hw3

Overview

We combine BFS with DFS to achieve efficient performance in dividing intervals. BFS is used to generate subproblems that are easily parallelizable, while DFS allows each processor to work on individual subproblems with lower memory overhead.

Algorithm Steps

1. **Initialize a Vector:**
    - Start by constructing a vector that contains the entire interval to be processed.
2. **BFS Subdivision:**
    - Use BFS on a single processor to repeatedly divide the interval into smaller sub-intervals.
    - Continue until the number of intervals exceeds the number of available processors.
3. **Distribute Intervals:**
    - Assign the intervals evenly across processors. If there are `n` intervals and `p` processors, each processor gets `n/p` intervals.
4. **Processor Work:**
    - Each processor runs DFS up to 100 times on its assigned intervals.
    - If the DFS does not finish processing the interval (e.g., stops early), the interval is returned for further subdivision.
5. **Iterative Refinement:**
    - Repeat steps 3 and 4, redistributing intervals that require further division, until no further subdivision is needed.

Result

Processors	MAX	Time	Speedup	Efficiency
1	3.33333	67	1.00	1.000
9	3.33333	7	9.57	1.063
18	3.33333	3	22.33	1.241
36	3.33333	1	67.00	1.861

The efficiency in this case seems to increase with the number of processors, suggesting that adding more processors leads to superlinear speedup.