CS170–Fall 2022 — Homework 7 Solutions

Deming Chen cdm@pku.edu.cn

Last Modified: January 8, 2023

Collaborators: NONE

2 Egg Drop Revisited

(a) Suppose we drop the first egg from floor j. If the egg breaks, we know $l \leqslant j-1$. So, $j-1 \leqslant M(x-1,k-1)$. If the egg does not break, we know that $l \geqslant j$. Then, we need to find l in at most $x-1$ drops and k eggs, which implies that $M(x,k)-j \leqslant M(x-1,k)$. To maximize $M(x,k)$ these inequalities are tight, so

$$M(x,k) = M(x-1,k-1) + M(x-1,k) + 1. \tag{1}$$

(b) The base cases are:

$$M(1,k) = M(x,1) = 1, \forall x,k \in \mathbb{N}^*. \tag{2}$$

So, the algorithm is

```
function M(x, k)
for j = 1, 2, ··· , k:
    M(1, k) = 1
for i = 2, 3, ··· , x:
    M(i, 1) = 1
    for j = 2, 3, ··· , k:
        M(i, j) = M(i − 1, j − 1) + M(i − 1, j) + 1 .
```

Obviously, the algorithm takes time $\mathcal{O}(kx)$.

(c) By definition, $M(x,k)$ is non-decreasing of k:

$$M(x,k_1) \leqslant M(x,k_2),\ \forall k_1 < k_2. \tag{3}$$

Besides, by equation (1) $M(x,k)$ is strictly increasing of x, i.e. when $k \geqslant 2$:

$$M(x_1,k) + 1 \leqslant M(x_2,k),\ \forall x_1 < x_2. \tag{4}$$

So, we just need to search x from small positive integers in order to compute $f(n,k)$ (shown as Algorithm 1).

(d) Last week we use the following recurrence relation:

$$f(n,k) = 1 + \min_{1\leqslant j\leqslant n} \max\{f(j-1,k-1), f(n-j,k)\}. \tag{5}$$

The subproblems constitute a two-dimensional $n \times k$ table, each of whose entries takes $\mathcal{O}(n)$ time to compute. The overall running time is $\mathcal{O}(n^2k)$.

Algorithm 1. Computing $f(n,k)$ by $\mathbf{M}(x,k)$

```
function f(n, k)
for j = 1, 2, ··· , k:
    M(1, k) = 1
for i = 2, 3, 4, ··· :
    M(i, 1) = 1
    for j = 2, 3, ··· , k:
        M(i, j) = M(i − 1, j − 1) + M(i − 1, j) + 1 .
    if M(i, k) ⩾ n:
        return i
```

(e) Since $M(n,k) \geqslant n$ for $k \geqslant 2$, the outer loop of Algorithm 1 terminates for some $i \leqslant n$. Therefore, its runtime is $\mathcal{O}(nk)$.

We need to modify Algorithm 1. As base cases we know $M(i,1) = 1$. Then we compute $M(i,2), i = 1, 2, \cdots, k$. Afterward, we compute $M(i,3), i = 1, 2, \cdots, k$. Now, $M(\cdot, 2)$ are not needed anymore. Formally, after compute $M(i,j), i = 1, 2, \cdots, k$, we can release the space of $M(i, j-1), i = 1, 2, \cdots, k$. So, the algorithm using $\mathcal{O}(k)$ space.

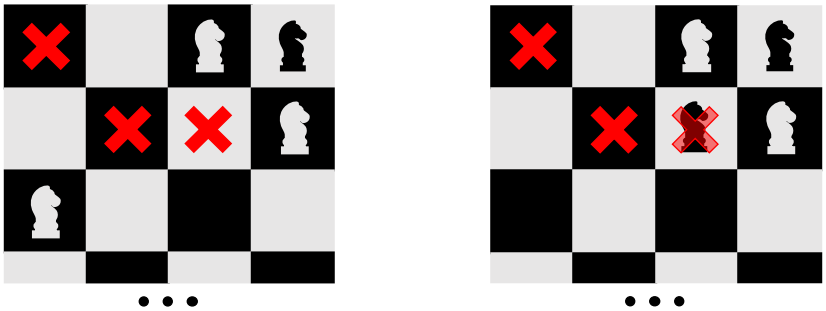
This Algorithm 1, which takes time $\mathcal{O}(nk)$, is much faster than last week's algorithm.

3 Knightmare

(a) Define $f : \mathbb{N}^* \times \{0,1\}^{2\times L} \to \mathbb{N}^*$. $f(m, A)$ is the number of ways you can place knights on an m by L chessboard such that if $A_{ij} = 1$, then you cannot place a knight on the row i column j. For example, as shown in Figure 1, let $L = 4$,

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}. \tag{6}$$

Then we cannot place a knight on the square at the second row and the third column.



(a) (b)

Figure 1: Given matrix A, (a) is a valid placement of knights, but (b) is invalid.

The answer to our problem is to take as input $m = M$ and an all-zero 2 by L matrix.

(b) The base cases is when $m = 2$, with all possible As. In fact, there are 2^{2L} possible As. However, given A it is easy to compute $f(2, A)$.

Now, let's consider the recurrence relation for f. Given A, we can place knights on squares at the first row whose corresponding entries of A is 0. Any placement P on the first row determines which squares we cannot place knights on at the third row, denoted by a zero-one row vector $b \in \{0,1\}^{1\times L}$. So, there is a map: $\pi : P \mapsto b$.

The recurrence relation for f is

$$f(m+1, A) = \max_P f(m, A') + \Sigma P, \tag{7}$$

where the first row of A' is the same as the second row of A, and the second row of A' is just $\pi(P)$, and ΣP is the number of knights we place on the first row of P.

(c) To solve the problem, we need to calculate $f(2, A)$ of every possible A (base cases), then $f(3, A)$ of every possible A, and so on until $m = M$. In recurrence, we enumerate all possible placements on the first row, and compare them to get the maximum number of knights. The correctness can be easily proven by induction.

(d) The subproblems constitute a $(M-1) \times 2^{2L}$ table, each of whose entries takes $\mathcal{O}(2^L)$ time to compute. The overall running time is $\mathcal{O}(M \cdot 2^{3L})$.

We do not need to restore all entries of the table permanently. Since only $f(m, \cdot)$ is needed when computing $f(m+1, \cdot)$, we only need to restore 2 rows. So, the space complexity is $\mathcal{O}(2^{3L})$.

5 Baker

(a) Let's say it makes x_1 batches of brownies and x_2 batches of cookies to produce. The objective function is:

$$\max \ 60x_1 + 30x_2. \tag{8}$$

Linear constrains are

$$\begin{aligned} 4x_1 + x_2 &\leqslant 80 \\ 2x_1 + 3x_2 &\leqslant 90 \\ x_1, x_2 &\geqslant 0. \end{aligned} \tag{9}$$

The feasible region is shown in Figure The optimal solution is 1500 at $x_1 = 15, x_2 = 20$.

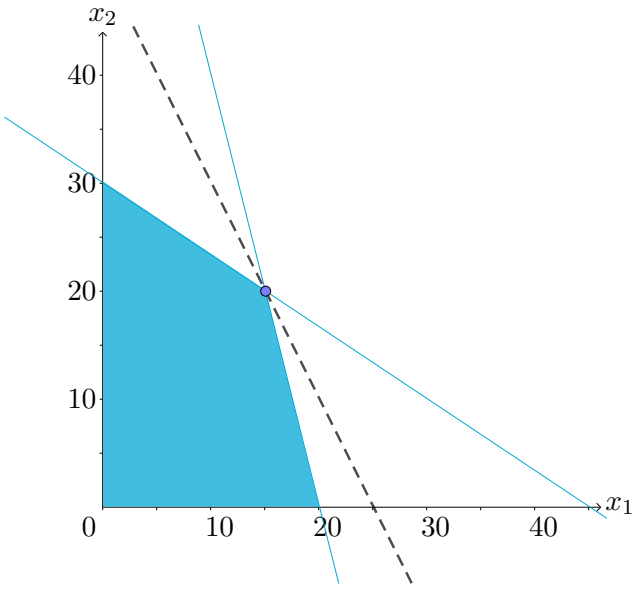


Figure 2: The feasible region for the problem.

(b) If the optimal solution is at $(15, 20)$, then the slop of objective function $r = Cx_1 + 30x_2$ is between -4 and $-2/3$. So, $20 \leqslant C \leqslant 120$.

6 Meal Replacement

(a) Suppose Jonny buys x_1 pound of salmon, x_2 pound of bread and x_3 pound of squid. His objective is to minimize his cost:

$$\min \ 5x_1 + 2x_2 + 4x_3. \tag{10}$$

The constrains are

$$\begin{aligned} 500x_1 + 50x_2 + 300x_3 &\geqslant 500 \\ 300x_2 + 100x_3 &\geqslant 800 \\ 500x_1 + 25x_2 + 200x_3 &\geqslant 700 \\ x_1, x_2, x_3 &\geqslant 0. \end{aligned} \tag{11}$$

(b) The dual problem is

$$\begin{aligned} \texttt{Objective function} \quad & \max \ 500y_1 + 800y_2 + 700y_3 \\ \texttt{Constrains} \quad & 500y_1 + 500y_2 \leqslant 5 \\ & 50y_1 + 300y_2 + 25y_3 \leqslant 2 \\ & 300y_1 + 100y_2 + 200y_3 \leqslant 4 \\ & y_1, y_2, y_3 \geqslant 0. \end{aligned} \tag{12}$$

(c) Suppose each pill provide exactly on calorie of protein, carbs, and fiber. The pharmacist needs to determine the price of pills. Let's y_1, y_2 and y_3 denote the price of one protein pill, one carb pill, and one fiber pill, respectively. In order to convince Jonny to buy pills instead of food, the price of pills should be no higher than the price of food which provide the same composition. For example, the total price of 500 calorie pills and 500 fat pills should be no higher than that of one pound of salmon, i.e. $500y_1 + 500y_2 \leqslant 5$. Since Jonny will buy 500 protein pills, one 800 carb pills, and one 700 fiber pills for one penguin, maximizing the pharmacist's profit is equivalent to maximizing $500y_1 + 800y_2 + 700y_3$.