Algorithms 2020 Spring HW3

Author: 90899201Y tony20715 黃悟淳(選讀)

1. Warm-Up

Knapsack

1.
    - Let capacity be the x-axis and the number of capacity be i, and the item be the y-axis and j. The weight of the j-th item is $w(j)$, and the value of the j-th item is $v(j)$. The total-value table is denoted by V, and $V[i,j]$ refers to the maximum value one can obtain if one' s knapsack has i capacity and one can pick items among the first j items in the list.
    - The recursive relation is
        - $V[i,j] = V[i,j-1]$, provided that $w(j) > i$.
        - $V[i,j] = max\{V[i,j-1], V[i-w(j), j-1] + v(j)\}$, provided that $w(j) \leq i$.
    - The table V will be

item j \ Capacity i		**1**	**2**	**3**	**4**	**5**	**6**
	0	0	0	0	0	0	0
1	0	0	4	4	4	4	4
2	0	0	4	4	4	6	6
3	0	2	4	6	6	6	8
4	0	2	4	6	6	6	8

    - Therefore, the maximum value of a knapsack with a capacity of 6 should be 8.

Game of Stone

1. (A): $dp(i,j) = P_i + min\{dp(i+2,j), dp(i+1,j-1)\}$
   (B): $dp(i,j) = min\{dp(i+1,j-1), dp(i,j-2)\} + P_j$

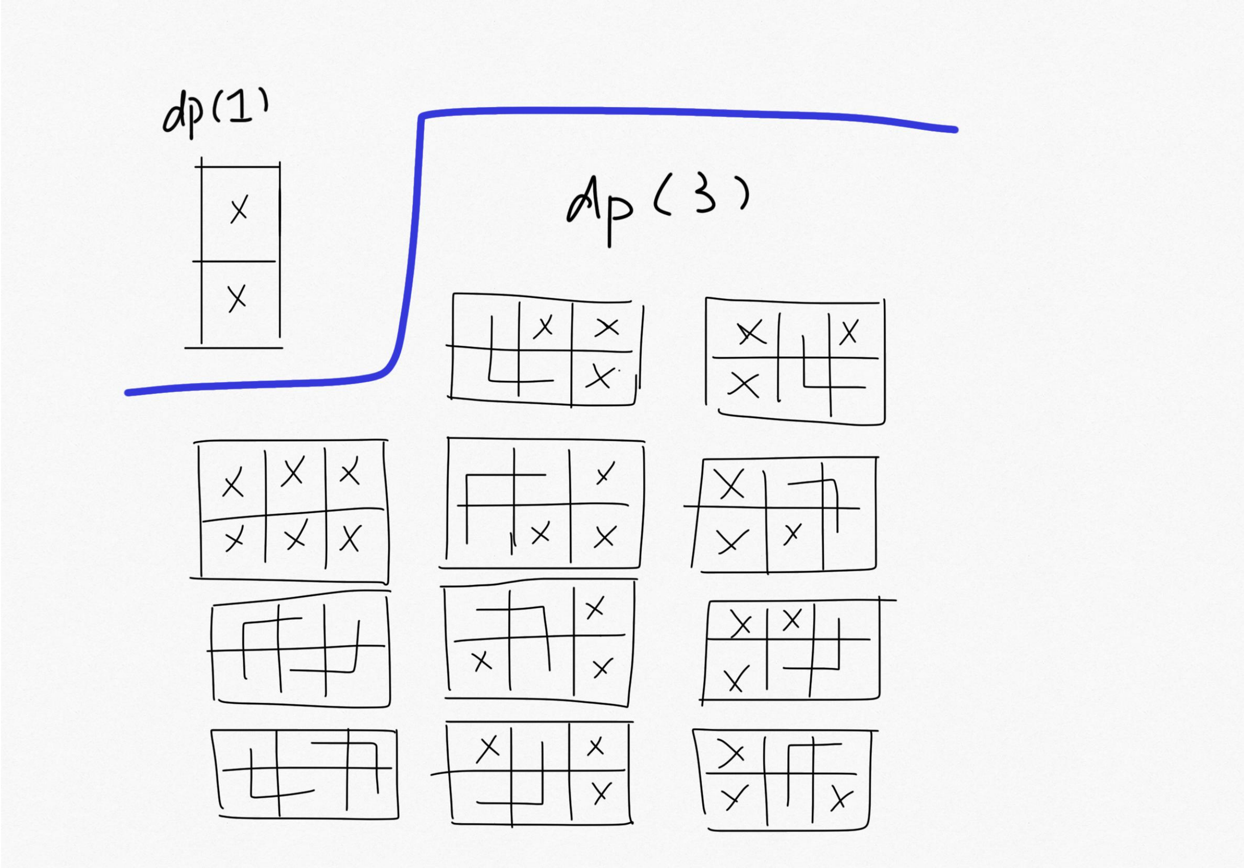
   我的策略是這一輪（n輪）拿最多的，對方下一輪(n+1輪)的最佳策略則是使我在下下輪(n+2輪)拿到最少的。由於堆疊有偶數個，必能保證我可以透過適當拿取策略，使得對手不得不露出我最想要的堆疊（也許是超級多的石頭）在最前端或最末端，若此為最佳解，先選者可以透過上述遞迴式，推演出最佳策略，即每次皆選取上述(A)、(B)中較大者。

2. 橫座標向右為i，縱座標向下為j。
   $dp(i,j) = Max\{P_i + min\{dp(i+2,j), dp(i+1,j-1)\}, min\{dp(i+1,j-1), dp(i,j-2)\} + P_j\}$

P_i		**2**	**8**	**3**	**7**	**5**	**3**
	$j \backslash i$	1	2	3	4	5	6
2	1	2	8	5	15	10	18
8	2		8	8	11	16	16
3	3			3	7	8	10
7	4				7	7	10
5	5					5	5
3	6						3

2. Bento

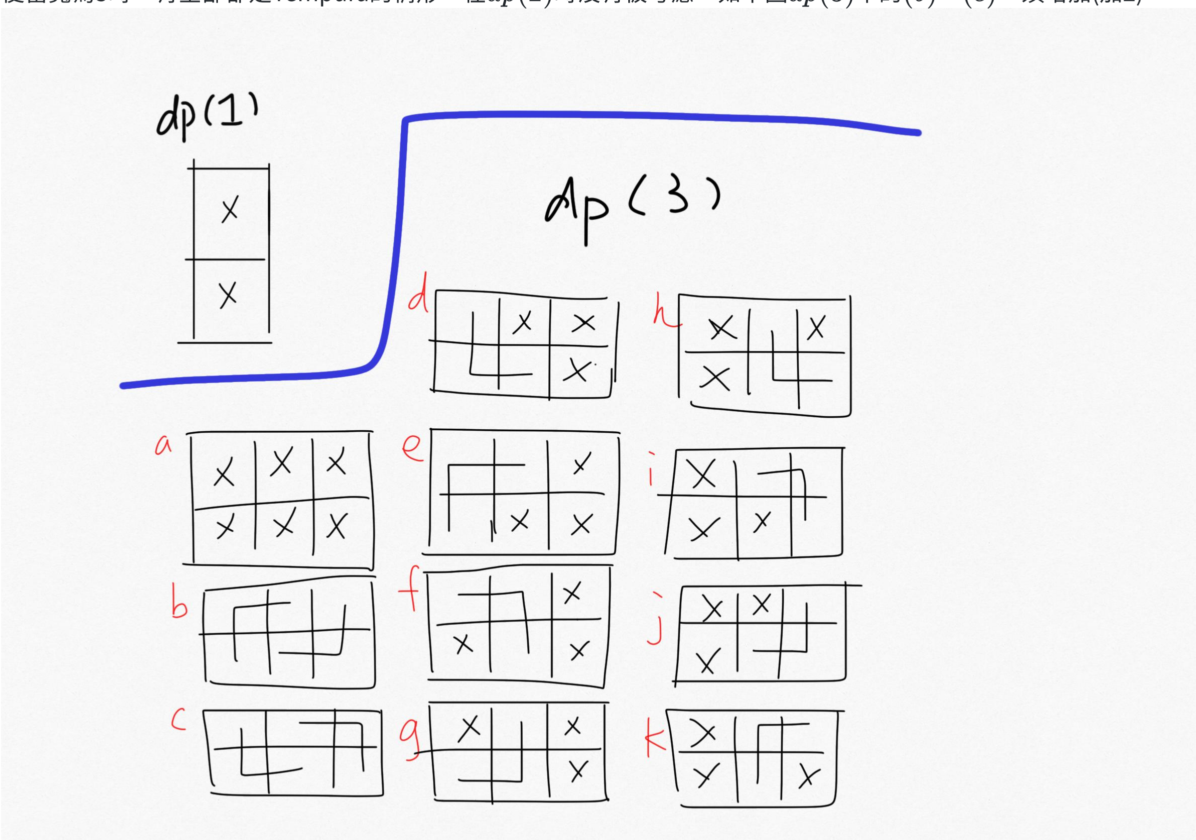
1. $dp(1) = 1$，$dp(3) = 11$。



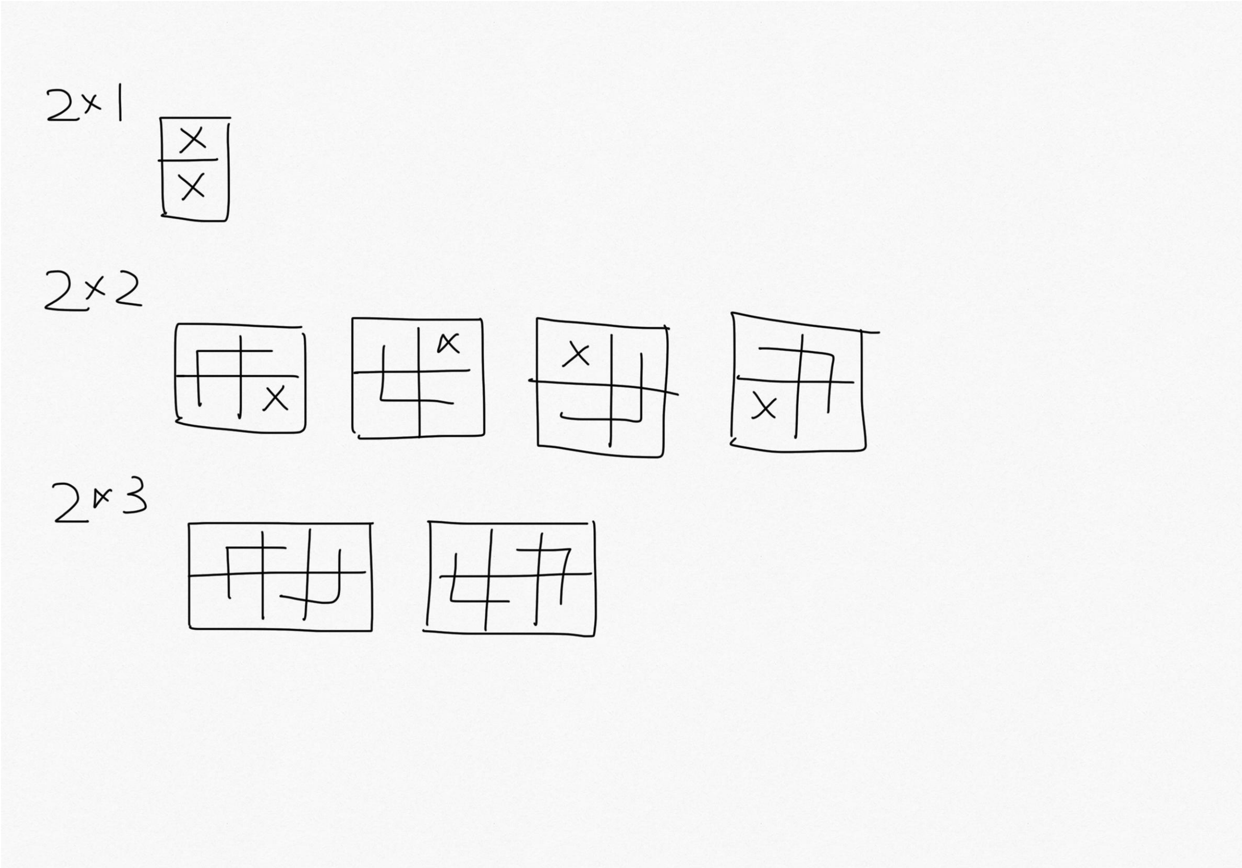
2. 不行。

   i. 有一情形在$dp(2) + dp(1)$和$dp(1) + dp(2)$重複出現（計算）了，即便當裡全都是Tekkamaki的狀況，如下圖$dp(3)$中的(a)，須扣除(減1)。

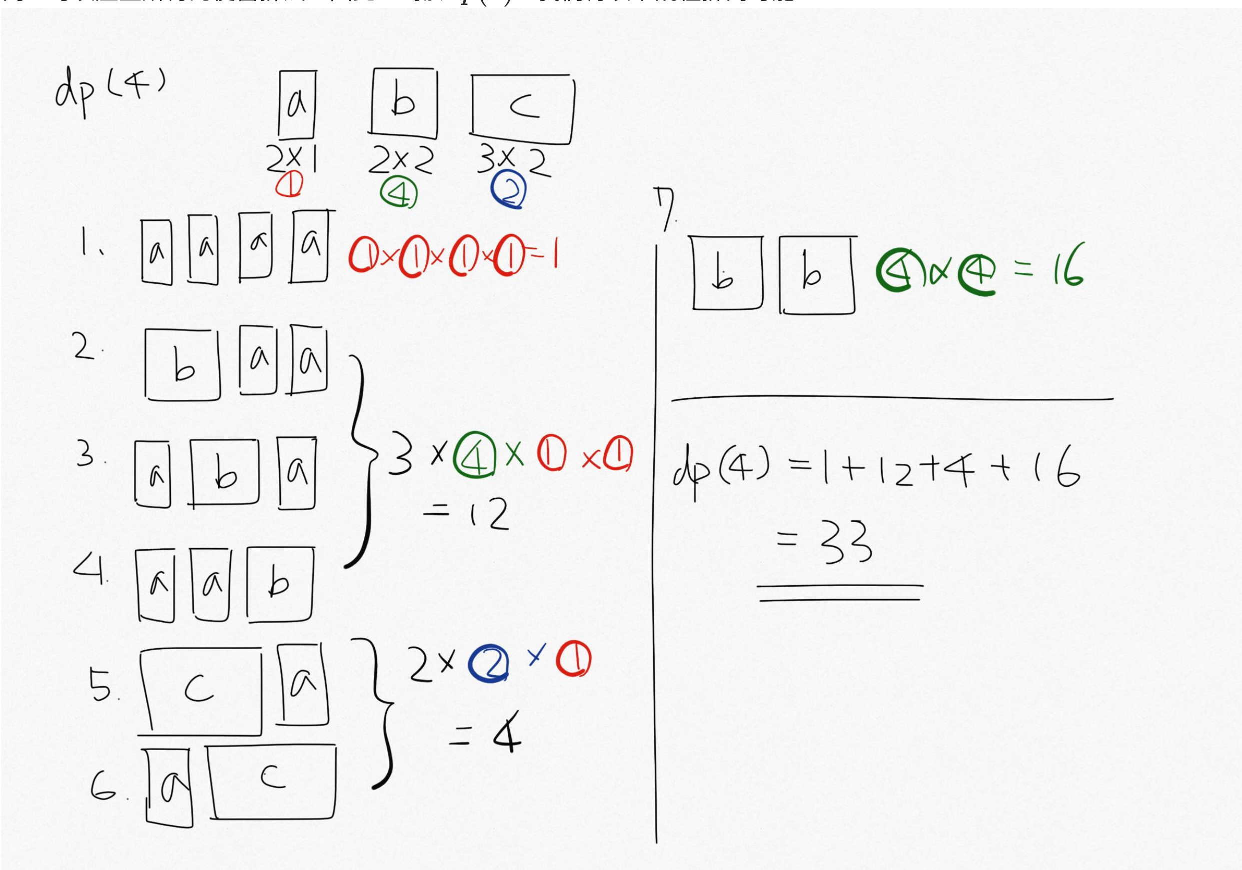
   ii. 便當寬為3時，有全部都是Tempura的情形，在$dp(2)$時沒有被考慮，如下圖$dp(3)$中的(b)、(c)，須增加(加2)。



3. 我們可以將便當樣貌拆解成2x1（a,1種）、2x2（b,4種）、2x3（c,2種）的類型基本獨立組合，並在計算$dp(4)$時將基本獨立組合排序，創造不同樣貌。



這些「基本獨立組合」是最小單元，也就是說，它不能由其它「基本獨立組合」排列得到。然而，它們經由相互搭配排列，可以產生所有的便當排法。因此，對於$dp(4)$，我們有以下幾種排列可能：



根據a, b, c分別有$1, 4, 2$種樣貌，及上圖分析a, b, c的排列方式，可以得到$dp(4) = 33$。

4. 對於$dp(i)$，首先我們需要知道2xi的便當可以如何由a(2x1)、b(2x2)、c(2x3)疊加、排列而成，亦即如何囊括所有重複1、2、3三個數字且總和為i的排列。設一排列中，總共有x個1，y個2，z個3。根據a, b, c分別有$1, 4, 2$種樣貌，可以得到某一排列有 $1^x * 4^y * 2^z$ 個便當樣貌。

我們先 (1)得到所有總和為i的組合，再 (2)討論各組合的有幾種排列，再將 (3)各排列分別依照a, b, c分別有$1, 4, 2$種樣貌的原則算出便當的樣貌數量，最後 (4)將所有便當樣貌數量加總。

i. 舉出所有組合。
   a. 我們可以首先將i用最少的數字組合，亦即盡量用3，不能的話用2，不得已才用1來組成i。例如對於$i = 17$，我們以$17 = 3 + 3 + 3 + 3 + 3 + 2$開始。
   b. 紀錄此組$1, 2, 3$的分別使用個數(x, y, z)，以此例為$(0, 1, 5)$。
   c. 對該組最右邊第一個不為1的組成數拆解，以該組成數等於（優先採取）或大於右邊的組成數為原則。若為2就拆成1+1；若遇到3，將(該組成數與其右邊的組成數)之總和以最多的2組成，不得已才放一個1在末端。由此得到一組新的組合。例：$(3 + 3 + 3 + 3 + 3 + 2)$→$(3 + 3 + 3 + 3 + 3 + 1 + 1)$→$(3 + 3 + 3 + 3 + 2 + 2 + 1)$
   d. 重複步驟2.~3.，直到所有組成數皆為1。

ii. 計算排列數目。
   a. 針對某一組合使用的3,2,1個數 (x, y, z) ，該組合可以有 $\frac{(x+y+z)!}{x!y!z!}$ 種排列方式。例如針對$(x, y, z) = (0, 1, 5)$，共有$\frac{(6)!}{0!1!5!} = 6$ 種排列方式。

iii. 計算一排列的便當樣貌數。
   a. 一種排列方式有 $1^x * 4^y * 2^z$ 種便當樣貌數。例如$(3, 3, 3, 3, 3, 2)$的排列，$(x, y, z) = (0, 1, 5)$，可以創造 $1^0 * 4^1 * 2^5 = 128$ 種便當樣貌數。

iv. 計算總共便當樣貌數。
   a. 相同組合不同排列因為x, y, z相同，便當樣貌數也相同。故同一組合，其總便當樣貌數等於排列數目乘以一排列便當樣貌數，即$\frac{(x+y+z)!}{x!y!z!} * 1^x * 4^y * 2^z$。
   b. 加總所有(x, y, z)組合的便當樣貌數，即為總共的便當樣貌數。

3. Watermelon

1. 每一格可以是0或1兩種可能，共有M格，故有 2^M 種狀態。
2. 將G用二進位表示會有M位，最小值為每一位都是0的狀態，此時的十進位數字為0；最大值為每一位都是1的狀態，此時的十進位數字為$2^M - 1$。故其值域為$[0..2^M - 1]$。
3. 將P與S做&運算（bitwise AND operation），即 `P & S` ，其值若為0即代表兩列間沒有相鄰的西瓜；若值不為0，則代表有相鄰的西瓜。將值展開即可看到是第幾行兩列間都種西瓜。
4. 做 (S AND (NOT $F[i]$))運算，即 `S & (~F[i])` ，其值若為0即代表西瓜都種在對的地方；若值不為0，則代表有西瓜誤種在不能種的地方。將值展開即可看到是第幾行有誤。