内存分配模拟报告

1.开发环境

- 系统：Windows 11
- 软件：Visual Studio 2022
- 语言：C#

2.项目需求

在计算机系统中，内存分配是一项重要的任务。通过合理的内存管理，可以提高系统的性能和资源利用率。不同的内存分配算法具有各自的优势和适用场景，因此了解和比较这些算法对于优化内存使用至关重要。

2.1.首次适应算法

首次适应算法是一种常见的内存分配算法，它的基本思想是在内存中从头开始查找第一个满足分配要求的空闲内存块，并将其分配给请求进程。因此，这个算法的名称也叫做"first fit"算法。

首次适应算法的实现比较简单，它只需要从内存的起始位置开始扫描，寻找第一个满足大小要求的空闲块。当找到一个空闲块时，就将其分配给请求进程，并将剩余的空间留作下一次分配。

虽然首次适应算法比较简单，但是它也存在一些问题。例如，它可能会导致内存碎片问题，即一些小的空闲块散布在内存中，无法满足大块内存的分配请求，从而降低内存利用率。此外，由于首次适应算法只考虑了内存空间的起始位置，因此它可能会导致低地址部分的内存空间被占满，而高地址部分的内存空间却很少被使用。

尽管存在这些问题，首次适应算法仍然被广泛应用于许多操作系统和应用程序中，因为它的实现简单、速度较快，并且能够满足一般的内存分配需求。

2.2.最佳适应算法

最佳适应算法是一种常见的内存分配算法，它的基本思想是在内存中查找最小可用的空闲块来满足请求，并将其分配给请求的进程。因此，这个算法的名称也叫做"best fit"算法。

最佳适应算法的实现需要在内存空间中查找最小可用空闲块。为了实现这个目标，算法需要遍历整个内存空间，找到大小最小且能够满足请求的空闲块。当找到一个满足要求的空闲块时，就将其分配给请求进程，并将剩余的空间留作下一次分配使用。

相比于首次适应算法，最佳适应算法可以更好地利用内存空间，因为它总是选择最小的可用空闲块来满足请求。但是，由于需要遍历整个内存空间，最佳适应算法的时间复杂度相对较高，可能会降低系统的性能。

3.项目目标

假设初始态下，可用内存空间为640KB，请分别用首次适应算法和最佳适应算法进程内存块的分配和回收，并显示出每次分配和回收后的空闲分区链的情况来。

本项目通过模拟和展示首次适应算法和最佳适应算法的内存分配过程，帮助我们深入理解这两种算法的工作原理、优势与劣势和适用场景。通过可视化的方式，我们可以更直观地观察和比较不同算法对内存分配的影响，从而帮助我们理解实际计算机系统中的内存管理方式。

4.代码实现

4.1.可视化窗体代码实现

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Drawing.Text;
using System.Linq;
using System.Runtime.InteropServices;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace 内存管理
{
    public partial class Form1 : Form
    {

        private int initialMemorySize;
        private GroupBox memoryParameterControl;
        public NumericUpDown initialMemorySizeNumeric;
        private Button memorySizeModifyButton;

        private int requestedMemorySize;
        private GroupBox requestMemoryControl;
        public NumericUpDown requestedMemorySizeNumeric;
        public ComboBox requestChoice;
        private Button memoryRequestButton;

        private GroupBox removeTaskControl;
        public NumericUpDown removeTaskNumNumeric;
        public ComboBox removeChoice;
        private Button removeTaskButton;

        private Size formSize;

        private GroupBox firstFitGroupBox;
        private RichTextBox firstFitBox;
        private Label firstFitBoxLabel;
        private Label firstFitSpaceLabel;
        int[] firstFitSpace;
        private GroupBox bestFitGroupBox;
        private RichTextBox bestFitBox;
        private Label bestFitBoxLabel;
        private Label bestFitSpaceLabel;
        int[] bestFitSpace;

        int ffnum = 1;
        int bfnum = 1;

        List<MemoryBlock> firstFitMemoryBlocks;
        List<MemoryBlock> bestFitMemoryBlocks;

        public Form1(int initialMemorySize = 640)
        {
            this.initialMemorySize = initialMemorySize;
            this.requestedMemorySize = 0;
            this.formSize = new Size(Math.Max(initialMemorySize + 44, 520),
800);
            this.Load += Form1_Load;
            this.AutoScroll = true;
            this.FormClosed += Form1_FormClosed;
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            this.Size = formSize;

            memoryParameterControl = new GroupBox
            {
                Text = "总内存大小控制",
                Parent = this,
                Font = new Font("Consolas", 12),
                Location = new Point(10, 10),
                Size = new Size(240, 80)
            };
            memoryParameterControl.BringToFront();

            initialMemorySizeNumeric = new NumericUpDown
            {
                Minimum = 480,
                Maximum = 800,
                Value = initialMemorySize,
                Parent = memoryParameterControl,
                Location = new Point(10, 30),
                Enabled = true,
                ReadOnly = false,
            };

            memorySizeModifyButton = new Button
            {
                Text = "修改",
                Parent = memoryParameterControl,
                AutoSize = true,
                Font = new Font("宋体", 12),
                Location = new Point(150, 30),
            };
            memorySizeModifyButton.Click += initialMemorySizeModify;

            requestMemoryControl = new GroupBox
            {
                Text = "内存空间申请",
                Parent = this,
                Font = new Font("Consolas", 12),
                Location = new Point(10, 110),
                Size = new Size(240, 120)
            };
            requestMemoryControl.BringToFront();

            requestedMemorySizeNumeric = new NumericUpDown
            {
                Value = requestedMemorySize,
                Parent = requestMemoryControl,
                Location = new Point(10, 30),
                Size = new Size(130, 30),
                Minimum = 0,
                Maximum = initialMemorySize,
            };

            requestChoice = new ComboBox
            {
                Text = "选择算法",
                Parent = requestMemoryControl,
                Location = new Point(10, 70),
                Size = new Size(130, 30),
            };
            requestChoice.Items.Add("首次适应算法");
            requestChoice.Items.Add("最佳适应算法");

            memoryRequestButton = new Button
            {
                Text = "申请",
                Parent = requestMemoryControl,
                AutoSize = true,
                Font = new Font("宋体", 12),
                Location = new Point(150, 68),
            };
            memoryRequestButton.Click += memoryRequest;

            removeTaskControl = new GroupBox
            {
                Text = "要释放的任务的编号",
                Parent = this,
                Font = new Font("Consolas", 12),
                Location = new Point(270, 110),
                Size = new Size(240, 120)
            };
            removeTaskControl.BringToFront();

            removeTaskNumNumeric = new NumericUpDown
            {
                Value = 0,
                Parent = removeTaskControl,
                Location = new Point(10, 30),
                Minimum = 1,
                Maximum = initialMemorySize,
                Size = new Size(130, 30),
            };

            removeChoice = new ComboBox
```

```
{
    Text = "选择算法",
    Parent = removeTaskControl,
    Location = new Point(10, 70),
    Size = new Size(130, 30),
};
removeChoice.Items.Add("首次适应算法");
removeChoice.Items.Add("最佳适应算法");
removeTaskButton = new Button
{
    Text = "释放",
    Parent = removeTaskControl,
    AutoSize = true,
    Font = new Font("宋体", 12),
    Location = new Point(150, 68),
};
removeTaskButton.Click += removeRequest;

firstFitMemoryBlocks = new List<MemoryBlock>();
bestFitMemoryBlocks = new List<MemoryBlock>();

firstFitSpace = new int[this.initialMemorySize + 5];
bestFitSpace = new int[this.initialMemorySize + 5];
for (int i = 0; i < this.initialMemorySize; i++)
{
    firstFitSpace[i] = 0;
}
for (int i = 0; i < this.initialMemorySize; i++)
{
    bestFitSpace[i] = 0;
}

firstFitGroupBox = new GroupBox
{
    Location = new Point(10, 240),
    Parent = this,
    Text = "首次适应算法",
    Font = new Font("Consolas", 12),
    Size = new Size(this.formSize.Width - 20, 250),
};
firstFitGroupBox.BringToFront();

firstFitBoxLabel = new Label
{
    Parent = firstFitGroupBox,
    BackColor = Color.White,
    BorderStyle = BorderStyle.Fixed3D,
    Location = new Point(10, 25),
    Size = new Size(firstFitGroupBox.Width - 20, 100),
};

firstFitBox = new RichTextBox
{
    Text = "",
    Parent = firstFitGroupBox,
                BorderStyle = BorderStyle.None,
                Location = new Point(10, 135),
                Size = new Size(firstFitGroupBox.Width - 20, 100),
            };

            bestFitGroupBox = new GroupBox
            {
                Location = new Point(10, 500),
                Parent = this,
                Text = "最佳适应算法",
                Font = new Font("Consolas", 12),
                Size = new Size(this.formSize.Width - 20, 250),
            };
            bestFitGroupBox.BringToFront();

            bestFitBoxLabel = new Label
            {
                Parent = bestFitGroupBox,
                BackColor = Color.White,
                BorderStyle = BorderStyle.Fixed3D,
                Location = new Point(10, 25),
                Size = new Size(bestFitGroupBox.Width - 20, 100),
            };

            bestFitBox = new RichTextBox
            {
                Text = "",
                Parent = bestFitGroupBox,
                BorderStyle = BorderStyle.None,
                Location = new Point(10, 135),
                Size = new Size(bestFitGroupBox.Width - 20, 100),
            };
        }
        void initialMemorySizeModify(object sender, EventArgs e)
        {
            int newSize = int.Parse(initialMemorySizeNumeric.Value.ToString());
            MessageBox.Show(newSize.ToString());
            if (newSize == initialMemorySize)
            {
                MessageBox.Show("总内存空间未改变! ");
                return;
            }
            Form1 form = new Form1(newSize);
            Hide();//隐藏旧窗体
            form.TopMost = true;
            form.Show();
        }
        void memoryRequest(object sender, EventArgs e)
        {
            int[] ffsum = new int[this.initialMemorySize + 1];
            int[] bfsum = new int[this.initialMemorySize + 1];
            int size = int.Parse(requestedMemorySizeNumeric.Value.ToString());
            string choice = requestChoice.Text.ToString();
            for (int i = 0; i <= this.initialMemorySize; i++)
            {
                ffsum[i] = (i > 0 ? firstFitSpace[i - 1] + ffsum[i - 1] : 0);
                bfsum[i] = (i > 0 ? bestFitSpace[i - 1] + bfsum[i - 1] : 0);
            }
            if (choice == "首次适应算法")
            {
                for (int i = 0; i < this.initialMemorySize; i++)
                {
                    if (i + size > this.initialMemorySize)
                    {
                        firstFitBox.AppendText("没有足够的可分配空间! \n");
                        break;
                    }
                    if (ffsum[i + size] - ffsum[i] == 0)
                    {
                        for (int j = i; j < i + size; j++)
                        {
                            firstFitSpace[j] = 1;
                        }
                        MemoryBlock ffmemoryBlock = new MemoryBlock(ffnum, size,
firstFitBoxLabel.Height, 12 + i, 25, this.firstFitGroupBox);
                        ffmemoryBlock.BringToFront();
                        firstFitMemoryBlocks.Add(ffmemoryBlock);
                        firstFitBox.AppendText($"编号为: {ffnum}的任务分配完成!
\n");
                        ffnum++;
                        break;
                    }
                }
            }
            else if(choice == "最佳适应算法")
            {
                int bestX = -1, minFitSpace = 1234567;
                for(int i = 0; i < this.initialMemorySize; i++)
                {
                    int space = 1234567, end = 0;
                    for (int j = i + size; j < this.initialMemorySize; j++)
                    {
                        if (bfsum[i + size] - bfsum[i] != 0)
                        {
                            break;
                        }
                        space = j - i;
                        end = j;
                    }
                    if(space < minFitSpace)
                    {
                        bestX = i;
                        minFitSpace = space;
                        i = end;
                    }
                }
                if(bestX == -1)
                {
                    bestFitBox.AppendText("没有足够的可分配空间! \n");
                    return;
                }
                else
                {
                    for(int i = bestX; i < bestX + size; i++)
                    {
                        bestFitSpace[i] = 1;
                    }
                    MemoryBlock memoryBlock = new MemoryBlock(bfnum, size,
bestFitBoxLabel.Height, 12 + bestX, 25, bestFitGroupBox);
                    memoryBlock.BringToFront();
                    bestFitMemoryBlocks.Add(memoryBlock);
                    bestFitBox.AppendText($"编号为: {bfnum}的任务分配完成! \n");
                    bfnum++;
                }
            }
            else
            {
                MessageBox.Show("未选择需要使用的算法! ");
            }
        }
        void removeRequest(object sender, EventArgs e)
        {
            string choice = removeChoice.Text;
            int num = int.Parse(removeTaskNumNumeric.Value.ToString());
            if(choice == "首次适应算法")
            {
                int pos = -1;
                for(int i = 0; i < firstFitMemoryBlocks.Count(); i++)
                {
                    if (firstFitMemoryBlocks[i].getNum() == num)
                    {
                        pos = i;
                        break;
                    }
                }
                if(pos == -1)
                {
                    MessageBox.Show($"编号为{num}的任务不存在! ");
                    return;
                }
                int start = firstFitMemoryBlocks[pos].getPosX();
                int size = firstFitMemoryBlocks[pos].getWidth();
                for (int j = start; j < start + size; j++)
                {
                    firstFitSpace[j] = 0;
                }
                firstFitMemoryBlocks[pos].remove();
                firstFitMemoryBlocks.RemoveAt(pos);


            }
            else if(choice == "最佳适应算法")
            {
                int pos = -1;
                for (int i = 0; i < bestFitMemoryBlocks.Count(); i++)
```

```
                    {
                        if (bestFitMemoryBlocks[i].getNum() == num)
                        {
                            pos = i;
                            break;
                        }
                    }
                    if (pos == -1)
                    {
                        MessageBox.Show($"编号为{num}的任务不存在！");
                        return;
                    }
                    int start = bestFitMemoryBlocks[pos].getPosX();
                    int size = bestFitMemoryBlocks[pos].getWidth();
                    for (int j = start; j < start + size; j++)
                    {
                        bestFitSpace[j] = 0;
                    }
                    bestFitMemoryBlocks[pos].remove();
                    bestFitMemoryBlocks.RemoveAt(pos);
                }
                else
                {
                    MessageBox.Show("未选择需要使用的算法！");
                }
            }
            private void Form1_FormClosed(object sender, FormClosedEventArgs e)
            {
                Application.Exit();//释放内存
            }
        }
}
```

4.2.内存块实现

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Drawing.Text;
using System.Linq;
using System.Runtime.InteropServices;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace 内存管理
{
    class MemoryBlock
    {
        private ToolTip blockInfo;
        private Label block;
        private int num;
        private int height = 80;
        private int width;
        private Point loc;
        private GroupBox parent;
        public MemoryBlock(int num, int width, int height, int pos_x, int pos_y,
GroupBox parent)
        {
            this.parent = parent;
            this.num = num;
            this.width = width;
            this.height = height;
            loc = new Point(pos_x, pos_y);

            block = new Label
            {
                Name = parent.Text + num.ToString(),
                Text = num.ToString(),
                Size = new Size(width, height),
                Location = loc,
                Parent = parent,
                BorderStyle = BorderStyle.FixedSingle,
                BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(39)))),
((int)(((byte)(234)))), ((int)(((byte)(255))))),
            };

            blockInfo = new ToolTip
            {
                IsBalloon = true
            };
            updateInfo();
        }
        void updateInfo()
        {
            blockInfo.SetToolTip(this.block, "任务编号为: " + this.num.ToString()
+ ", 大小为: " + this.width.ToString());
        }
        public void BringToFront()
        {
            block.BringToFront();
        }
        public void remove()
        {
            this.parent.Controls.Remove(block);
        }
        public int getNum()
        {
            return this.num;
        }
        public int getWidth()
        {
            return this.width;
        }
        public int getPosX()
        {
            return loc.X - 12;
        }
```

```
    }
}
```