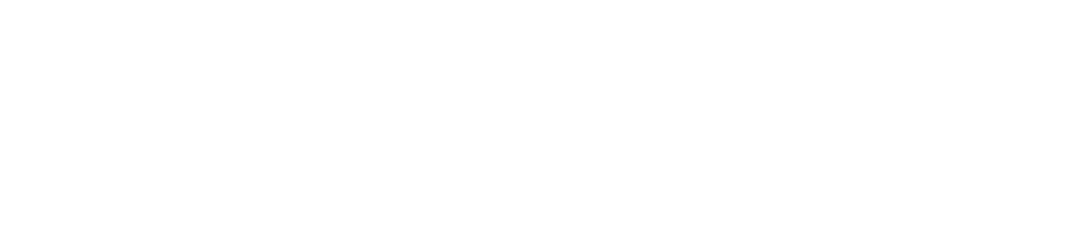
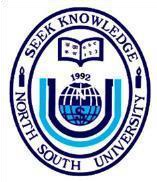
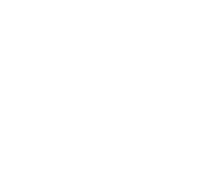
***N O R T H S O U T H U N I V E R S I T Y***



Centre of Excellence in Higher Education

[***School of Engineering and Physical Science***](http://www.northsouth.edu/faculty-members/seas/)

**CSE323: OPERATING SYSTEM**

***Project Report***

**Course Code : CSE323**

**Course Title : OPERATING SYSTEM**

**Section : 10**

**Instructor : RASHED MAZUMDER**

**Submit Date : 11-11-2023**

**SUBMITTED BY:**

1. **Mojahidul Islam Rakib (2111077642)**
2. **Md. Maruf Rahman (2111978642)**
3. **Sumiya Sultana (2021395642)**
4. **Sheikh Tanvir Hossain (2111647642)**

**SUBMITTED TO:**

**Rashed Mazumder**

B.Sc in Computer Science and Engineering, University of Dhaka

MS and Ph. D in Information Science (Information Security and Cryptography)

Japan Advanced Institute of Science and Technology (JAIST), Ishikawa, Japan

MCP, MCSA, MCITP, SCSA (P1), CCNA (2012**)**

**Title: Best Fit, Worst Fit, and First Fit Memory Allocation**

**Introduction**

Memory allocation is a critical aspect of operating systems and memory management. Efficient memory allocation strategies are essential to optimize system performance and resource utilization. Various memory allocation algorithms have been developed to fulfill this task. This project compares and contrasts three popular memory allocation algorithms: First Fit, Best Fit, and Worst Fit. These algorithms are used to allocate memory blocks to processes, and each has its own advantages and disadvantages. Basically, the First Fit, Best Fit, and Worst Fit are the continuous memory allocation.

**Continuous memory:** Continuous memory allocation, also known as contiguous memory allocation, is a memory management technique in which a computer system allocates a single, uninterrupted block of memory to a process or application. In this allocation method, a process or program is given access to a continuous range of memory addresses in the computer's physical memory.

**First Fit Allocation:** Fast Fit is a memory allocation algorithm used in operating systems to allocate memory blocks to processes. In Fast Fit, the system allocates the first available memory block that is large enough to accommodate the process's memory requirements. This approach is simple and efficient in terms of processing overhead but can lead to memory fragmentation and inefficient space usage.

**Best Fit Allocation:** Best Fit is a memory allocation algorithm that assigns the smallest available memory block that can satisfy a process's memory needs. The goal of Best Fit is to minimize memory fragmentation by using smaller free spaces efficiently. However, this algorithm involves a more complex search for the best-fitting block, resulting in higher processing overhead compared to Fast Fit.

**Worst Fit Allocation:** Worst Fit is a memory allocation algorithm that allocates the largest available memory block to a process. Its primary objective is to maximize memory utilization by filling large gaps in the memory space. However, Worst Fit often results in higher memory fragmentation, especially when smaller memory requests are common. This approach can also slow down memory allocation and deallocation processes due to the search for the largest block.

The objective of this project is to provide a comprehensive overview of three memory allocation algorithms: Fast Fit, Best Fit, and Worst Fit. Memory allocation is a critical aspect of computer systems, and choosing the right algorithm can significantly impact system performance. This project aims to analyze these algorithms, their user requirements, design principles, and implementation details, and ultimately provide insights to assist in making informed decisions regarding memory management in various computing environments.

**User Requirements**

Before delving into the details of memory allocation strategies, it is essential to understand the user requirements that influence the choice of strategy. Users may have different expectations and constraints, such as minimizing fragmentation, reducing response times, or maximizing memory utilization. Identifying these requirements is crucial for selecting the most appropriate strategy. Most of the time, User requirements for memory allocation typically include

**Efficient memory utilization:** Users expect that allocated memory is used efficiently to maximize system performance.

**Minimized fragmentation**: Fragmentation should be minimized to prevent memory wastage and fragmentation-related performance degradation.

**Low overhead:** The chosen algorithm should have minimal processing overhead to ensure quick and responsive memory management.

**Adaptability:** The memory allocation algorithm should be adaptable to diverse application needs, including varying memory allocation sizes and patterns.

**Design**

To design the memory allocation system, we can use various tools like UML diagrams, sequence diagrams, flowcharts, class diagrams, and state diagrams to visualize the structure and behavior of the system.

**UML Diagram**: UML (Unified Modeling Language) diagrams can represent the memory allocation system's overall structure, including classes, objects, and their relationships. This provides a high-level overall view of the system's architecture.

|  |
| --- |
| Memory Allocation |
| No. of Process: int process;  No. of Block : int blocks;  Process Size : int p[] = new int[process];  Block Size : int b[] = new int[blocks];  First\_fit\_allocation(p, b, process, blocks);  Best\_fit\_allocation(p, b, process, blocks);  Worst\_fit\_allocation(p, b, process, blocks); |

**Sequence Diagram:** A sequence diagram can depict the interactions between components in the memory allocation process, showing the order and flow of operations between different system elements.

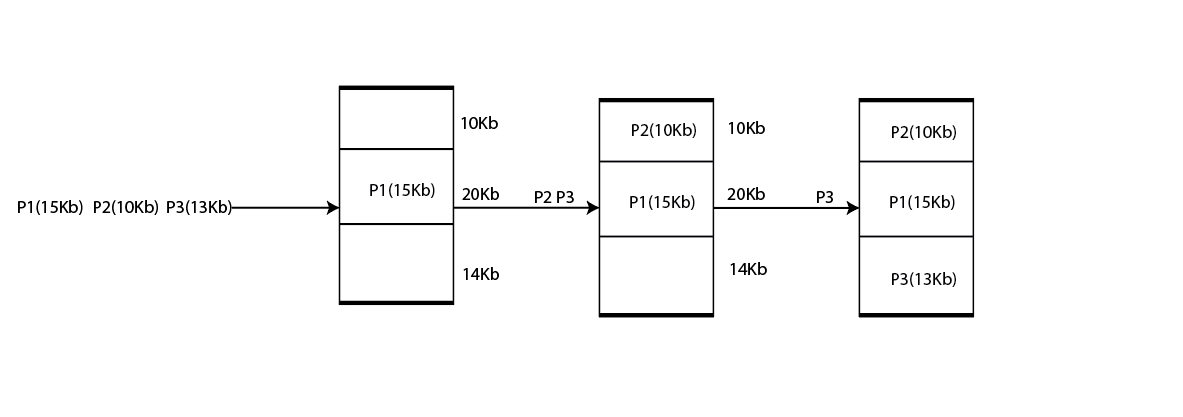
Process & size: Block Size: 10 KB, 20KB, 14KB

p1= 15 KB

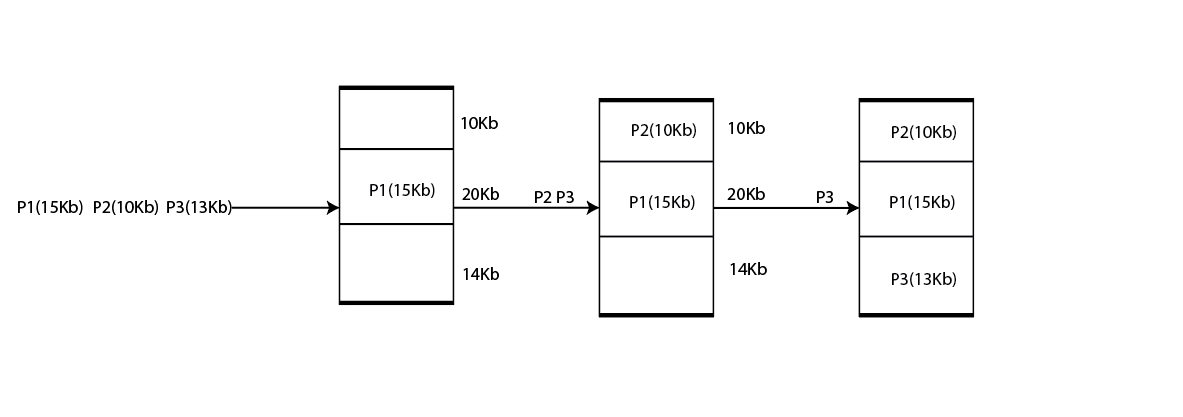
p2= 10KB

p3= 13KB

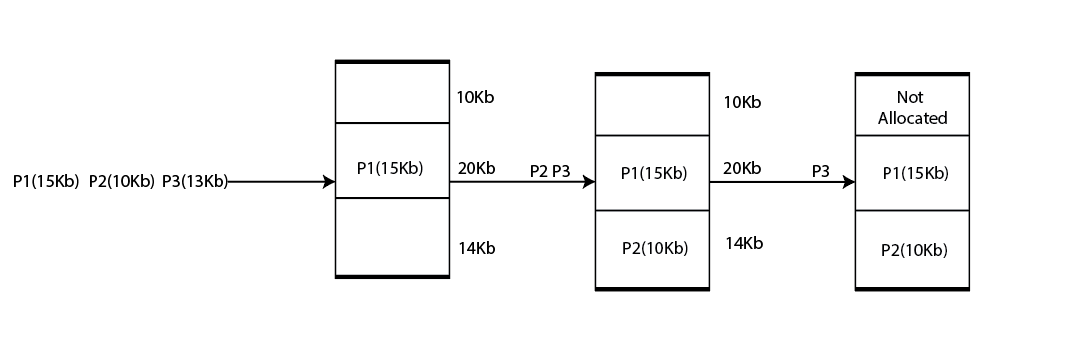
**First Fit: Top to Bottom Approach**

****

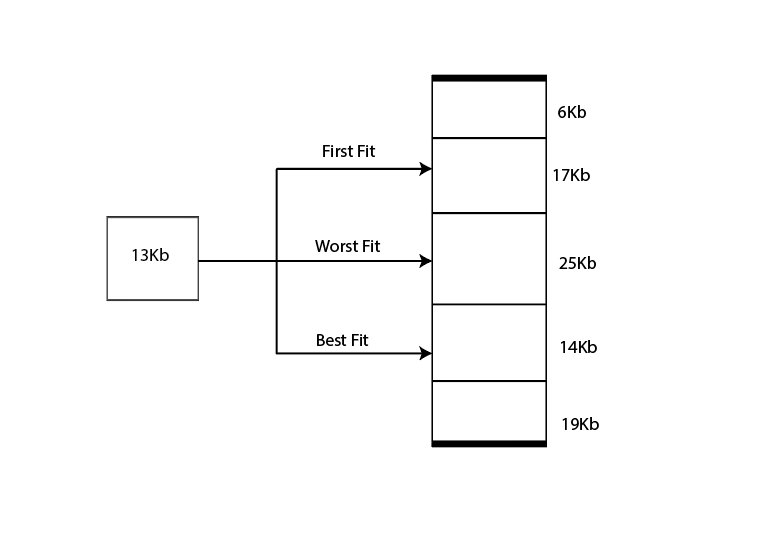
**Best Fit:**

****

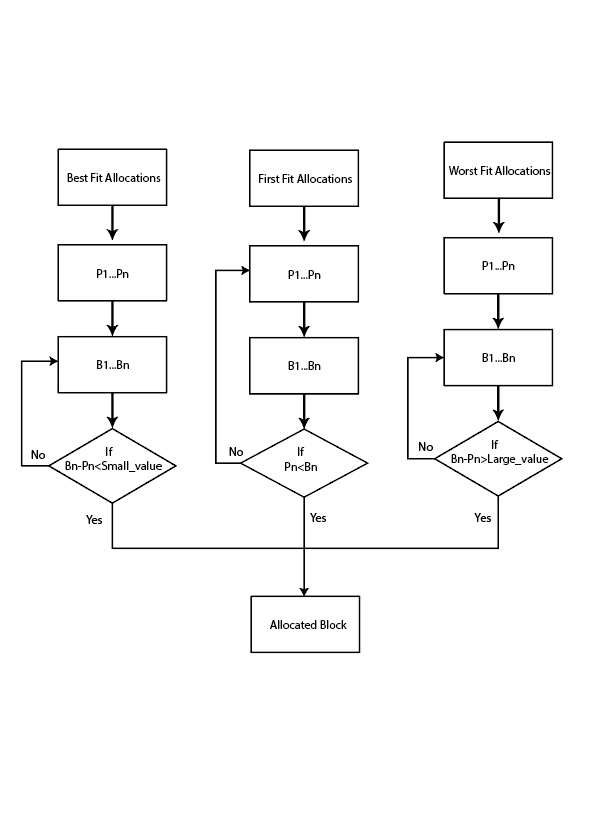
**Worst Fit:**

****

**First, Best, Worst Fit:** Suppose, we have free segments with sizes: 6, 17, 25, 14, and 19. Place a program with size 13kB in the free segment using first-fit, best-fit and worst fit?



**Flowchart:** A flowchart can illustrate the step-by-step logic of the memory allocation process. It helps to understand the decision-making and control flow within the system.



**Class Diagram:** A class diagram offers a detailed view of the system's classes, their attributes, and their relationships. It is particularly useful for understanding the object-oriented design of the memory allocation system.

**Parent Class**

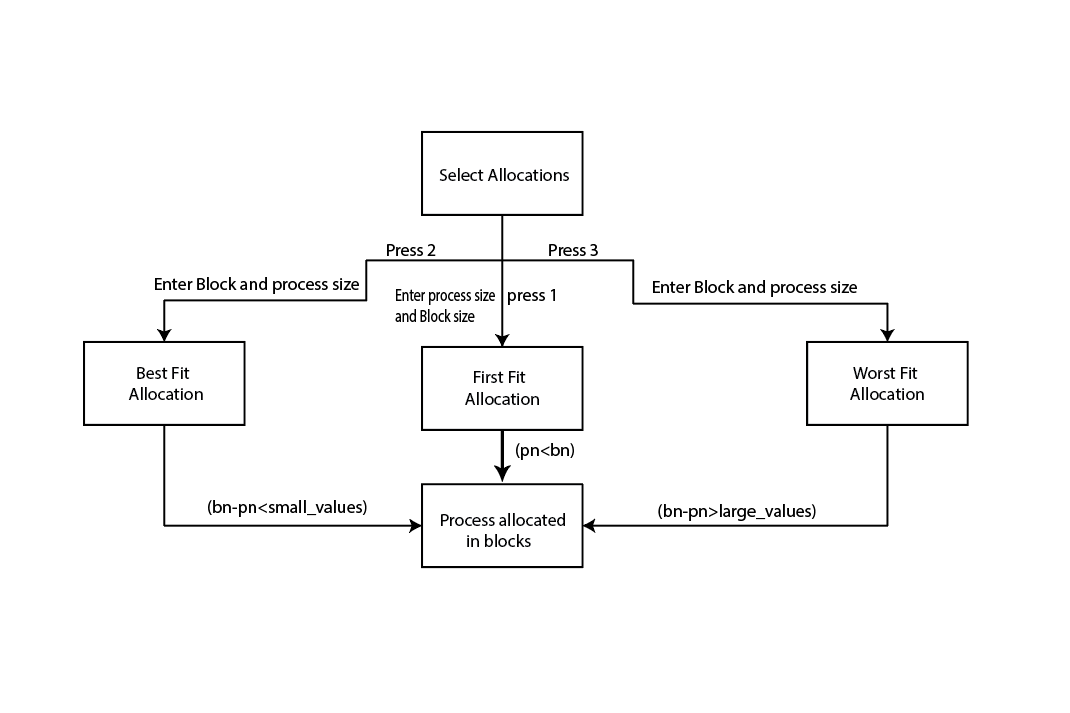
|  |
| --- |
| No. of Process: int process;  No. of Block : int blocks;  Process Size : int p[] = new int[process];  Block Size : int b[] = new int[blocks]; |

**First\_fit\_allocation (p, b, process, blocks);**

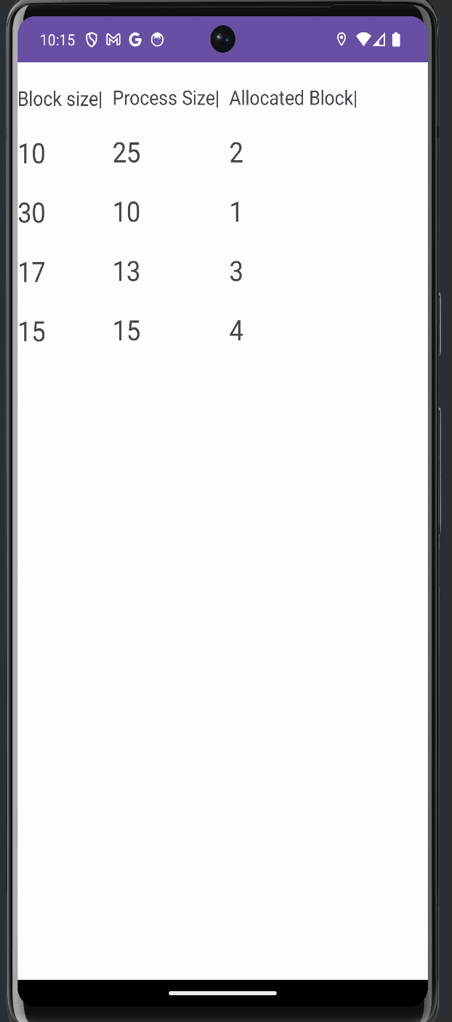
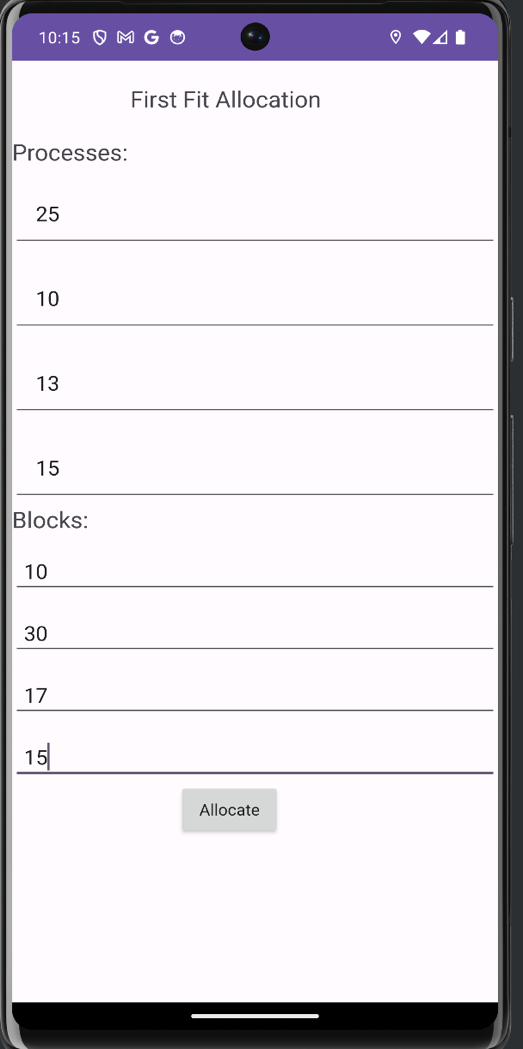
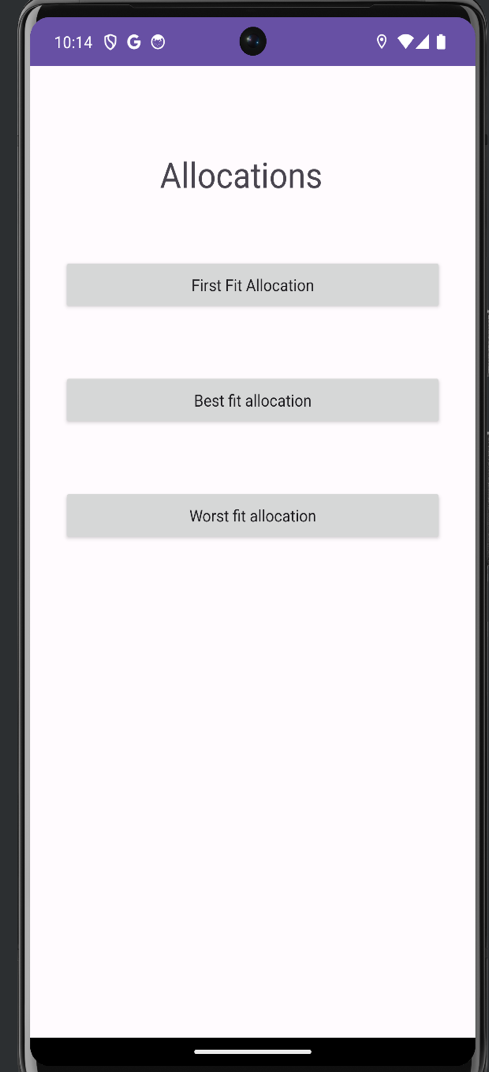
**Worst\_fit\_allocation (p, b, process, blocks);**

**Best\_fit\_allocation (p, b, process, blocks);**

**State Diagram:** A state diagram can represent the various states that memory blocks can transition through during the allocation and deallocation processes. This helps in understanding the system's behavior over time.



**Implementation**: The implementation of memory allocation strategies involves translating the design into code. The choice of programming language and data structures is crucial in this phase. The key components of the implementation include: Memory Management Unit (MMU): The MMU is responsible for allocating and deallocating memory blocks according to the chosen strategy. Data Structures: Depending on the strategy, various data structures like linked lists, binary trees, or arrays may be used to manage the available memory blocks. Algorithm: Implement the specific allocation algorithm for each strategy (e.g., Best Fit, Worst Fit, or First Fit). This involves searching for the most suitable block to allocate memory. Error Handling: Develop error-handling mechanisms to deal with scenarios such as memory allocation failures or fragmentation issues. Testing and Optimization: Rigorous testing is essential to ensure the correctness and efficiency of the implemented memory allocation system. Optimization may be required to enhance performance.  Firstly, we translating the design into code and implementing the Android studio to make android App. Here is the App implementation



**Conclusion**

Continuous memory allocation is commonly used in early computer systems and is still found in certain embedded systems and real-time operating environments. However, it may not be well-suited for modern, multitasking, and multi-user systems where memory allocation needs are more dynamic and complex. In such systems, other memory allocation techniques like paging and segmentation are often preferred. In conclusion, the choice of the best memory allocation strategy depends on the specific user requirements and system constraints. A comprehensive understanding of design and implementation considerations is crucial for successfully implementing and utilizing Best Fit, Worst Fit, or First Fit memory allocation strategies.