SCT Blitz 1 Editorial

June 1st, 2020

Bus Stop

If Justin waits strictly more than M minutes at a stop, he will have to board a new bus. Count the number of stops such that $c_i > M$ and add 1 to account for the first bus he boards.

Constructive Array

This question has many possible approaches. One possible solution is $\{M, M, S-2M\}$. Note that no matter what values of M and S are used, the median will always be M.

AND Pairs

Consider every bit in N independently. If the bit is `1`, then the corresponding bit in both a and b must be `1`. However, if the bit is `0`, there are three options for the corresponding bits.

Only pairs $(a,\ b)$ where $a \le b$ are considered in the answer, which means we overcount by a factor of two—except for one case where $a = b = N$. If k is the number of unset bits in N, then our answer is $(3^k + 1)/2$.

0-1 Knapsack

This is a very straightforward $N \times H$ DP transition. Let $dp[i][h]$ be the number of ways to get height h using a subset of the first i blocks. Clearly, $dp[i][h] = dp[i-1][h] + dp[i-1][h-h_i]$.

Looking at the memory constraints, however, we'll see that making an $N \times H$ matrix won't pass. The crucial observation here is that when calculating the DP for i, we only care about the values we got for $i-1$. If we manage memory efficiently, we're left with a solution in $O(NH)$ time and $O(H)$ memory, which will pass.

Bowling

First, consider a modification to this problem. Instead of finding the sum $\sum_{i=l}^{r}(i-l+1)w_i$, we will calculate the sum $\sum_{i=l}^{r} i w_i$. If we multiply every value of the array by its index before reading queries, as well as handle update queries in a similar fashion, this becomes a very straightforward use of a segment tree (or Fenwick tree).

Now, back to the original problem. With a bit of simple algebra, we can rewrite the target sum as $\sum_{i=l}^{r} iw_i - (l-1)\sum_{i=l}^{r} w_i$. Now answering queries is very straightforward. We can maintain a second segment tree with the original values, and simply calculate the answer in $O(\log N)$ time.

Paths

First, let's find the number of paths Justin can take that use exactly k turns, then iterate from 0 to K to find the final answer.

Assume Justin starts out driving east. He drives east for a bit, then turns and drives south, then turns and drives east, and so on. If he turns k times, then he will drive east $\lceil (k+1)/2 \rceil$ times and he will drive south $\lfloor k+1)/2 \rfloor$ times. Since he drives east and south a fixed number of blocks, this is the same as finding the number of ways to partition a set of N elements.