1. Tug of War

There are 2n persons whose individual weight is an integer.

1. Design an algorithm to decide whether they can be divided into two teams with equal weight.

想為Knapsack problem, 全部人重量為sum,有兩個 knapsack,

n items , size sum/2 for each knapsack , 每個人value 為 1

weight[n]; //weight of person

k=sum/2

p(n,k); //value of knapsack

void knapsack(n, k)

if(n>0 || k>0)

p(n,k) = max(knapsack(n-1, w-weight[n]) + weight[n] ,knapsack(n-1, w));

else: return 0

result: p(n,k) , sum-p(n,k)

b. Design an algorithm to decide whether they can be divided into two teams with equal weight and each team has exactly n persons.

由上述演算法，利用count 技術可得其中一個knapsack中放進了多少人

void knapsack(n, k)

count ++ //count how many people in the knapsack

if( n != 0 || k != 0)

p(n,k) = max(knapsack(n-1, w-weight[n]) + weight[n] ,knapsack(n-1, w));

else: return 0

result: p(n,k) , sum-p(n,k)

if(n == count)

each team has exactly n persons

2. Exercises 16.1‐3

Not just any greedy approach to the activity‐selection problem produces a maximum‐size set of mutually compatible activities. Give an example to show that the approach of selecting the activity of least duration from among those that are compatible with previously selected activities does not work. Do the same for the approaches of always selecting the compatible activity that overlaps the fewest other remaining activities and always selecting the compatible remaining activity with the earliest start time.

1. Always selecting the compatible activity with least duration.

3

3

5

依照提示選上面的，但下面的為最佳解

1. always selecting the compatible activity that overlaps the fewest other remaining activities

2

3

1

3

1

3

5

3

5

依照提示選第一個，但第二個為最佳解

1. always selecting the compatible remaining activity with the earliest start time.

4

1

3

5

依照提示選上面的，但下面的為最佳解

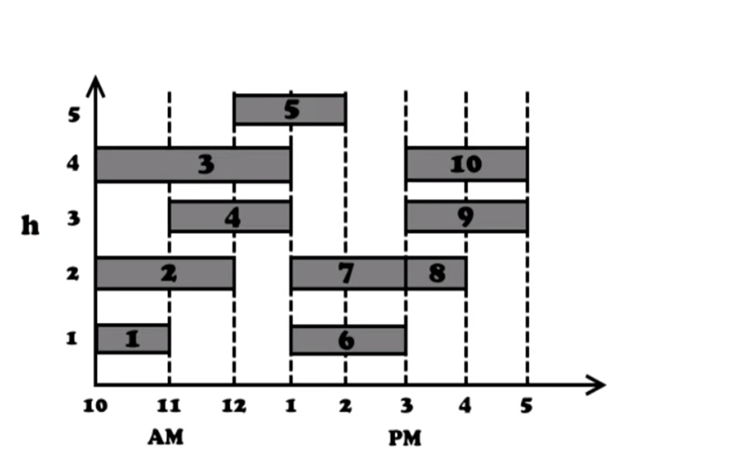
3. Exercises 16.1‐4

Suppose that we have a set of activities to schedule among a large number of lecture halls, where any activity can take place in any lecture hall. We wish to schedule all the activities using as few lecture halls as possible. Give an efficient greedy algorithm to determine which activity should use which lecture hall.(This problem is also known as the interval‐graph coloring problem. We can create an interval graph whose vertices are the given activities and whose edges connect incompatible activities. The smallest number of colors required to color every vertex so that no two adjacent vertices have the same color corresponds to finding the fewest lecture halls needed to schedule all of the given activities.)

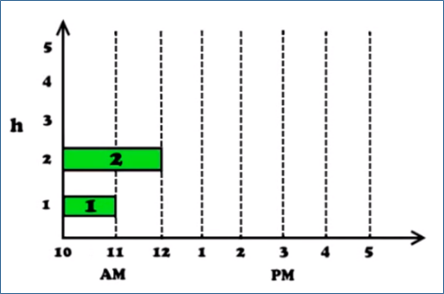
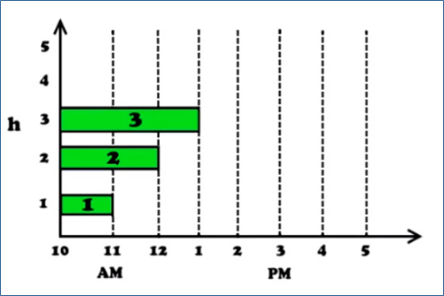
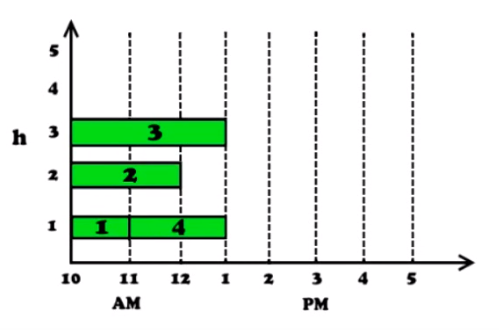
假設有10個activity

第一步 : 將所有activity依照開始時間由小到大做排序

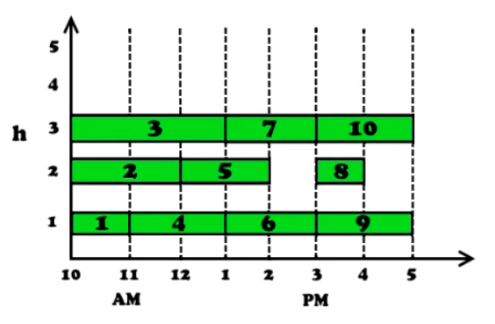
S[1] ≤ S[2] ≤ S[3] ≤ … ≤ S[10]



第二步 : 將 activity 依照 S[i] 放入 lecture halls ，當目前 lecture halls 中 activity 的空間不夠時( 結束時間小於目前所有 activity 的開始時間 )，才分配新的 lecture halls ，由此可使用最少的 lecture halls 。

最後結果



pseudo code

Sort activities by starting time so that s1 ≤ s2 ≤ ... ≤ sn.

d = 1 // lecture halls 數量(最少一個)

for i = 1 to n {

if (activity i is compatible with some lecture hall k) //activity可以放入某個lecture halls

schedule activity i in lecture hall k

else //沒有lecture halls 可以放目前的activity

allocate a new lecture hall d + 1 //分配新的lecture halls

schedule activity i in lecture hall d + 1 //將activity放入新的lecture halls

d = d + 1 //數量加1

}

4.

想法：不使用排序，用價格比重量的中位數決定拿哪些，先拿比中位數大的

m=median

v\_i = value of item i

w\_i = weight of item i

Let *G* = { *i* : *v\_i*​/*w\_i*​ >*m*} (比m大) , W\_G = ∑​w\_i​ in G

E = { i : v\_i / w\_i = m} , W\_E = ∑​w\_i in E

L = {i : v\_i / w\_i <m} (比m小) , W = knapsack capacity

Case1 ： If W\_G > W，就不拿並對集合G做遞迴

Case2 ： If *W\_G*​ ≤ *W* ≤W\_G+W\_E，G全拿並在E拿滿背包

Case3 ： If W\_G​ + W\_E​ < W，就全拿並對集合L做遞迴

Time complexity：*T*(*n*)≤*T*(*n*/2)+Θ(*n*) , hence T(n)=O(n)

5.

Given a 0‐1 knapsack problem with the knapsack size K and n items, where each item has its weight in integer and its value in real.

a. Design an algorithm to find the most valuable load of the items that fit into the knapsack.

b. Design a pseudo‐polynomial time algorithm to determine the optimal solution that the total weight exactly equals to K.

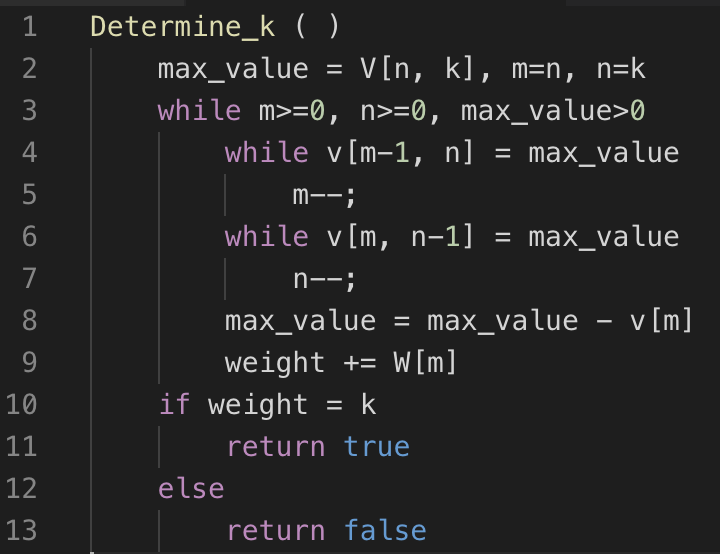
ANS:

1. 直接運用課本的 0-1 背包問題

設初始條件

則遞迴式為

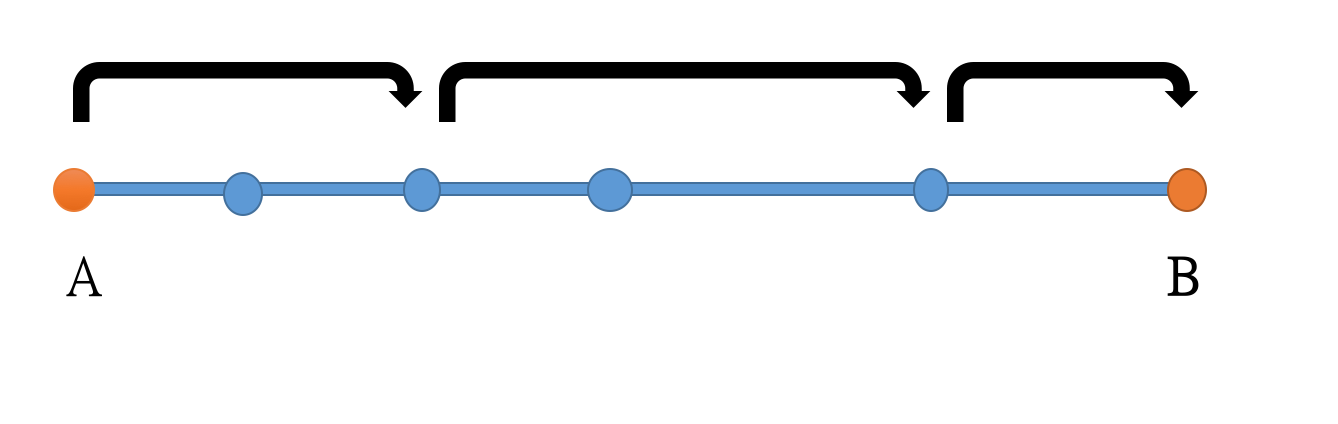
1. 從得到的表格去 trace back



所以其時間花費最多即為 nK (最外圈的 while)，而其 K有可能會大於 因此為 pseudo‐polynomial 的演算法。

6.

題目簡單來說是要問，裝滿水可以走的距離為 m，從A走到B之間有許多加水點可以加水，加水的次數要最少。



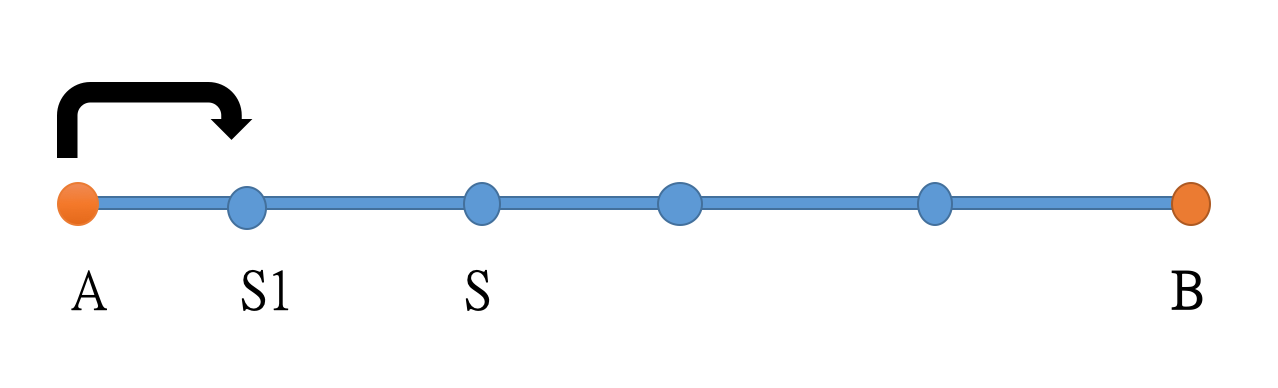
我們使用以下的greedy choice

1.在最遠且可以抵達的加水點加水。

2.每經過一個加水點時，判斷目前的水是否足夠走到下一個加水點。

如果可以，略過這個加水點，如果不行，停下來加水。

Proof

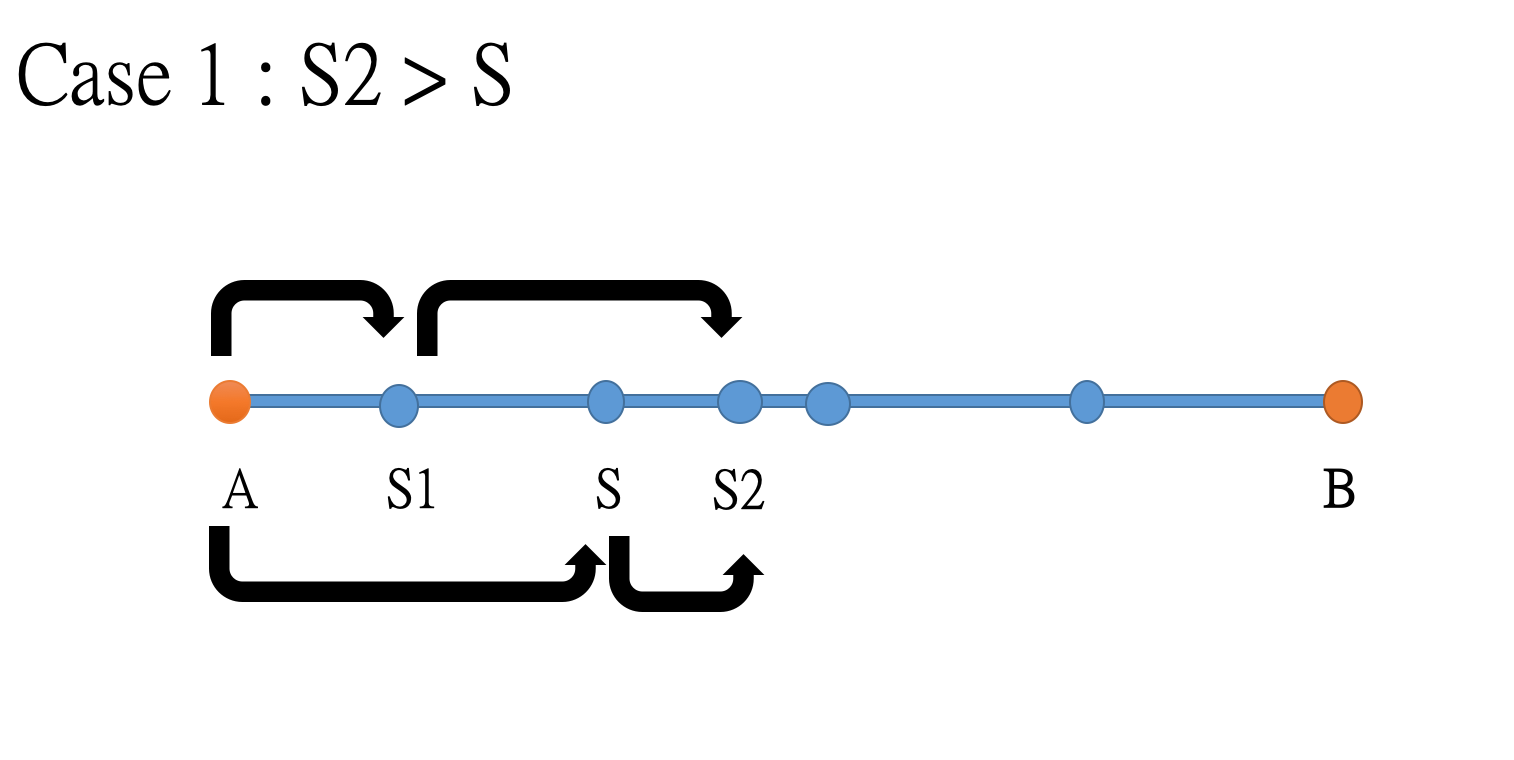


假設Ｓ是照上述步驟的第一個停下來的加水點，S1是在S之前的某個加水點，如果今天選擇先在S1加水的話下次加水點S2可以有三種情況。

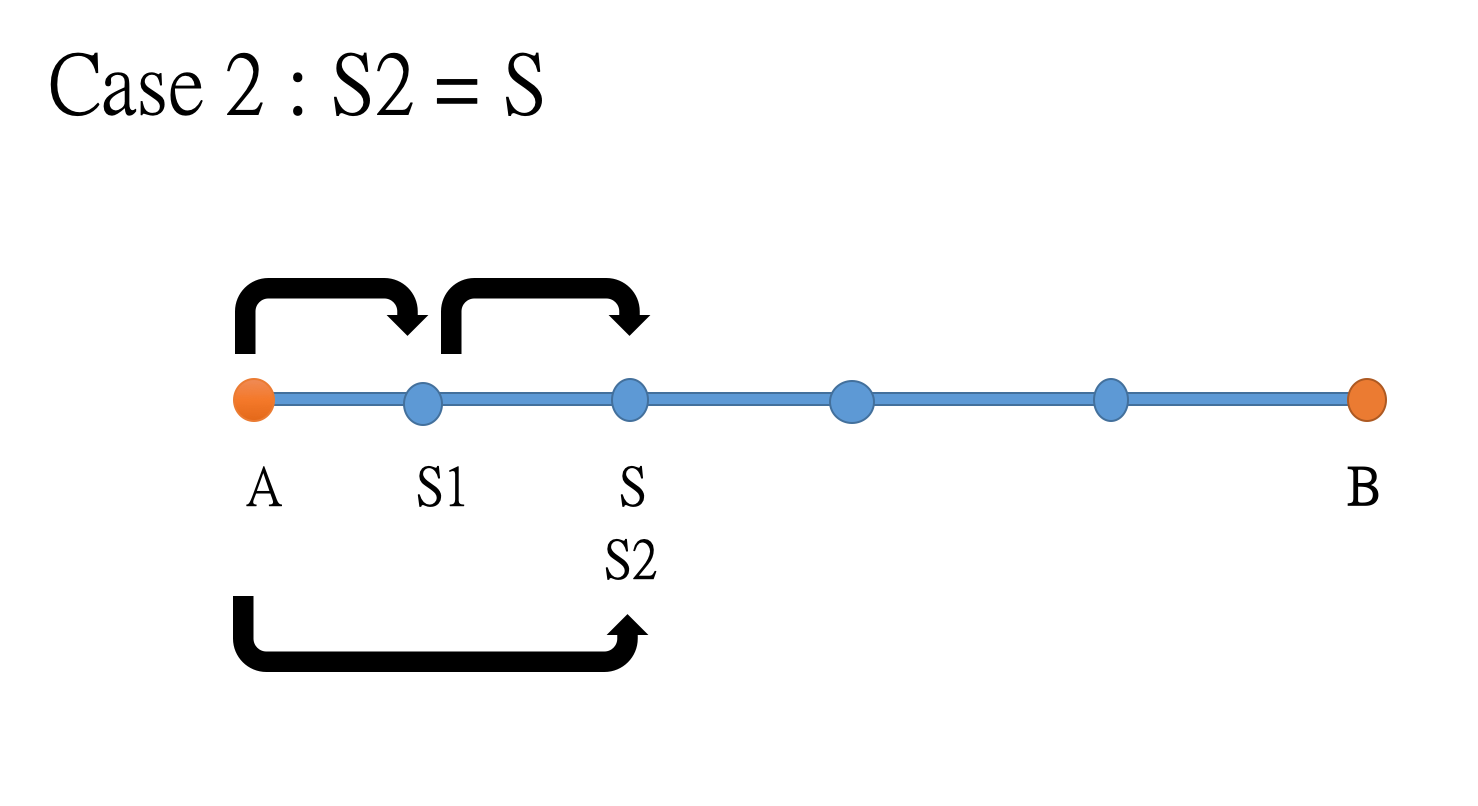
Case 1: S2>S

Case 2: S2=S

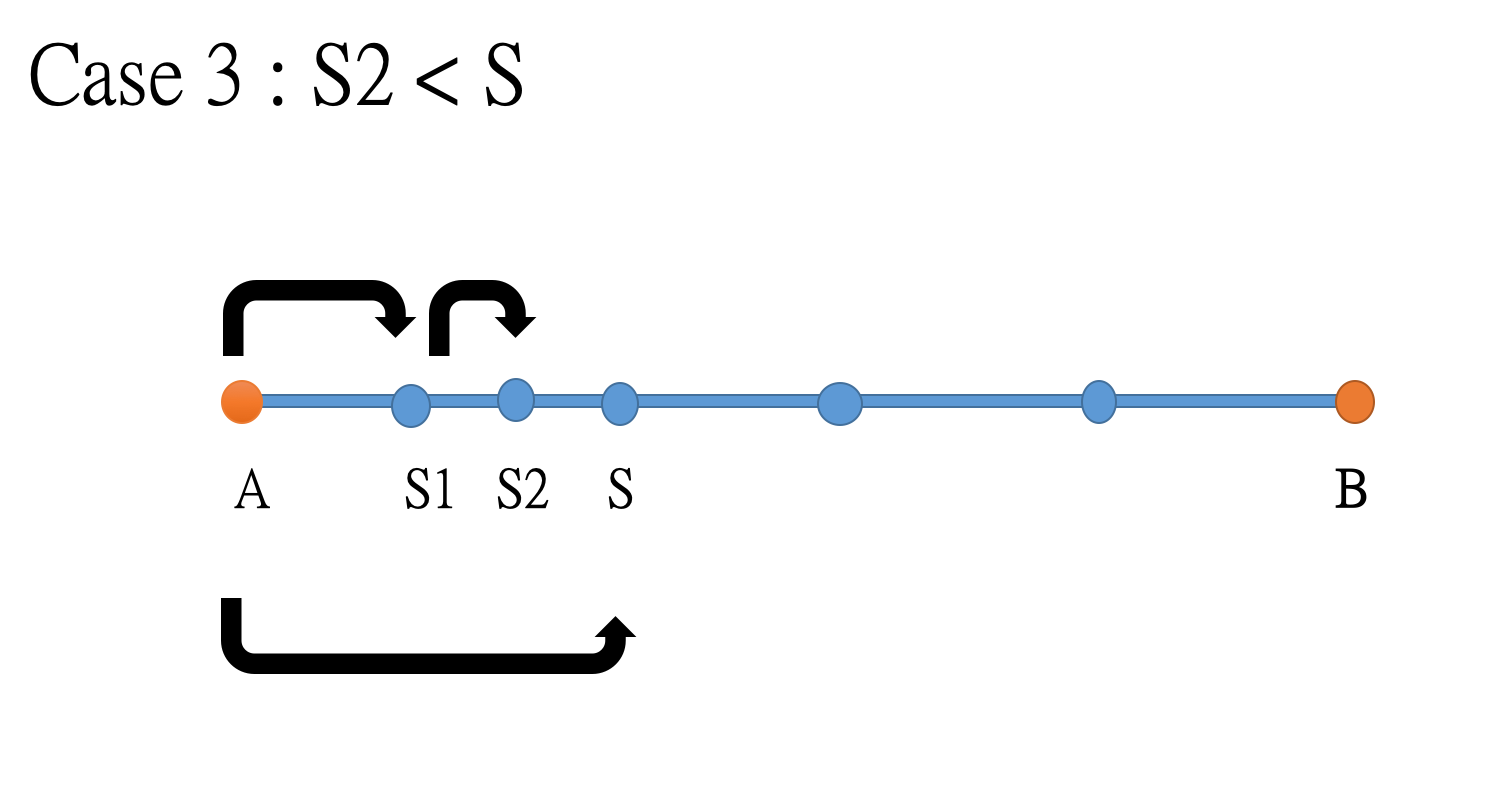
Case 3: S<S2



可以發現加水的次數一樣。



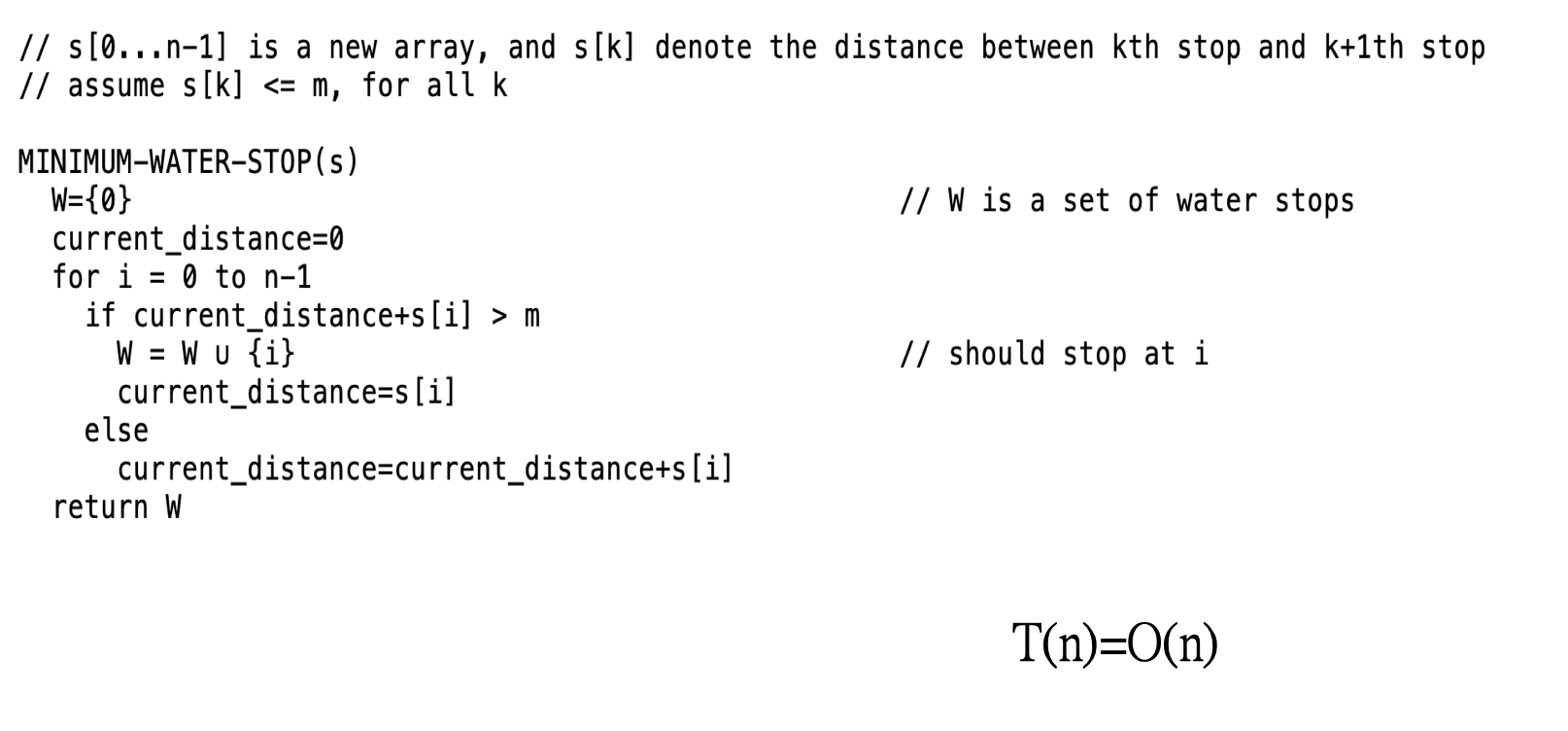
可以發現先在S1加水再到S2加水，還不如直接在S2加水。



可以發現先在S1加水再到S2加水，還不如直接在S加水。

總結以上三個case都不會有更好的解果，因此一個加水點一個加水點證明下去就會有最佳解。

可以參考如下的pseudocode



7.

(1)1Let for I = 1 to n

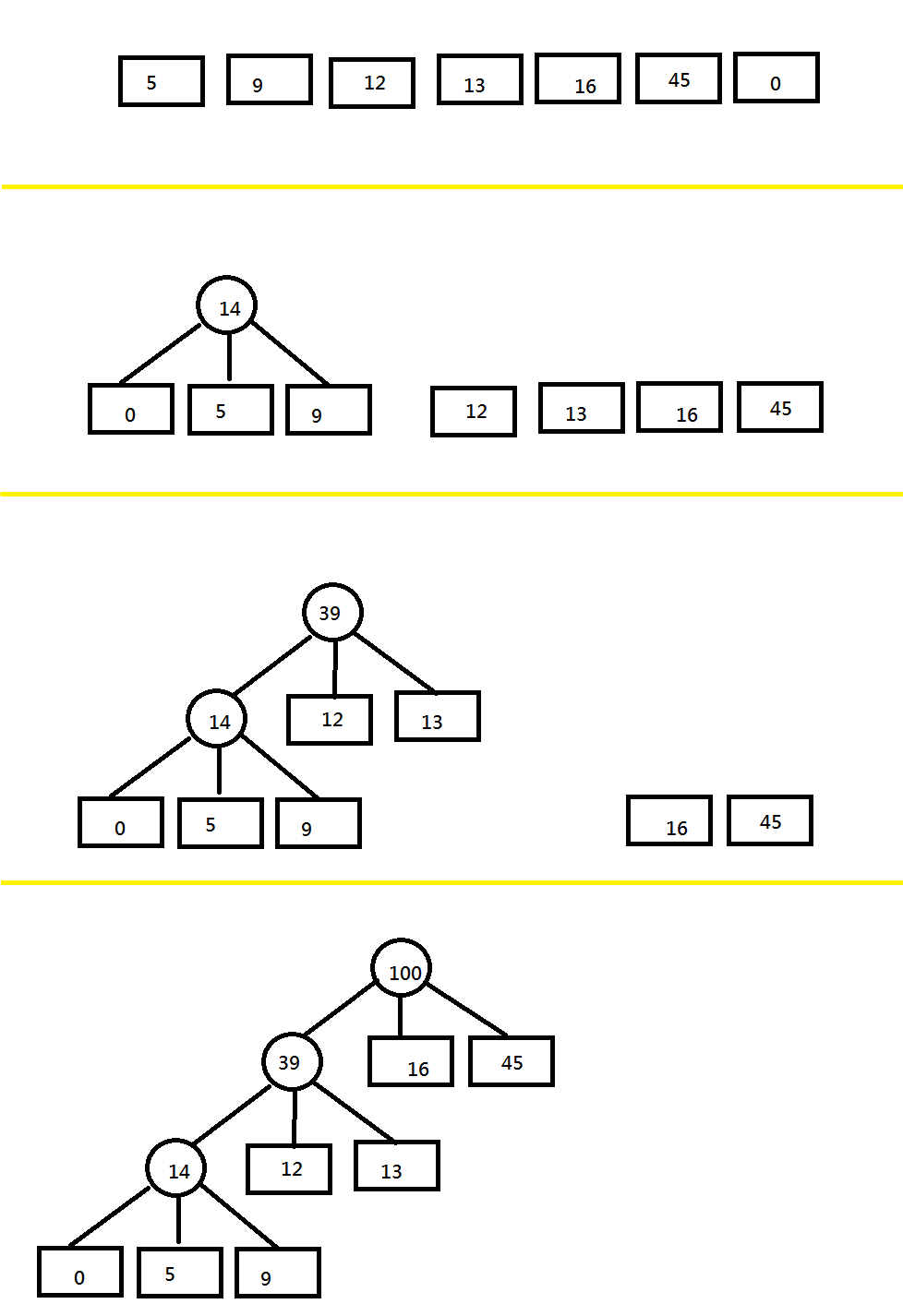
(2)For the set of ,find the median. Time complexity T(n)=

(3)If median S1 , if median S2

(4)Get the object from S1 , median to S2

8.

Generalize Huffman's algorithm to ternary codewords (i.e., codewords using the symbols 0, 1, and 2), and prove that it yields optimal ternary codes.



虛擬碼：

HUFFMAN\_TERNARY(C)

{

IF |C|=EVEN

THEN ADD DUMMY CHARACTER Z WITH FREQUENCY 0.

N=|C|

Q=C; //WE ARE BASICALLY HEAPIFYING THE CHARACTERS

FOR I=1 TO floor(N/2)

{

ALLOCATE NEW\_NODE;

LEFT[NEW\_NODE]= U= EXTRACT\_MIN(Q)

MID[NEW\_NODE] = V= EXTRACT\_MIN(Q)

RIGHT[NEW\_NODE]=W= EXTRACT\_MIN(Q)

F[NEW\_NODE]=F[U]+F[V]+F[W];

INSERT(Q,NEW\_NODE);

}

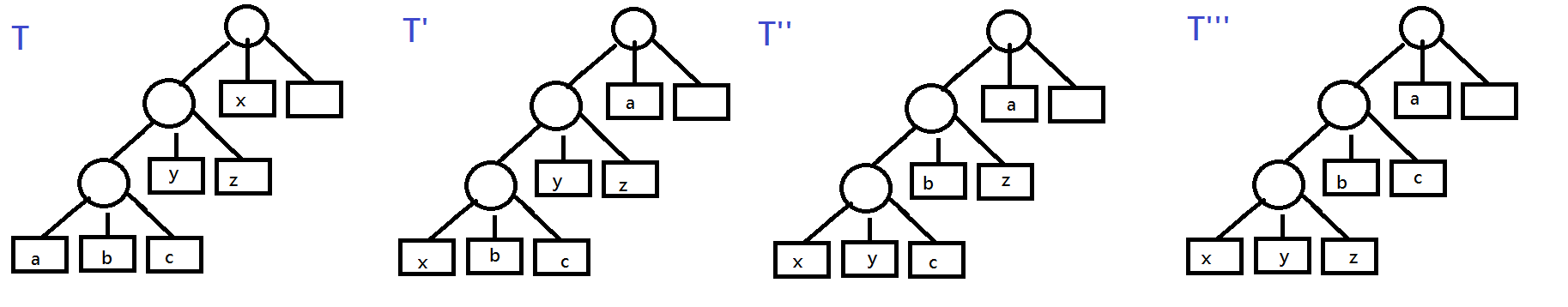
RETURN EXTRACT\_MIN(Q);

} //END-OF-ALGO

證明：

定理1：

C是一個字母集合其中每個字元c屬於C有頻率f[c]。令x,y,z是在C中有最低頻率的字元，則存在一個C的最佳前置碼，在其內x,y,z的字碼有相同的長度，並且只在最後一位元不同。



採用一個任意的最佳前置碼T，並修改它成為另一個最佳前置碼T'''，x,y,z出現在T'''的最大深度且互為siblings。

從T到T'花費的差異：

B(T)-B(T')=Σf(c)-dT(c)- Σf(c)-dT'(c)

=f(x)dT(x)+ f(a)dT(a)- f(x)dT'(x)- f(a)dT'(a)

=f(x)dT(x)+ f(a)dT(a)- f(x)dT(a)- f(a)dT(x)

=( f(a)-f(x) )( dT(a)-dT(x) ) >=0

因為B(T)>= B(T') 且T是最佳的，所以推得B(T)= B(T')，T'也是最佳的。同理，T''和T'''也是最佳的。

此定理代表藉由合併來建立最佳樹的過程具有一般性。而在每個步驟的所有可能合併中，HUFFMAN選擇最低花費的那一個，greedy choice屬性成立。

定理2：

C是一個字母集合其中每個字元c屬於C有頻率f(c)。令x,y,z是在C中有最低頻率的字元。令C'=C-{x,y,z}∪{z1} ，(移除x,y,z並加入新字元z1)。定義C'的f同C，除了f(z1)=f(x)+f(y)+f(z)。令T'代表C'內的任意最佳前置碼；則T是T'將z1以x,y,z取代所得到的，T代表C內的任意最佳前置碼。

B(T)=B(T')+f(x)+f(y)+f(z)，等價於B(T')=B(T)-f(x)-f(y)-f(z)，

使用反證法，假設T並非C內的一個最佳前置碼，則存在某T''，B(T'')<B(T)，由前一定理得知T'有x,y,z互為siblings。在T''中，用一片f(z1)= f(x)+f(y)+f(z)的葉子z1來取代x,y,z的parent得到T'''，則：

B(T''')=B(T'') -f(x)-f(y)-f(z)

<B(T) -f(x)-f(y)-f(z)

=B(T')

矛盾，所以T必須是C內的一個最佳前置碼。

此定理顯示建構最佳前置碼的過程具備optimal-substructure屬性。

總結：由以上兩定理得到HUFFMAN會產生一個最佳前置碼。