

## 1 Reading And/Or Weeping


**Subproblem:** Let  $\text{OPT}(i, j)$  be the total unhappiness you will experience from reading sections  $j, \dots, n$  optimally over the course of days  $i, \dots, D$ .

**Recurrence:**

Notation:

- Let  $T(j, j') = \sum_{k=j}^{j'} t_k$ . That is,  $T(j, j')$  is the time it takes to read sections  $j$  through  $j'$ . Let  $T(j, j-1) = 0$  - it takes no time to read nothing.
- Let  $F_i(j, j') = \max(m_i - T(j, j'), 0)$ . That is,  $F_i(j, j')$  is the total free time you would have on day  $i$  if you chose to read sections  $j$  through  $j'$  that day.
- Let  $S_i(j, j') = \max(T(j, j') - m_i, 0)$ . That is,  $S_i(j, j')$  is the total sleep you would lose on day  $i$  if you chose to read sections  $j$  through  $j'$  that day.
- Let  $U_i(j, j') = F_i(j, j')^4 + S_i(j, j')$ . That is,  $U_i(j, j')$  is the total unhappiness you would experience on day  $i$  if you chose to read sections  $j$  through  $j'$  that day.

$$\text{OPT}(i, j) = \min_{j' \geq j-1} [U_i(j, j') + \text{OPT}(i+1, j'+1)].$$

**Proof of Recurrence:** In choosing a schedule for sections  $j$  and onwards and days  $i$  and onwards, you may choose any prefix of  $j, \dots, n$  read on day  $i$ . If you read jobs  $j$  through  $j'$  on day  $i$ , your total unhappiness will be the unhappiness from day  $i$ ,  $U_i(j, j')$ , plus the additional unhappiness from the rest of your reading, which is  $\text{OPT}(i+1, j'+1)$ , as you must now read sections  $j'+1$  through  $n$  on days  $i+1$  through  $D$ . Since you are scheduling to minimize the total unhappiness, you choose the best of all prefixes to read on job  $i$ , meaning you  take the minimum.

**Base Cases:**  $\text{OPT}(D+1, j) = \infty$  and  $\text{OPT}(i, n+1) = \sum_{k=i}^D m_i^4$ . The first base case comes from the fact that you must read all the sections within the month. The second comes from the fact that if you have nothing left to read, you are bored for the rest of the month.

**Algorithm:**

```
Memo[ ][ ] = new int[D+1][n+1]
```

```
for j from 1 to n
```

```
    Memo[D+1][j] = ∞
```

```
compute  $\sum_{k=i}^D m_i^4$  for each  $i$ 
```

```
for i from 1 to D
```

```
    Memo[i][51] =  $\sum_{k=i}^D m_i^4$ .
```

```
for i from D+1 to 1
```

```
    for j from n+1 to 1
```

$$\text{Memo}[i][j] = \min_{j' \geq j-1} \left( U_i(j, j') + \text{Memo}[i+1][j'+1] \right)$$

- return Memo[1][1]

**Runtime:** We write the runtime in terms of the number of days  $D$ , and the number of sections  $n$ . Computing the first base case requires  $O(n)$  time. Computing  $\sum_{k=i}^D m_i^4$  for each  $i$  can be implemented in  $O(D)$  time. It follows that computing the second base case is also an  $O(D)$  operation. Finally, filling in the table involves filling  $O(Dn)$  entries, and each entry requires  $O(n)$  table lookups to fill. It follows that the algorithm runs in  $O(Dn^2)$  time.

## 2 FiEncournstters

[Note: for this problem, we are indexing 1, but will want to take modulus with respect to  $p$  and  $q$ . The mod  $x$  operation as usually defined maps to  $\{0, \dots, x-1\}$ , but it's easier to talk about  $\{1, \dots, x\}$ . Therefore, I want to define the modulus the normal way, except when it would evaluate to 0, in which case we modify it to evaluate to  $x$ . For example,  $5 \bmod 3$  is still 2, but  $6 \bmod 3$  becomes 3.]

**Subproblem:**  $\text{OPT}(i, j)$  = whether or not it is possible to write  $c_{i+j-1} \dots c_n$  as an interleaving of  $a$  and  $b$ , with the first copy of  $a$  starting at  $a_{i \bmod p}$  and the first copy of  $b$  starting at  $b_{j \bmod q}$ .

**Recurrence:**  $\text{OPT}(i, j) = [(c_{i+j-1} == a_{i \bmod p}) \wedge \text{OPT}(i+1, j)] \vee [(c_{i+j-1} == b_{j \bmod q}) \wedge \text{OPT}(i, j+1)]$

**Proof of recurrence:** When we're trying to determine whether or not it is possible to write  $c_{i+j-1} \dots c_n$  as an interleaving of  $a$  and  $b$ , with the first copy of  $a$  starting at  $a_i$  and the first copy of  $b$  starting at  $b_{j+1 \bmod q}$ , we need to decide whether  $c_{i+j-1}$  belongs to  $a$  or  $b$ . For it to belong to  $a$ , it must be that  $c_{i+j-1} = a_{i+1 \bmod p}$ , and that we can then write  $c_{i+j} \dots c_n$  as an interleaving of  $a$  and  $b$ , with the first copy of  $a$  starting at  $a_{i+1 \bmod p}$  and the first copy of  $b$  starting at  $b_j \bmod q$ . Similarly, if  $c_{i+j-1}$  belongs to  $b$ , it must be that  $c_{i+j-1} = b_{j \bmod q}$ , and that we can then write  $c_{i+j} \dots c_n$  as an interleaving of  $a$  and  $b$ , with the first copy of  $a$  starting at  $a_i \bmod p$  and the first copy of  $b$  starting at  $b_{j+1 \bmod q}$ . As long as one of these two sets of conditions holds, then we can write  $c_{i+j-1} \dots c_n$  as an interleaving of  $a$  and  $b$ , with the first copy of  $a$  starting at  $a_i$  and the first copy of  $b$  starting at  $b_j$ .

**Base Cases:** For all  $i$  and  $j$  such that  $i + j - 1 = n + 1$ ,  $\text{OPT}(i, j) = \text{T}$  if and only if  $i = 1 \bmod p$  and  $j = 1 \bmod q$ .

**Iterative Algorithm:**

Memo[][] = new boolean[n+1][n+1]

for all  $i$  and  $j$  such that  $i + j - 1 = n + 1$ , Memo[i][j] = T if and only if  $i = 1 \bmod p$  and  $j = 1 \bmod q$ .

for  $k$  from 1 to  $n$

for  $i$  from 1 to  $n+1-k$

$j = n + 2 - k - i$

$$\text{Memo}[i][j] = ((c_{i+j-1} == a_{i \bmod p}) \wedge \text{Memo}[i+1][j]) \vee ((c_{i+j-1} == b_{j \bmod q}) \wedge \text{Memo}[i][j+1])$$

return Memo[1][1]

**Runtime:** Our memo table is of size  $O(n^2)$ . To fill in an entry requires constant work. Our runtime is therefore  $O(n^2)$ .

### 3 Cat People

**Subproblem:** Let  $\text{OPT}(i, k, c)$  be whether or not it is possible for the domestics to win both divisions with universities  $i, \dots, n$  yet to be assigned to a division,  $k$  universities already assigned to division A, and a lead of  $c$  for the domestics in division A from universities already assigned.

**Recurrence:**

$$\text{OPT}(i, k, c) = \begin{cases} \text{OPT}(i+1, k, c) & \text{if } k = n/2 \\ \text{OPT}(i+1, k+1, c+d_i-f_i) \parallel \text{OPT}(i+1, k, c) & \text{otherwise} \end{cases}$$

**Proof of recurrence:** Imagine that you've already assigned universities  $1, \dots, i-1$ , and you're wondering if you can pull out a Domestic victory with some assignment of the remaining universities. If you have already assigned  $n/2$  universities to division A, then you cannot assign any more, and must assign university  $i$  to division B. The number of universities in division A will still be  $k$ , and the lead will still be  $c$ . You will have to assign universities  $i+1, \dots, n$  yet. This is the first case in the recurrence.

If there are not  $n/2$  universities in division A, then you are left with the choice of assigning  $i$  to division A or division B. If you assign it to division A, you'll be left with  $i+1, \dots, n$ , with  $k+1$  in division A and a lead of  $c+d_i-f_i$ . Otherwise, you'll have  $i+1, \dots, n$ ,  $k$  in division A, and a lead of  $c$ . As long as a win is possible with one of these two options, it will be possible in the subproblem with  $i, \dots, n$ ,  $k$  in division A, and a lead of  $c$ .

**Base Cases:** For all  $k$  and  $c$ ,  $\text{OPT}(n+1, k, c) = \text{T}$  if and only if  $k = n/2$  and both  $c > 0$  and  $\sum_i (d_i - f_i) - c > 0$ .

**Iterative Algorithm:**

Let  $D = \sum_i d_i$  and  $F = \sum_i f_i$ .

Memo[][][] = new boolean[n+1][n/2 + 1][M]

for all  $k$  and  $c$ , Memo[n+1][k][c] =  $(k == n/2) \wedge (c > 0) \wedge (D - F - c > 0)$

for  $k$  from 0 to  $n/2$

for  $c$  from  $-F$  to  $D$

if  $k == n/2$  Memo[i][k][c] = Memo[i+1][k][c]

else Memo[i][k][c] = Memo[i+1][k+1][c+d<sub>i</sub>-f<sub>i</sub>] || Memo[i+1][k][c]

```
return Memo[1][0][0]
```

**Runtime:** Our table is of size  $O(n^2M)$ , and it takes constant work to fill in an entry, so the runtime is  $O(n^2M)$ .