# 第一次上机作业

在这里我使用算法实现的思想是二分查找，一种在有序数组中查找特定元素的高效算法。以下是算法的具体实现步骤和思想：

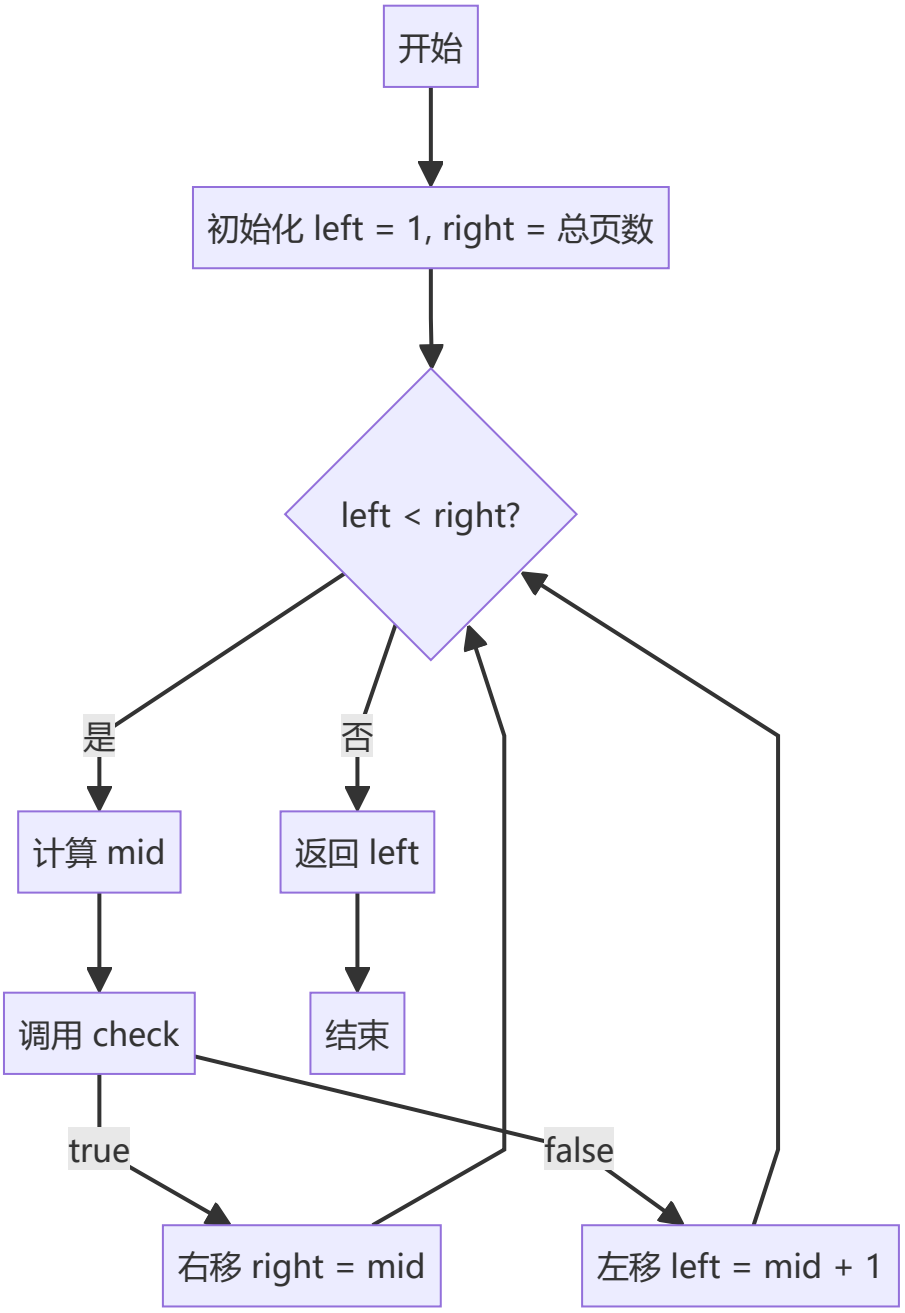
### 算法思想：

1. **确定思路**：我们需要找到一个最小的页数x，使得我们可以将N本书分配给M个学生，每个学生阅读的总页数不超过x。
2. **二分查找**：由于问题要求的是最小页数，我使用二分查找来确定这个最小值。
3. **检查函数**：定义一个check函数，它接受当前的页数x作为参数，并返回一个布尔值，表示是否可以将书分配给学生，使得每个学生阅读的页数不超过x。
4. **初始化搜索范围**：设置两个变量left和right，分别代表搜索范围的下限和上限。left初始化为1，right初始化为所有书的总页数。
5. **循环查找**：在left小于right的条件下循环，每次计算中间值mid，并调用check函数检查是否可以分配。
6. **调整搜索范围**：如果check函数返回true，说明当前的mid值太大，可以减小搜索范围，将right更新为mid。如果返回false，说明mid值太小，需要增加搜索范围，将left更新为mid + 1。
7. **返回结果**：当left和right相等时，循环结束，此时left（或right）就是我们要找的最小页数。

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

// 检查是否可以将书分配给学生，使得每个学生阅读的页数不超过x。
bool check(const vector<int>& pages, int x, int M) {
    int students = 1; // 当前学生数量
    int sum = 0; // 当前学生阅读的页数总和
    for (int i = 0; i < pages.size(); i++) {
        if (sum + pages[i] > x) {
            // 如果当前学生阅读的页数加上当前书的页数超过x，则分配给下一个学生
            students++;
            sum = pages[i];
            if (students > M)
                // 如果学生数量超过M，则无法分配
                return false;
        else
            // 如果当前学生可以阅读当前书，则累加页数
            sum += pages[i];

    }
    return true;
}

int findMinPages(vector<int> pages, int M) {
    int left = 1;
    int right = accumulate(pages.begin(), pages.end(), 0); // 所有书的总页数
    while (left < right) {
        int mid = left + (right - left) / 2;
        if (check(pages, mid, M))
```

```
            // 如果可以分配，则尝试减小x
            right = mid;
        else
            // 如果不可以分配，则增加x
            left = mid + 1;
    }
    return left;
}

int main() {
    int N ;
    vector<int> pages = {};
    int M ;
    cin >> N;
    for (int i = 0; i < N; i++) {
        int temp;
        cin >> temp;
        pages.push_back(temp);
    }
    cin >> M;
    cout << "The minimum number of pages is: " << findMinPages(pages, M) << endl;
    return 0;
}
```

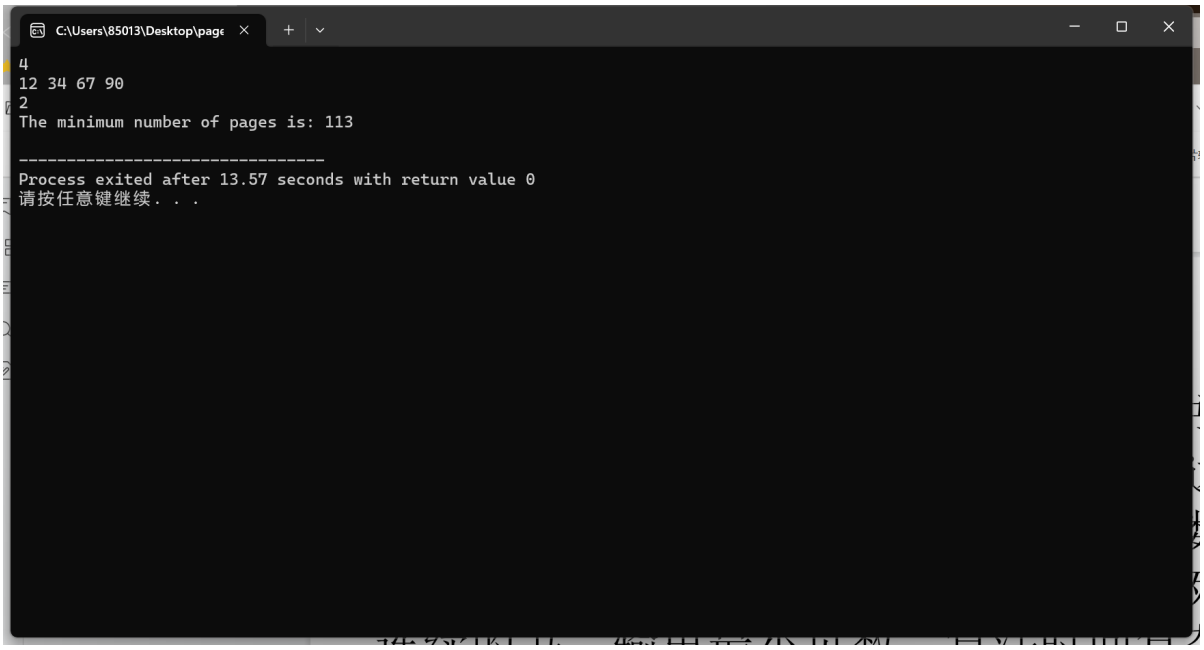
流程图：

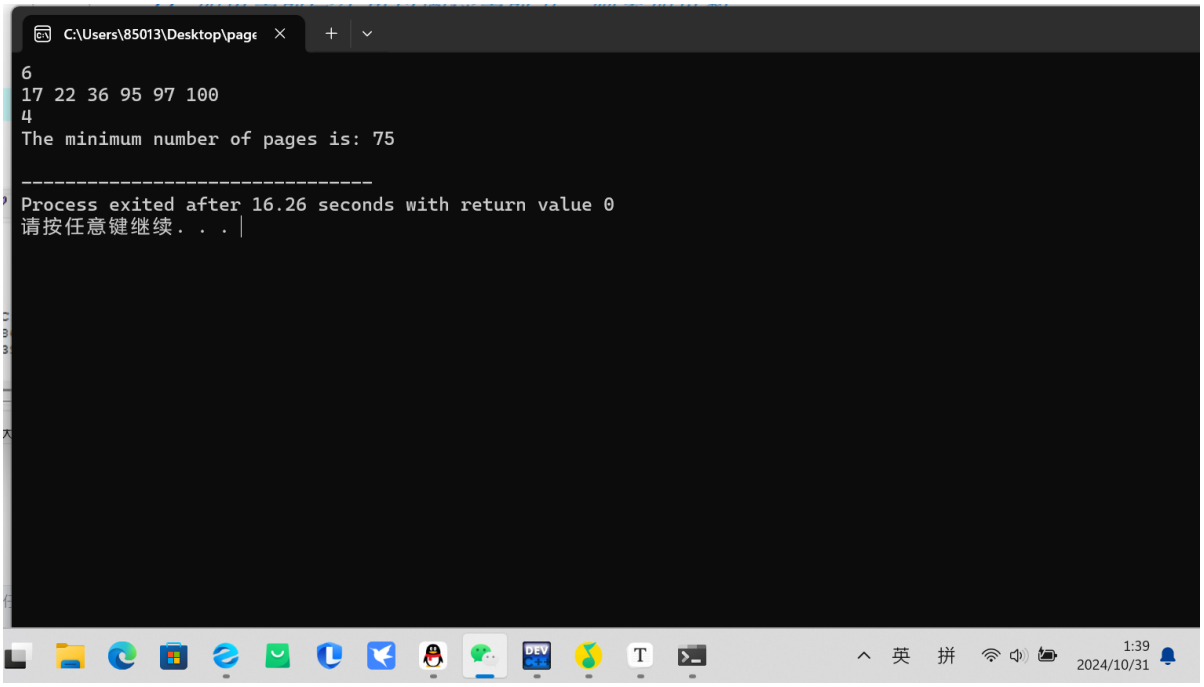


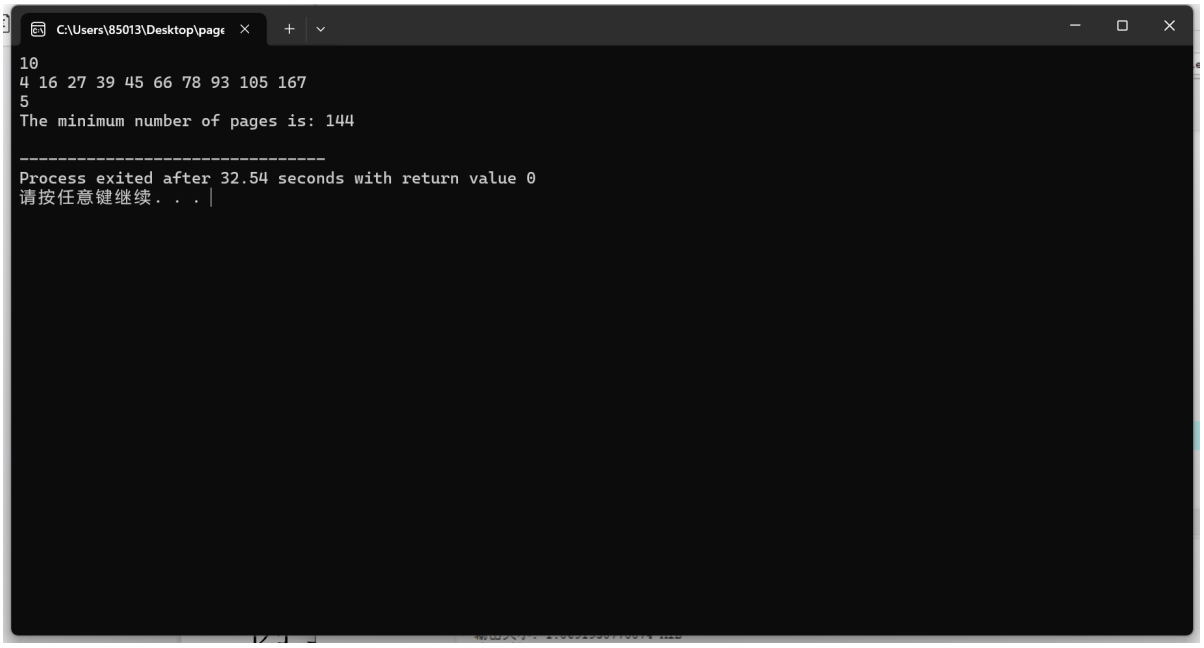

**代码实现：**

代码中的check函数通过模拟分配过程来确定是否可以将书分配给学生，使得每个学生阅读的页数不超过x。如果无法分配，返回false；如果可以分配，返回true。findMinPages函数则使用二分查找来找到最小的x。

## 结果截图：


C:\Users\85013\Desktop\page
4
12 34 67 90
2
The minimum number of pages is: 113
--------------------------------
Process exited after 13.57 seconds with return value 0
请按任意键继续. . .


C:\Users\85013\Desktop\page
6
17 22 36 95 97 100
4
The minimum number of pages is: 75
--------------------------------
Process exited after 16.26 seconds with return value 0
请按任意键继续. . .
英
拼
1:39
2024/10/31


C:\Users\85013\Desktop\page
10
4 16 27 39 45 66 78 93 105 167
5
The minimum number of pages is: 144
--------------------------------
Process exited after 32.54 seconds with return value 0
请按任意键继续. . .

可得出算法正确。

## 时间复杂度：

这种算法的时间复杂度是O(N log S)，其中N是书的数量，S是所有书的总页数。需要O(N)的时间来检查每个mid值，而二分查找需要O(log S)次迭代。