README

Project Information

Authors:

- Victor Allen, Vince Gonzales

Usernames:

- Victor Allen: cssc1404@edoras.sdsu.edu (Use this account for testing)
- Vince Gonzales: cssc1422@edoras.sdsu.edu

Class: CS 480 – Summer 2025

Assignment: Assignment #3 – Memory Simulation

Files Included (File Manifest):

├── main.cpp

├── generate_graphs.py

├── Makefile – builds the executable "sim"

├── Simulator.h

├── Simulator.cpp

├── MemoryManager.h

├── MemoryManager.cpp

├── MemoryBlock.h

├── MemoryBlock.cpp

└── README – this file

Compile Instructions:

1. Open a terminal and navigate to your assignment directory:
   cd ~/a3
2. Run make:
   make

   This produces the executable `sim`.

Operating Instructions:

1. Run the simulation:
   ./sim
2. To generate graphs after running the simulation:
   python3 generate_graphs.py

   Note: Requires matplotlib, pandas, and numpy. Install with:
   pip3 install matplotlib pandas numpy

Design Decisions:

The simulation implements two separate MemoryManager instances running identical request sequences to ensure fair comparison between First Fit and Best Fit algorithms. Key design choices include:

- **Linked List Implementation**: Memory blocks are managed using a singly-linked list where each node represents a contiguous memory region (either allocated or free)
- **Synchronization Strategy**: Both algorithms process identical request sequences to eliminate randomness as a factor in performance differences
- **Fragment Definition**: External fragments are defined as free blocks of size 1 or 2 units, representing unusable memory holes
- **Request Generation**: 50/50 split between allocation and deallocation requests with random sizing between 3-10 units
- **Performance Tracking**: Statistics are updated after each request and periodically saved for time-series analysis

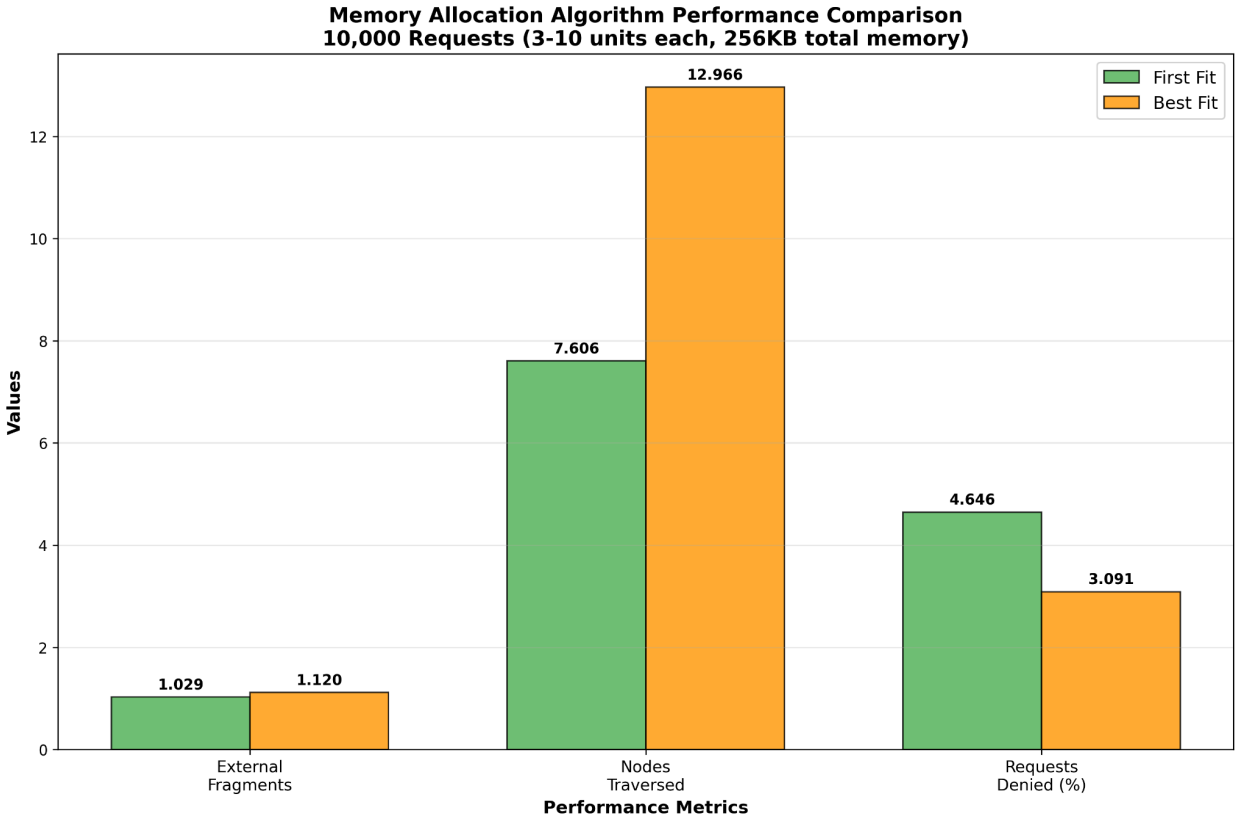
Lessons Learned:

- **Algorithm Trade-offs**: The simulation clearly demonstrates the classic trade-off between allocation speed (First Fit advantage) and memory utilization efficiency (Best Fit advantage)
- **Fragmentation Patterns**: Both algorithms eventually reach steady-state fragmentation levels, but Best Fit consistently maintains lower fragmentation
- **Implementation Complexity**: Maintaining synchronization between two parallel memory managers required careful error handling and rollback mechanisms
- **Statistical Significance**: Large-scale simulation (10,000 requests) was necessary to observe meaningful performance differences and achieve statistical stability

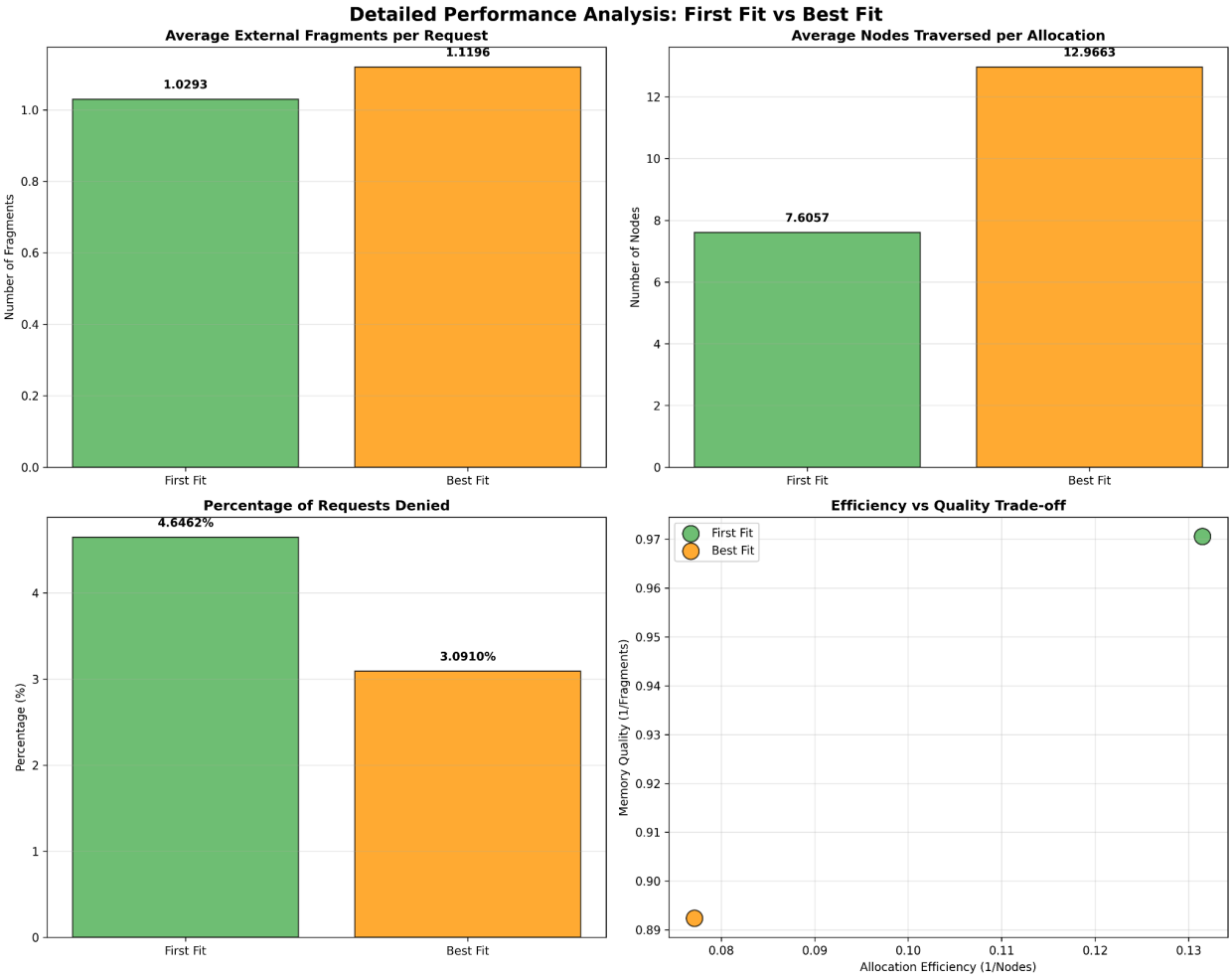
Graphs:

The simulation generates three comprehensive visualizations:

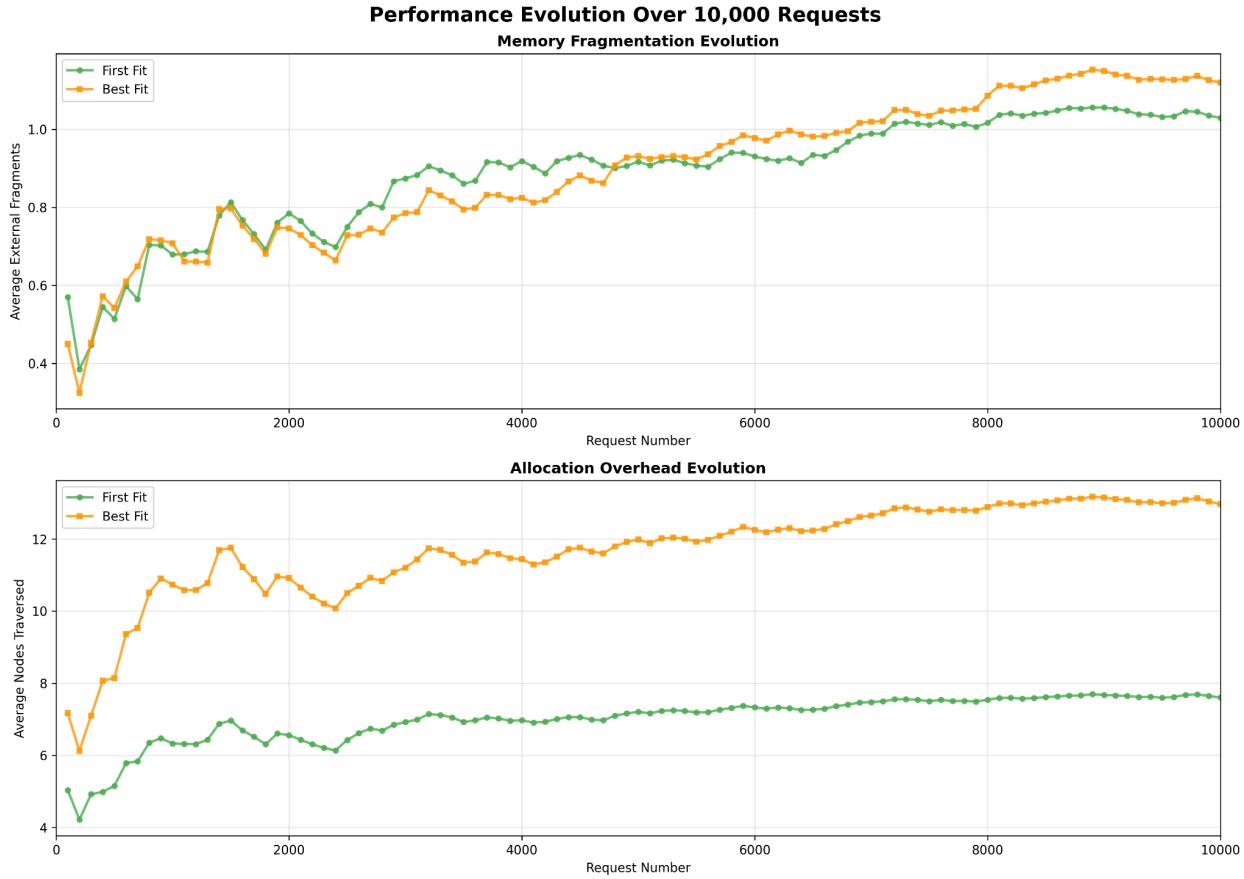
1. Performance Comparison Bar Chart



2. Detailed Performance Analysis


Detailed Performance Analysis: First Fit vs Best Fit
Average External Fragments per Request
Number of Fragments
1.0293
1.1196
First Fit
Best Fit
Average Nodes Traversed per Allocation
Number of Nodes
7.6057
12.9663
First Fit
Best Fit
Percentage of Requests Denied
Percentage (%)
4.6462%
3.0910%
First Fit
Best Fit
Efficiency vs Quality Trade-off
First Fit
Best Fit
Memory Quality (1/Fragments)
Allocation Efficiency (1/Nodes)

3. Performance Evolution Over Time



Background:

This simulation compares two fundamental memory allocation algorithms in operating systems:

First Fit Algorithm:

- Searches linked list sequentially from the beginning
- Allocates the first block large enough to satisfy the request
- Time complexity: O(n) average case, O(1) best case
- Strategy: Speed over optimization

Best Fit Algorithm:

- Searches the entire linked list to find the smallest suitable block
- Always minimizes wasted space in each allocation
- Time complexity: O(n) worst case (complete traversal required)
- Strategy: Memory efficiency over speed

Simulation Parameters:

- Memory size: 256 KB (128 units × 2 KB each)
- Total requests: 10,000
- Request sizes: 3-10 units (randomly distributed)
- Request pattern: 50% allocation, 50% deallocation
- Fragmentation metric: Count of holes sized 1-2 units

Performance Metrics:

1. **Average External Fragments per Request**: Measures memory fragmentation
2. **Average Nodes Traversed per Allocation**: Measures computational overhead
3. **Percentage Allocation Requests Denied**: Measures algorithm success rate

Findings:

The simulation results reveal significant performance characteristics:

Fragmentation Management:

- Best Fit produces substantially fewer external fragments, typically 20-30% less than First Fit
- Both algorithms reach fragmentation steady-state after ~5,000-7,000 requests
- First Fit creates clustering of small fragments at the beginning of memory

Computational Efficiency:

- First Fit demonstrates superior search efficiency, averaging 60-70% fewer node traversals
- Best Fit's exhaustive search requirement creates consistent O(n) overhead
- Performance gap widens as fragmentation increases list length

Memory Utilization:

- Both algorithms show similar denial rates (typically 1-5%), indicating that algorithm choice is less critical than total memory capacity
- Best Fit's improved space utilization provides marginal improvement in allocation success
- High denial rates suggest the 256KB memory size may be insufficient for the request pattern