P2 solution.

문제 분석.

주어진 N개의 segment들에 있어, 그것들을 다시 하나로 합칠 때의 최소 시간(cost)를 구하는 것.
→ i번째(1≤i≤N) segment와 j번째(1≤j≤N) segment 사이를 합치는데에 드는 최소 시간을 **minCost[i][j]**라 하자. 그리고 i번째 segment까지의 누적 길이의 합을 **accumSum[i]**라 하자. 그렇다면 두 segment를 포함한 그 사이의 segment들의 합을 **betweenSum[i][j] = accumSum[j] - accumSum[i-1]**로 표현할 수 있다.
→ 연속된 segment 2개를 합치는 데에 드는 시간 mincost[i][i+1]는 betweenSum[i][i+1]으로 알 수 있고, 해당 값을 구하면 연속된 segment 3개에 합치는데에 드는 시간을 구할 수 있다.
→ k를 segment 구간을 두 개로 나누는 기준(i≤k≤j)으로 두면 minCost[i][i+2]는 minCost[i][k] + mincost[k+1][i+2] + betweenSum[i][i+2]로 구할 수 있고, 이러한 과정을 반복하여 segment N개를 합치는 최소 시간을 구한다.

솔루션.

1. N과 N개의 segment들을 입력받아 cableSegments라는 array를 생성한다.
2. 각 segment까지의 누적합을 표현하는 accumSum을 array를 생성한다.
3. segment끼리의 결합시에 발생하는 최소 시간을 저장하는 matrix, minCostFromTo를 생성한다. 이 때, minCostFromTo[i][j]는 i번째 segment부터 j번째 segment까지의 최소 결합 시간을 의미한다.
4. segment의 결합의 수가 두 개 이상인 것부터 마지막 n까지 아래의 과정을 반복한다. -→ **O(N)**
    a. 첫 번째 segment부터 결합이 마지막 N번째 segment까지 이뤄질 수 있을 때까지 아래의 과정을 반복한다. -→ **O(N)**
        i. minCostFromTo[i][j]를 구하는데에 있어, j는 i번째 segment로부터 결합되는 범위에 포함되는 마지막 segment의 위치를 의미한다.
        ii. minCostFromTo[i][j]의 기존값과 minCostFromTo[i][k] + minCostFromTo[k+1][j] + betweenSum[i][j]의 값을 비교하여 최소값을 업데이트한다. 이 과정을 i≤k≤j-1 동안 반복한다. → **O(N)** (최악의 경우)
5. minCostFromTo[0][n-1]을 출력한다.

시간 복잡도. 주어진 segment의 총 개수 N에 대하여 O(N^3)만큼의 시간 복잡도를 갖는다.