \leftrightarrow f_2(f_1(x)) \in L_3$

$L_1 \leq_p L_3$

$\therefore$ $\leq_p$ relation is transitive

3. The set-partition problem takes as input a set S of numbers. The question is whether the numbers can be partitioned into two sets A and $\bar{A} = S - A$ such that $\sum_{x \in A} x = \sum_{x \in \bar{A}} x$. Show that the set -partition problem is NP-complete.

Sol:

用Subset sum problem來解這個problem

令S and t是 set跟target from our subset sum problem

使 $x = \sum_{s \in S} s$ ，設 $S' = \{x - 2t\} \cup S$

此時 S' 的element和為 $2x-2t$

將 S' 帶入至 set-partition problem

若 S' 在set-partition problem有解，則存在1個 S' 的subset A，

A之element和為 $x-t$

若A的element也包含在S，則 ∵ S的element和為 x，

∴ S-A之element和即為t

若 A之element 包含 $x-2t$，則 $A-\{x-2t\}$ 之element和為 $x-t-(x-2t)=t$

且element 均包含在S。

因此原本subset sum problem也有解

若 S' 在set-partition problem無解，則不存在1個 S' 之subset滿足element和為 $x-t$

假設此時 S在 subset sum problem有解，則存在1個S之subset B，

其element和為t。

但此時 $t+(x-2t)=x-t$，set-partition problem亦有解

與先前假設 S' 在set-partition problem 無解矛盾 $\rightarrow\leftarrow$

∴ 此時 subset sum problem亦無解。

∴ subset sum problem $\le_p$ set-partition problem

故 set-partition為 NP-complete

文字版:

Subset Sum Problem: 給定一個集合 S 和一個目標值 t，我們需要確定是否存在 S 的一個子集，其元素和恰好為 t。

Set Partition Problem: 給定一個集合 S’ 我們需要確定是否可以將其劃分成兩個子集，使得這兩個子集的元素和相等。

Pf:

1.假設我們有一個子集和問題(S,t)，我們將其轉化為一個集合劃分問題。

構建集合 S’：定義集合S的元素和為x，即$\mathrm{x} = \sum_{s \in S} s$。

我們構建一個新集合 S’如下：

S’=$S \cup \{x-2t\}$

集合 S’的元素和為 $2x-2t$

2. 如果 S’在 Set Partition Problem 中有解，則存在一個子集$A \subseteq S$

使得$\sum_{a \in A} a = \sum_{a \in S'/A} a$，即$\sum_{a \in A} a$=(2x-2t)/2=x-t

如果 A 不包含 x−2t，則 A 完全包含在 S 中，並且其元素和為 x−t，

那麼 S−A 的元素和為 t。

如果 A 包含 x−2t，我們考慮 A’=A∖{x−2t}。

則 $\sum_{a \in A'} a = x - t - (x - 2t) = t$，且 A’⊆S，

說明 S 中存在一個子集 A’，其元素和為 t。

所以 S’在 Set Partition Problem 有解，S 在 Set Partition Problem 中也有解

3.如果 S’在 Set Partition Problem 無解，則不存在一個子集 A⊆S’，

使得$\sum_{a\in A} a = -t$

假設 S 在 Subset Sum Problem 中也有解，則存在 B⊆S，使得$\sum_{b\in B} b = t$

但是這意味著(S-B) ∪{x-2t}的元素和為 x-t，從而在 Set Partition Problem 也

有解，這與假設 S’在 Set Partition Problem 無解矛盾，

因此 S 在 Subset Sum Problem 也無解。

4.所以我們將 subset sum problem 在多項式時間內 reduce 至 Set Partition

Problem，因為已知 subset sum problem 是 NP-complete 問題，所以 Set

Partition Problem 也是 NP-complete 問題

4. The 3-color problem is that we are given an undirected graph $G = (V, E)$, determine whether G can be colored with three colors or not. Show that the 3-color problem is NP-complete.

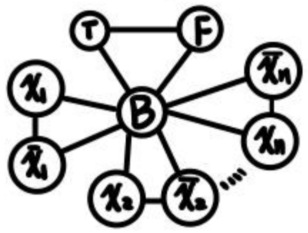
Sol:

利用3-SAT來解3-color problem

使 $x_1, x_2, \ldots, x_n$ 是3-SAT problem instance中的variables

讓 $C_1, C_2, \ldots, C_m$ 是3-SAT problem instance中的clauses

first 建1個graph as follow:



⇒ node T代表true value

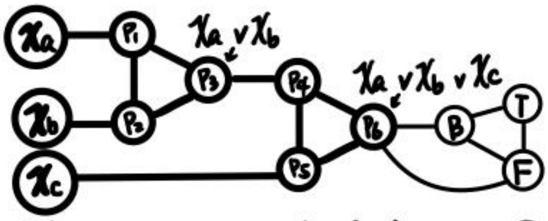
node F代表false value

For each x_i, $1 \le i \le n$, either $x_i = T \vee \bar{x}_i = T$

然後 for each clause C_j, $1 \le j \le m$,

建1個graph as follow:

假設 $C_j = x_a \vee x_b \vee x_c$ $(1 \le a, b, c \le n)$



If x_a, x_b, x_c 都false, 則 P_1, P_2 是 B and T,

P_3 can only 是F,

且∵ P_3 跟 x_c 是false, following the same inference

P_6 也是F

If 至少 x_a, x_b, x_c 之1個為true, 則 always exists

1個3-coloring such that P_6 為T

By constructing the graph for each clause,

我們可在polynomial time將3-SAT轉為3-color problem

∵ 3-SAT是NPC

∴ 3-color problem也是NPC