Algorithm 2017 Spring
Homework 3 Solutions

指導教授：謝孫源 教授

助教：許景添 陳琮皓 林玉陞 何岱璇

1. The Knapsack problem:

3 items , the capacity is 8

Profits: (p1, p2, p3) = (8, 6, 3) , Weights: (w1, w2, w3) = (6, 5, 3)

(a) (5pts) fraction, greedy

- 因為$\frac{P_1}{W_1} = \frac{4}{3}$, $\frac{P_2}{W_2} = \frac{6}{5}$, $\frac{P_3}{W_3} = 1$
- 取物順序 item1 -> item2 -> item3
- 先拿item1：因為$W_1 = 6$ 可全拿
- 再拿item2：因為$W_2 = 5$ 只能拿 2 kg
- 總共獲利為：$8 + 6 \times \frac{2}{5} = 10.4$

(b) (5pts) 0/1 knapsack problem, dynamic programming

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	8	8	8
0	0	0	0	0	6	8	8	8
0	0	0	3	3	6	8	8	9

2. The Knapsack problem:

(a) (4pts)If the size of each object is arbitrary real number, does the dynamic programming method still work?

No

(b) (6pts)Explain your answer.

若W[i]為實數，則有可能使得k-W[i]不為整數

3.

m

i \ j	1	2	3	4
1	0	30	66	102
2		0	60	80
3			0	48
4				0

Cost = 102

s

i \ j	2	3	4
1	1	2	2
2		2	2
3			3

$(A_1A_2)(A_3A_4)$

$$m[1,2] = m[1,1] + m[2,2] + p_0p_1p_2 = 30 \quad ,k = 1$$
$$m[2,3] = m[2,2] + m[3,3] + p_1p_2p_3 = 60 \quad ,k = 2$$
$$m[3,4] = m[3,3] + m[4,4] + p_2p_3p_4 = 48 \quad ,k = 3$$

$$m[1,3] = \min\begin{cases} m[1,1] + m[2,3] + p_0p_1p_3 = 150 & ,k = 1 \\ m[1,2] + m[3,3] + p_0p_2p_3 = 66 & ,k = 2 \end{cases}$$

$$m[2,4] = \min\begin{cases} m[2,2] + m[3,4] + p_1p_2p_4 = 88 & ,k = 2 \\ m[2,2] + m[4,4] + p_1p_3p_4 = 180 & ,k = 3 \end{cases}$$

$$m[1,4] = \min\begin{cases} m[1,1] + m[2,4] + p_0p_1p_4 = 148 & ,k = 1 \\ m[1,2] + m[3,4] + p_0p_2p_4 = 102 & ,k = 2 \\ m[1,3] + m[4,4] + p_0p_3p_4 = 138 & ,k = 2 \end{cases}$$

4.

	j	0	1	2	3	4	5	6
i		y_j	*B*	*D*	*C*	*A*	*B*	*A*
0	x_i	0	0	0	0	0	0	0
1	*A*	0	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	*B*	0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	*C*	0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
4	*B*	0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3
5	*D*	0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	*A*	0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	*B*	0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4

4.

沒箭頭，沒字母扣三分

重大錯誤扣五分

5.Determine an LCS of s1, s2, s3

Sol.

- LCS: 出現於每一個序列、而且是最長的子序列，可能有許多個。
- Ans.(a) <1, 0, 0, 1, 1, 0>
- Ans.(b) <c, e, a>, <d, e, a>

6.Find the maximal revenue obtainable with the prices below.

Length i	1	2	3	4	5	6	7	8
Price pi	1	5	8	9	10	17	17	20

Sol.

- (a) $r_4 = \max(p_1 + r_3, p_2 + r_2, p_3 + r_1, p_4 + r_0)$
  $= \max(1 + 8, 5 + 5, 8 + 1, 9 + 0)$
  $= \max(9, 10, 9, 9)$
  $= 10$
- (b) $r_5 = \max(p_1 + r_4, p_2 + r_3, p_3 + r_2, p_4 + r_1, p_5 + r_0)$
  $= \max(1 + 10, 5 + 8, 8 + 5, 9 + 1, 10 + 0)$
  $= \max(11, 13, 13, 10, 10)$
  $= 13$

7.

Q: Give a dynamic-programming solution to the 0-1 knapsack problem that runs in $O(nW)$ time, where n is the number of items and W is the maximum weight of items that the thief can put in his knapsack.

Sol.

- Let K(w) denote the max. value of a knapsack of capacity w to be gilled with items of weights $w_1, w_2, \dots, w_n$ and values $v_1, v_2, \dots, v_n$ so that the total weight on the items in the knapsack does not exceed w.
- If an optimal packing involves an item I, then removing this item leads to an optimal packing of a knapsack of capacity $K(w - w_i)$. Hence,

$$K(w) = K(w - w_i) + v_i$$

- for some i. To find the i we try all possibilities, which leads to a recursion

$$K(w) = \max_{i:w_i \le w} \{K(w - w_i) + v_i\}$$

- with the initial condition $K(0) = 0$.

7.

Q: Give a dynamic-programming solution to the 0-1 knapsack problem that runs in $O(nW)$ time, where n is the number of items and W is the maximum weight of items that the thief can put in his knapsack.

Sol.法一

Algorithm 2 KNAPSACK 2

```
K(0) = 0                                   // knapsack value
S = ∅                                      // knapsack content (a multiset)
for (w = 1; w ≤ W; w++) do
    K(w) = K(w − 1)
    ind = 0                                // index for which the max is attained
    for (i = 1; i ≤ n; i++) do             // computing the max in the recursion
        if (w_i ≤ w && K(w) > K(w − w_i) + v_i) then
            ind = i
    if (ind > 0) then                      // if an improvement of K(w) is possible, do it
        K(w) = K(w − w_ind) + v_ind
        S = S ∪ {ind}
return K(W) and S
```

7.

Q: Give a dynamic-programming solution to the 0-1 knapsack problem that runs in $O(nW)$ time, where n is the number of items and W is the maximum weight of items that the thief can put in his knapsack.

Sol.法二

- Define **c[i,w] to be the value of the solution for items 1~i and maximum weight w**
- $$\begin{cases} 0 & \text{if } i = 0 \text{ or } w = 0 \\ c[i-1, w] & \text{if } w_i > w \\ \max(v_i + c[i-1, w-w_i], c[i-1, w] & \text{if } i > 0 \text{ and } w \geq w_i \end{cases}$$

```
Dynamic-0-1-Knapsack(v,w,n,W)
Let c[0..n,0..W] be a new array
For w = 0 to W
  c[0,w] = 0
For i = 1 to n        -O(n)
  c[i,0] = 0
  for w=1 to W        -O(w)
    if w_i ≤ w
      if v_i + c[i-1,w- w_i]>c[i-1,w]
        c[i,w] = v_i + c[i − 1, w − w_i]
      else c[i,w] = c[i-1,w]
    else c[i,w] = c[i-1,w]
```

7.

Q: Give a dynamic-programming solution to the 0-1 knapsack problem that runs in $O(nW)$ time, where n is the number of items and W is the maximum weight of items that the thief can put in his knapsack.

Sol.

- Time complexity : $O(nW)$
- 配分方式 (10%)
- Codes : 7 pts
- Time complexity analyze : 3 pts

8.

Q: Show, by means of a counterexample, that the following "greedy" strategy does not always determine an optimal way to cut rods. Define the density of a rod of length i to be p_i, that is, its value per inch. The greedy strategy for a rod of length n cuts off a first piece of length i, where $1 \leq i \leq n$, having maximum density. It then continues by applying the greedy strategy to the remaining piece of length $n - i$

Sol.

length i	1	2	3	4
price p_i	1	20	33	36
p_i/i	1	10	11	9

- Here is a counterexample for the "greedy" strategy:
- Let the given rod length be 4.
- According to a greedy strategy, we first cut out a rod of length 3 for a price of 33, which leaves us with a rod of length 1 of price 1.
- The total price for the rod is 34. The optimal way is to cut it into two rods of length 2 each fetching us 40 dollars.

8.

Q: Show, by means of a counterexample, that the following "greedy" strategy does not always determine an optimal way to cut rods. Define the density of a rod of length i to be p_i, that is, its value per inch. The greedy strategy for a rod of length n cuts off a first piece of length i, where $1 \leq i \leq n$, having maximum density. It then continues by applying the greedy strategy to the remaining piece of length $n - i$

Sol.

- 配分方式 (10%)
- 題目有開頭表示使用反舉立法，不是用舉反例不給予分數

9.Consider the following statements, answer true or false and explain why.

(a) Dynamic programming always provides polynomial time algorithms.

false

(b) Dynamic programming uses tables to design algorithm.

true

(c) Optimal substructure is an important element of Dynamic programming algorithm.

true

(d) The single source shortest path problem has the property of optimal substructure.

true

(e) In the Knapsack problem, if the size of each object is arbitrary real number, the Dynamic programming method still work.

false

10. What is an optimal Huffman code for the following set of frequencies, based on the first 8 Fibonacci number.
a:1 b:1 c:2 d:3 e:5 f:8 f:13 h:21

---

a: 1111111

b: 1111110

c: 111110

d: 11110

e: 1110

f: 110

g: 10

h: 0