

Summation Warmup. Give a $\theta(\cdot)$ bound or a closed form for the following sums.

1. $\sum_{i=1}^n 2^i$

2. $\sum_{i=1}^n \frac{1}{2^i}$

3. $\sum_{i=1}^n \left(\frac{9999}{10000}\right)^i$

4. $\sum_{i=1}^n i$

5. $\sum_{i=1}^n \frac{1}{i}$

Merge Recurrences. Solve the following recurrences to a closed $\theta(\cdot)$ form.

$$1. T(n) = \begin{cases} T(n/2) + T(n/3) + T(n/6) + n & \text{if } n \geq 2 \\ 381 & \text{if } n < 2 \end{cases}$$

$$2. T(n) = \begin{cases} T(n/2) + 3T(n/4) + n^3 & \text{if } n \geq 2 \\ 451 & \text{if } n < 2 \end{cases}$$

$$3. T(n) = \begin{cases} T(676n/6767) + T(6767n/67676) + n & \text{if } n \geq 1 \\ 67 & \text{if } n < 1 \end{cases}$$

Hint: Using the previous part, can you deduce this recurrence's closed form without doing the whole tree method?

Schedule-slopping Friendslop. Recently, *Big Walk*, a new hit Friendslop game, released to critically acclaimed reviews¹. Justin would like to play *Big Walk* with his friends, but they are all old adults with jobs and prior commitments or something. Your objective is to find the number of times which Justin and his friends can all play (assuming there is such a time). They only want to play *Big Walk* if they can all meet at least 6 times (or 7 if they're bad²).

Formally, there are m friends. Each friend i has a list of times they are busy represented by disjoint, sorted ranges $L_i = \{[a_1^i, b_1^i], \dots, [a_t^i, b_t^i]\}$. Each range $[a_j^i, b_j^i]$ represents the *inclusive* start a_j^i and end b_j^i of their busy times. Assume time (measured in some arbitrary unit) starts at 0 and ends at the largest number given in the input. In total, there are n busy times. Give an algorithm that takes $(L_1, L_2, \dots, L_m)$ as input and outputs the number of times they can meet. In other words, output the number of occurrences where there is a gap between busy times.³

Example ($m = 2, n = 5$):

Input: $L_1 = [(1, 2), (4, 6), (8, 12)]$

$L_2 = [(1, 3), (5, 9)]$

Output: 2 because the number of free times are $[(0, 1), (3, 4)]$

¹As of writing this problem, it has a 10/10 on Steam, 4.6/5 on the PlayStation store, and a 91% on Metacritic.

²__ __-

³Hint: Can you give a $O(n^2)$ algorithm? Can you give a faster algorithm?