算法作业7-3

2154312 郑博远

- 用一个二维数组来实现动态规划，dp[i][j]表示第i个作业在A机器累计工作j时长时B机器的累计工作时间;
- 对于第i个作业，若上一个作业是A机器运作，则dp[i][j] = dp[i-1][j-wa[i]]；否则dp[i][j] = dp[i][j] + wb[i];
- 最终比较下标n（最后一个作业）在各个下标下A机器时长与B机器时长的最大值的最小值，即为本题答案。

```
#include <iostream>
#include <fstream>
#include <algorithm>
#include <vector>
#define MAXNUM  100
#define MAXTIME 10000
// dp[i][a]表示第i个作业在A机器累计工作a时长时B机器的累计工作时间
int dp[MAXNUM][MAXTIME];
using namespace std;

int solve(vector<int>& a, vector<int>& b)
{
    int suma = 0, ans;

    for (int i = 1; i ≤ a.size(); i++) {
        // suma决定了遍历dp数组第二个下标的上限
        suma += a[i - 1];

        /* 枚举第i个作业的可能情况 */
        // 如果A机器总时长还没有当前作业大，那只能是B机器收下
        for (int j = 0; j < a[i - 1]; j++) {
            dp[i][j] = dp[i - 1][j] + b[i - 1];
        }
        // 否则则有两种可能
        for (int j = a[i - 1]; j ≤ suma; j++) {
            dp[i][j] = min(dp[i - 1][j - a[i - 1]], dp[i - 1][j] + b[i - 1]);
        }
    }

    ans = suma;
    for (int i = 0; i ≤ suma; i++) {
        ans = min(ans, max(dp[a.size()][i], i));
    }

    return ans;
}

int main()
```

```
{
    ifstream infile;
    infile.open("./input.txt", ios::in);
    ofstream outfile;
    outfile.open("./out.txt", ios::out);

    if (!infile.is_open() || !outfile.is_open())
        return 0;

    int n;
    infile >> n;

    vector<int> a, b;
    for (int i = 0; i < n; i++) {
        int e;
        infile >> e;
        a.push_back(e);
    }
    for (int i = 0; i < n; i++) {
        int e;
        infile >> e;
        b.push_back(e);
    }

    outfile << solve(a, b);

    infile.close();
    outfile.close();

    return 0;
}
```