南京信息工程大学 实验（实习）报告

实验名称 最佳适应算法 日期 2023.11.30指导教师 赵晓平

专业信息安全年级班级 21奇安信姓名朱宸扬学号 202183760012

1. 实验目的

深入理解最佳适应算法

1. 实验内容

模拟最佳适应算法

1. 实验原理

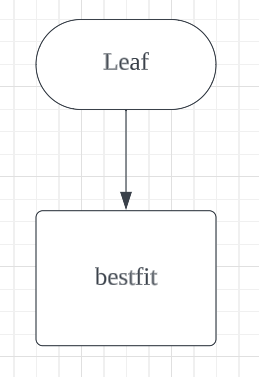
最佳适应算法

1. 实验设计及编码
2. 模块分析

进程，有自己的工作量

页面，最佳适应算法

1. 流程图



1. 代码实现

import pygame

import sys

class MemoryBlock:

def \_\_init\_\_(self, size):

self.size = size

self.is\_allocated = False

self.process\_id = None

def visualize\_memory(memory\_blocks):

pygame.init()

block\_height = 50

# screen\_width = sum(block.size + margin for block in memory\_blocks) + margin

screen\_width = 820

screen\_height = block\_height + 70

screen = pygame.display.set\_mode((screen\_width + 200, screen\_height))

pygame.display.set\_caption('Memory Visualization')

x\_position = 0 # 将x\_position移到循环外

screen.fill((255, 255, 255))

for block in memory\_blocks:

color = (100, 100, 100) if block.is\_allocated else (255,255,255)

font = pygame.font.Font(None, 24)

text\_surface = font.render("White: Free, Grey: Allocated", True, (0, 0, 0))

text\_rect1 = text\_surface.get\_rect()

text\_rect1.center = (600, 30)

# 将方块底部对准下方

pygame.draw.rect(screen, color, (x\_position, screen\_height - block\_height, block.size, block\_height))

font = pygame.font.Font(None, 36)

text = font.render(str(block.size), True, (0, 0, 0))

text\_rect = text.get\_rect(center=(x\_position + block.size / 2, screen\_height - block\_height / 2))

screen.blit(text, text\_rect)

screen.blit(text\_surface, text\_rect1)

#

x\_position += block.size

pygame.display.flip()

for event in pygame.event.get():

if event.type == pygame.QUIT:

pygame.quit()

sys.exit()

pygame.time.wait(1000)

def best\_fit(memory\_blocks, process\_size):

best\_fit\_block = None

for block in memory\_blocks:

if not block.is\_allocated and block.size >= process\_size:

if best\_fit\_block is None or block.size < best\_fit\_block.size:

best\_fit\_block = block

# 在memory\_blocks里找一个最佳的空位，best\_fit\_block为这个空位

temp\_size = best\_fit\_block.size

if best\_fit\_block is not None:

# 分配给进程

best\_fit\_block.is\_allocated = True

best\_fit\_block.size = process\_size

best\_fit\_block.process\_id = "Process" # 模拟进程ID

# 创建新的空块

remaining\_size = temp\_size - process\_size

if remaining\_size > 0:

remaining\_block = MemoryBlock(remaining\_size)

index = memory\_blocks.index(best\_fit\_block)

memory\_blocks.insert(index + 1, remaining\_block)

return best\_fit\_block

else:

return None

def deallocate(memory\_blocks, process\_id):

for i, block in enumerate(memory\_blocks):

if block.is\_allocated and block.process\_id == process\_id:

# 释放进程占用的内存块

block.is\_allocated = False

block.process\_id = None

# 合并相邻的空块

if i < len(memory\_blocks) - 1 and not memory\_blocks[i + 1].is\_allocated:

block.size += memory\_blocks[i + 1].size

del memory\_blocks[i + 1]

if i > 0 and not memory\_blocks[i - 1].is\_allocated:

block.size += memory\_blocks[i - 1].size

del memory\_blocks[i - 1]

break

def print\_memory\_status(memory\_blocks):

for i, block in enumerate(memory\_blocks):

status = "Allocated" if block.is\_allocated else "Free"

print(f"Block {i + 1}: Size = {block.size}, Status = {status}, Process ID = {block.process\_id}")

def Pallocation(size,pool):

allocated\_block = best\_fit(pool, size)

if allocated\_block is not None:

print(f"Allocated {size} units to Block {memory\_blocks.index(allocated\_block) + 1}")

else:

print(f"Unable to allocate {size} units")

visualize\_memory(pool)

print("\n")

def PDeallocation(index, pool):

if 0 <= index < len(pool):

block\_to\_deallocate = pool[index]

if block\_to\_deallocate.is\_allocated:

block\_to\_deallocate.is\_allocated = False

block\_to\_deallocate.process\_id = None

# 合并相邻的空块

if index < len(pool) - 1 and not pool[index + 1].is\_allocated:

block\_to\_deallocate.size += pool[index + 1].size

del pool[index + 1]

if index > 0 and not pool[index - 1].is\_allocated:

pool[index - 1].size += block\_to\_deallocate.size

del pool[index]

print(f"Deallocated Block {index + 1}")

else:

print(f"Block {index + 1} is already Free")

visualize\_memory(pool)

else:

print(f"Invalid index: {index}")

if \_\_name\_\_ == "\_\_main\_\_":

# 初始化内存块

memory\_blocks = [MemoryBlock(1000)]

# 打印初始内存状态

print("Initial Memory Status:")

print\_memory\_status(memory\_blocks)

print("\n")

Pallocation(50,memory\_blocks)

Pallocation(200,memory\_blocks)

Pallocation(300,memory\_blocks)

PDeallocation(1,memory\_blocks)

Pallocation(100,memory\_blocks)

Pallocation(200,memory\_blocks)

1. 结果及其相关分析（结果必须是图示）

另附

1. 实验小结

加深了对最佳适应算法的理解