Kaung Khant Lin
6540131
542

Divine & Conquer

Problem A

- O(x), where x is exponential number
- O(log(x))

Problem B

- Splitting the data exactly in the middle keeps the work quick, giving you the best speed of O(n log n) because the process only repeats O(log n) times. If you only split off one item at a time, the process takes too long (O(n) repeats), making the overall speed much slower.
- No, it isn't sufficient enough to assume that the answer for the sequence is max(ma, mb).
- Yes, maxSum of crossing subsequence may give the larger sum than the one in step 3.
- The crossing subsequence must:
    - Start somewhere in the left half (index between i and mid)
    - End somewhere in the right half (index between mid+1 and k)
    - Must include both numbers[mid] and numbers[mid+1] (the elements immediately adjacent to the split point)
- In other words, it crosses the midpoint. The beginning index ≤ mid and the ending index ≥ mid+1.
- The sum of the maximum subsequence of the original subsequence (a_i,.., a_k) would be the maximum of three values:
    - The maximum sum found entirely in the left half (**m_L**).
    - The maximum sum found entirely in the right half (**m_R**).
    - The maximum crossing sum (**m_c**).
- **O(n)**
- **T(n) = O(n log n)**
- The divide-and-conquer algorithm is slower than Kadane's algorithm for solving the maximum subarray sum problem.
    - The **Divide-and-Conquer** approach has a time complexity of **O(n log n)** because it recursively splits the array and performs O(n) work at each of the log n levels to find crossing sums.
    - **Kadane's Algorithm** is faster, with a time complexity of **O(n)** (linear time), as it only requires a single pass over the array, tracking the current and absolute maximum sums without recursion or re-visiting elements.