Homework 4

林宏陽、周若涓、曾守瑜

Problem1

Consider algorithm *Mapping* (see notes/slides). Is it possible that the set S will become empty at the end of the algorithm? Show an example, or prove that it cannot happen.

Problem1(cont'd)

If A(original set) $= \emptyset$, then $S = \emptyset \subseteq A$ apply the algorithm is still empty.
If $A \neq \emptyset$, suppose S will become empty at the end. Then the previous step is eliminating the last element(called a) in S and $c[a] = 0$.

1. If $f(a) = a$
   It is impossible because $c[a] = 0$.
2. If $f(a) \neq a$
   It is impossible because a is the only one element in S.

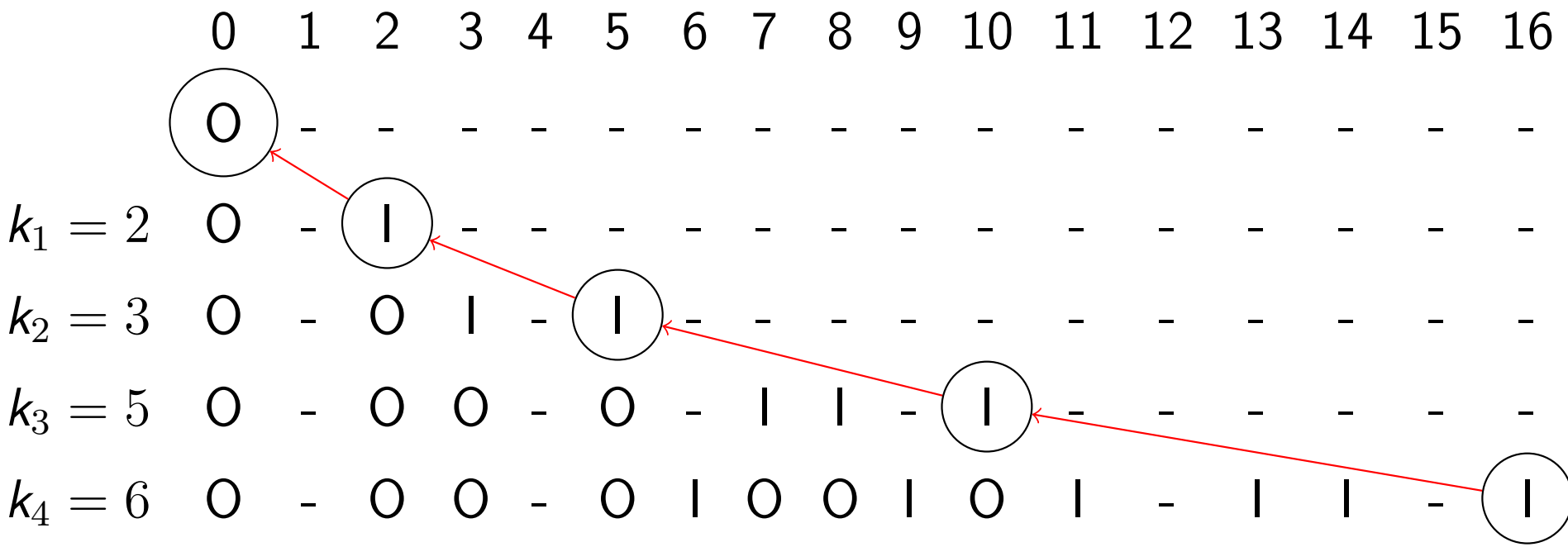
Hence, we conclude that when $A \neq \emptyset$, S will not become empty.

Problem2

依照課堂提到的演算法，我們可以得到一個陣列 P
利用陣列 P 的資訊，寫一個 function 去得到資訊
思路：

- 查看 $P[i, k]$，如果 $exist = true$，查看 $belong$ 的值
- 如果 $belong = true$，代表 $S[i]$ 屬於這組解，下一步查看 $P[i - 1, k - S[i]]$
- 如果 $belong = false$，查看 $P[i - 1, k]$
- 從 $P[n, K]$ 開始看，看到 $P[0, 0]$ 結束

Problem2



$I = [6, 5, 3, 2]$

Problem2

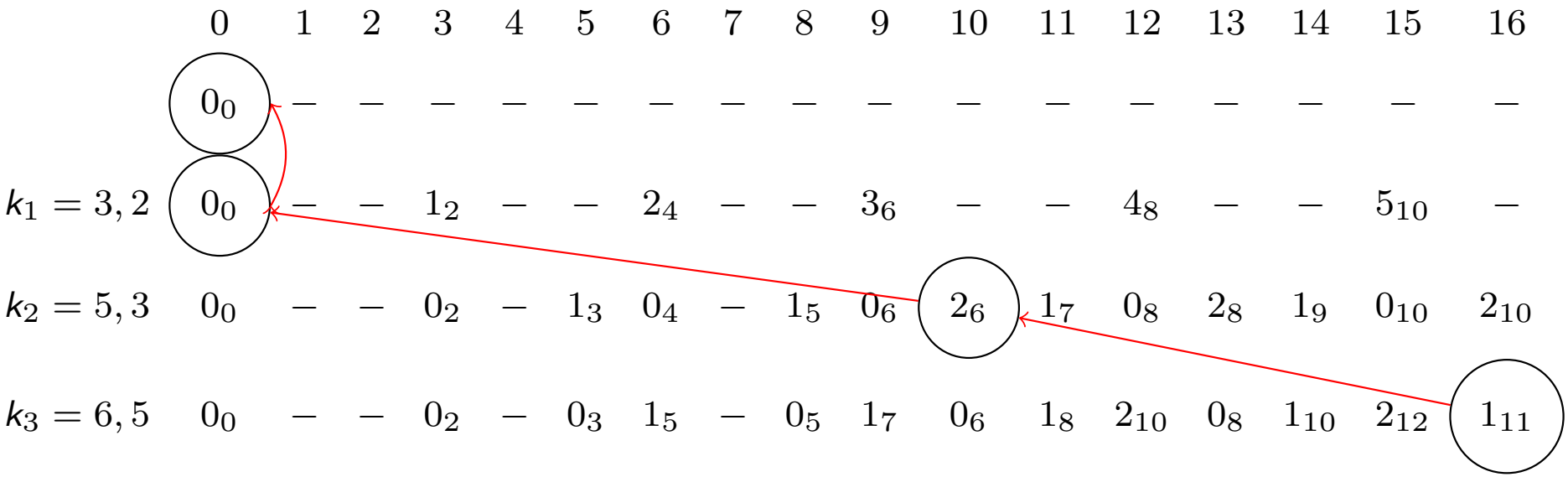
```
function KNAPSACK RECOVER(S, K, P)
    k : int := K, l : list := [];
    for i := n to 1 do
        if P[i, k].exist = false then
            return "impossible";
        if P[i, k].belong = true then
            l.append(S[i]);
            k := k − S[i];
    if k ≠ 0 then
        return "impossible";
    return l;
```

Problem3

思路：

- 一個 item 的單位質量價值越高就越優先考慮
- (optional) 同樣質量的 item，只考慮價值最高者
- 把原本的 belong 看成 0 與 1 的變數，並用非負整數改寫

Problem3


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
0_0 – – – – – – – – – – – – – – – –
$k_1 = 3, 2$ 0_0 – – 1_2 – – 2_4 – – 3_6 – – 4_8 – – 5_{10} –
$k_2 = 5, 3$ 0_0 – – 0_2 – 1_3 0_4 – 1_5 0_6 2_6 1_7 0_8 2_8 1_9 0_{10} 2_{10}
$k_3 = 6, 5$ 0_0 – – 0_2 – 0_3 1_5 – 0_5 1_7 0_6 1_8 2_{10} 0_8 1_{10} 2_{12} 1_{11}

Problem3

```
function KNAPSACK(S, V, K)
    P[0, 0].exist := true;
    for k := 1 to K do
        P[0, k].exist := false;
    for i := 1 to n do
        for k := 0 to K do
            P[i, k].exist := false;
            if P[i − 1, k].exist then
                P[i, k].exist := true;
                P[i, k].amount := 0;
                P[i, k].value := P[i − 1, k].value;
            if k ≥ S[i] and P[i, k − S[i]].exist and
P[i, k − S[i]].value + V[i] > P[i, k].value then
                P[i, k].exist := true;
                P[i, k].amount := P[i, k − S[i]].amount + 1;
                P[i, k].value := P[i, k − S[i]].value + V[i];
```

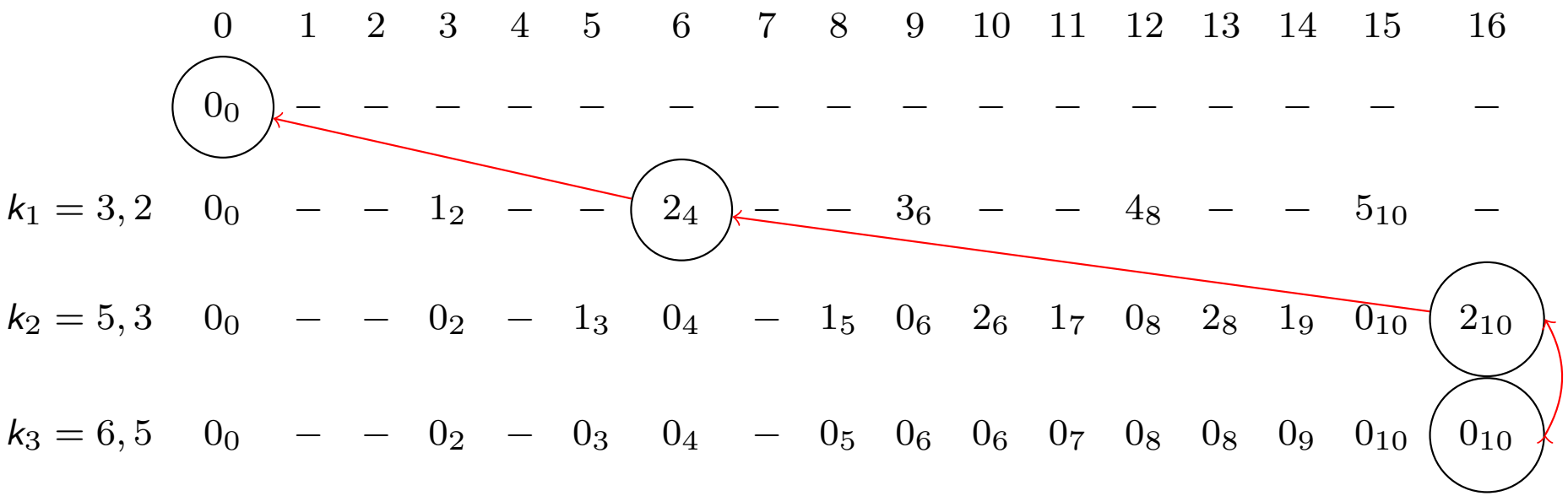
Problem3

常錯的點：沒有考慮 exist
題目所求：裝滿，若小於 K 則裝不滿，若大於 K 則裝不下，所以是正好為 K

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	0_0	0_0	0_0	0_0	0_0	0_0	0_0	0_0	0_0	0_0	0_0	0_0	0_0	0_0	0_0	0_0	0_0
$k_1 = 3, 2$	0_0	0_0	0_0	1_2	1_2	1_2	2_4	2_4	2_4	3_6	3_6	2_6	4_8	4_8	4_8	5_{10}	5_{10}
$k_2 = 5, 3$	0_0	0_0	0_0	0_2	0_2	1_3	0_4	0_4	1_5	0_6	0_6	1_7	0_8	0_8	1_9	0_{10}	0_{10}
$k_3 = 6, 5$	0_0	0_0	0_0	0_2	0_2	0_3	1_5	1_5	0_5	1_7	1_7	1_8	2_{10}	1_{10}	1_{10}	1_{12}	1_{12}

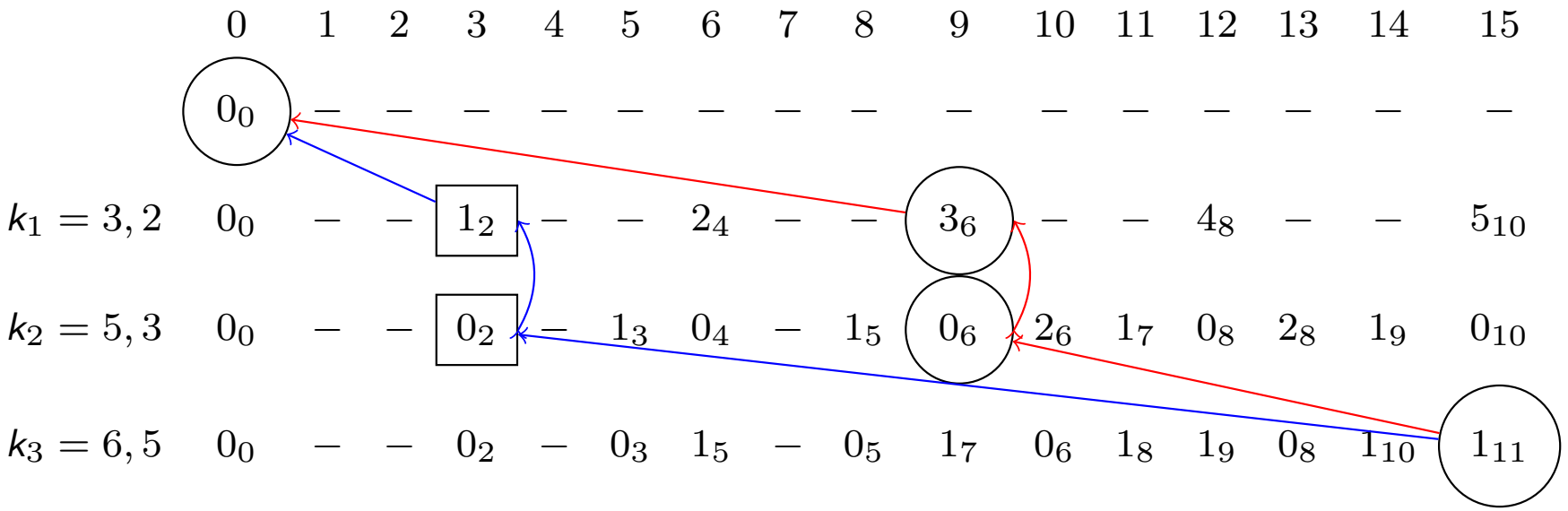
Problem3

常錯的點：沒有正確 update 更佳解之 1
只要 exist 就直接繼承而不檢查 update (else if)
若把 item 按照單位質量價值排序？



Problem3

常錯的點：沒有正確 update 更佳解之 2
看向 $P[i-1, k-S[i]]$ 而非 $P[i, k-S[i]]$
理論上你需要從 $k-S[i]$ 到 $k-j\times S[i]$ 全看過，才決定要塞 j 份進去



演算法算出的路徑（紅色）與更佳塞法的路徑（藍色）

Problem3

解答並沒有實作剛剛的第一二條思路（照單位質量價值排序）
不照順序排，每個直排都需要持續 update 總價值
若事先排序，仍然需要 update：

	0	100	109	111
	0_0	$-$	$-$	$-$
$k_1 = 100, 10000$	0_0	1_{10000}	$-$	$-$
$k_2 = 3, 3$	0_0	0_{10000}	3_{10009}	37_{111}
$k_3 = 11, 1$	0_0	0_{10000}	0_{10009}	1_{10001}

Problem4

Let $x_1, x_2, ..., x_n$ be a set of non-negative integers, and let $S = \sum_{i=1}^{n} x_i$. Design an algorithm to partition the set into two subsets of equal sum, or determine that it is impossible to do so. The algorithm (presented in suitable pseudocode) should run in time $O(nS)$.

Problem4(cont'd)

function ALGORITHM PARTITION(X)
    if S is odd **then**
        print "impossible";
    else
        $halfS := \frac{S}{2}$;
        $Knapsack(n, halfS)$;
        if $P[n, halfS].exist$ **then**
            add elements found by Problem2's algorithm to set1
            add the rest to set2
        else
            print "impossible";

Question5(a)

```
Algorithm Hanoi(n, A, B, C);
1 begin
2    if n=1 then
3        move from A to B
4    else if n>1
5        Hanoi(n-1, A, C, B);
6        move the largest disk from A to B
7        Hanoi(n-1, C, B, A);
8 end
```

[Base case] 1 disk (move from A to B)

[Inductive step]**move n disks**

[Induction hypothesis]Moving **n-1** disks is available

1. move n-1 disks from A to C (induction hypothesis)
2. move the largest disk to B (base case)
3. move n-1 disk from C to B (induction hypothesis)

Question5(b)

假設 M(n) 為搬動次數

$$\begin{cases} M(1) = 1 \\ M(n) = M(n-1) + 1 + M(n-1) = 2 \cdot M(n-1) + 1 \end{cases}$$

M(n)=2M(n-1)+1

$=2[2M(n-2)+1]+1$

$=2^2 \cdot M(n-2)+(2+1)$

$=2^2 \cdot [2M(n-3)+1]+(2+1)$

$=2^3 \cdot M(n-3)+(2^2+2+1)$

$\vdots$

$=2^i \cdot M(n-i)+(1+2+2^2+\ldots+2^{i-1})$

$=2^i \cdot M(n-i)+(\frac{2^i-1}{2-1})$

$\because M(1)=1 \therefore$ i 以 (n-1) 代入

$=2^{n-1} \cdot M(1)+2^{n-1}-1$

$=2^n-1$