Assignment 2: MINPEAKMEMORY

Himanshu Gangwal

April 14, 2024

1 Algorithm Overview

The algorithm utilizes dynamic programming principles to iteratively compute the minimum peak memory requirement for subsets of processes. It maintains two arrays: `pks` to store the minimum peak memory requirement for each subset, and `frwd` to store the forwarded memory from each subset to the undecided suffix. Additionally, it uses an `outs` array to store the sum of memory going out of each memory element to avoid recalculating it multiple times.

2 Calculations/ Update Steps

- **Forward Memory Calculation:** The forwarded memory (`frwd_n`) is computed by subtracting the inward memory (`inwd_n`) from the existing forwarded memory (`frwd`) and adding the memory going out of the new process (`outs[nwelmnt]`).
- **Peak Memory Update:** The peak memory requirement for the new subset is updated based on the existing peak memory requirement (`pks[thsset]`), the forwarded memory, and the memory of the new process. The maximum of the two is the new peak for such a configuration. Then, we take the minimum of this and (`pks[newset]`), i.e. update if we found a better(lower) maxpeak.(here, `newset=1 << nwelmnt) | thsset`).

3 Complexity

- **Time Complexity:** The algorithm iterates through all possible subsets of processes, resulting in a time complexity of $O(2^n \times n)$, where n is the number of processes.
- **Space Complexity:** It utilizes additional space to store the `pks`, `frwd`, and `outs` arrays, resulting in a space complexity of $O(2^n)$.