Aditya Shirode, Rhythm Shah, Shrenuj Gandhi

1. [8 points (for the general solution): 3 points for the recursive formula, 2 points for the table filling algorithm, 1 point for the time bound and 2 points for the algorithm that retrieves the choices]

   Suppose you have inherited the rights to 500 previously unreleased songs recorded by the popular group Raucous Rockers. You plan to release a set of 5 CD's (numbered 1 through 5) with a selection of these songs. Each disk holds a maximum of 60 minutes of music and each song must appear in its entirety on one disk. Since you are a classical music fan and have no way of judging the artistic merits of the songs, you decide to use the following criteria: the songs will be recorded in order by the date they were written, and the number of songs included will be maximized. Suppose you have a list $\ell_1, \ell_2, \ldots, \ell_{500}$ of the lengths of the songs, in order by the date they were written (no song is more than 60 minutes long). Give an algorithm to determine the maximum number of songs that can be included using the given criteria. Hint: Let $T(i, j)$ be the minimum amount of time needed for recording any i songs from among the first j songs. You should interpret T to include blank time at the end of a completed disk. For example, if a selection of songs uses all of the first disk plus 15 minutes of the second disk, the time for that selection should be 75 minutes even if there is blank space at the end of the first disk. Generalize your solution to a situation where there are m CD's, n songs, and a capacity c for each of the CD's. What is the asymptotic runtime in terms of m and n (treating c as a constant)?

Answer:

We have a list $l_1, l_2, \ldots, l_{500}$ of the lengths of the songs ordered by the date they were written.
Selection of songs from left to right will satisfy part of the criterion.
Now in order to *maximize* the number of songs on the CDs, we must *minimize* the total amount of time needed for recording those selected songs.

With the goal of minimizing the recording time, we can formulate the following recursion -

$$T[1][1] = L_1 \qquad\qquad \text{Base case}$$

$$T[1][j] = \min_{1 \le j \le n} (T[1][j-1], L_j)$$

$$T[i][j] = \min_{1 \le i \le n, i \le j \le n} \left(T[i][j-1], \begin{cases} T[i-1][j-1] + L_j & \text{if } (T[i-1][j-1] \bmod c) + L_j \le c \\ K * c + L_j & \text{if } (T[i-1][j-1] \bmod c) + L_j \ge c \text{ and} \\ & K = \left\lceil \frac{T[i-1][j-1]}{c} \right\rceil \end{cases} \right)$$

where $T[i][j]$ is the minimum amount of time needed for recording any i songs from amongst first j songs, and c is the capacity of each CD.

From the above recurrence, we have the following table filling algorithm -

```
▷Lower triangular matrix assigned to infinity (selecting more number of songs than available songs)
for i ← 1 to n, do
        for j ← 1 to n, do                                      O($n^2$)
                if i > j, then
                        T[i][j] ← ∞

▷Selecting one song out of one song
T[1][1] ← L[1]                                                   O(1)

▷Selecting (or not) the $j^{th}$ song
for j ← 2 to n, do
        T[1][j] ← min(T[1][j−1], L[j])                           Θ($n$)

for i ← 2 to n, do
        for j ← 2 to n, do                                      O($n^2$)
                if i=j, then
                ▷ record all required songs for first $j$ songs
                        if T[i−1][j−1] + L[j] ≤ c, then
                                T[i][j] ← T[i−1][j−1] + L[j]
                        else
                                K = $\left\lceil \frac{T[i-1][j-1]}{c} \right\rceil$
                                T[i][j] ← K ∗ c + L[j]
                ▷ S[i][j] ← True

                else if i ≤ j, then
                ▷record any i songs from first j songs
                        if T[i−1][j−1] mod c + L[j] ≤ c

                                T[i][j] ← min(T[i][j−1], T[i−1][j−1] + L[j])
                                ▷ S[i][j] ← False if min is T[i][j−1]
                                ▷ S[i][j] ← True otherwise

                        else
                                K = $\left\lceil \frac{T[i-1][j-1]}{c} \right\rceil$
                                T[i][j] ← min( T[i][j−1], K ∗ c + L[j])
                                ▷ S[i][j] ← False if min is T[i][j−1]
                                ▷ S[i][j] ← True otherwise
```

where n is No. of Songs, c is capacity for each CD.
To determine the maximum no of songs that can be included using the given criteria, we observe the last column in bottom-up manner. The row number $(i) \leq$ the product of m and c indicates the maximum no of songs.

```
for i ← n downto 1, do
        if T[i][n] ≤ cm                          O($n$)
                count ← i
                ▷atmost i songs can fit into m CDs of c capacity
        return
```

```
i ← count

for j ← n downto 1, do
          if S[i][j] = True                                    O($n$)
                   A ← A ∪ i
                   i ← i−1
```

▷This retrieves the selected choices

```
print    reverse of A
```

As we can see, the most complex operation is the table filling operation, which makes runtime of this algorithm O(n^2).