银行家算法　邓依伊　201411212043

- 首先，理一下算法思路：

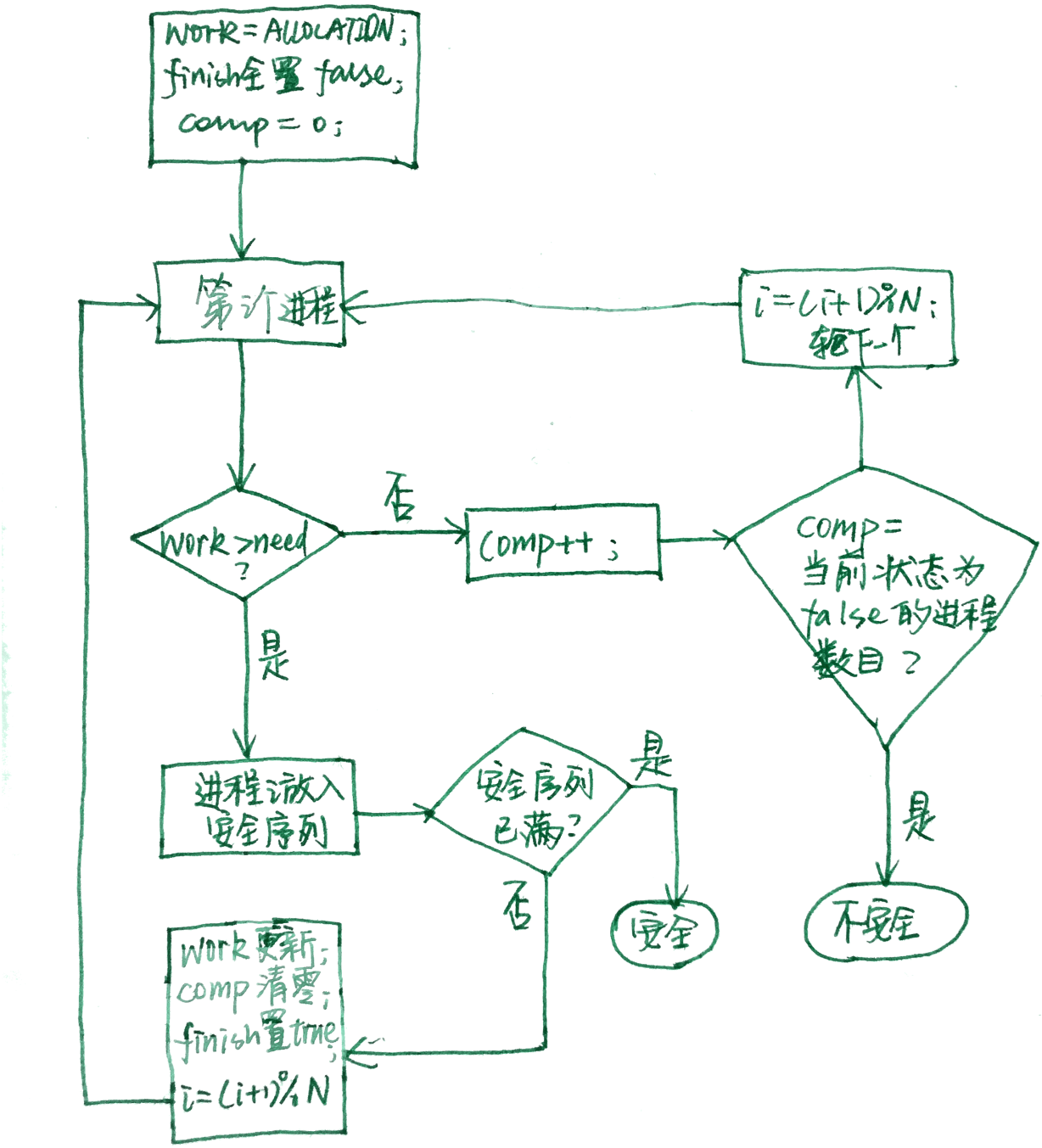
逐轮寻找满足要求的安全进程，并判断序列是否安全，这个问题如何用代码实现，我考虑了很久。一开始只是简单地用for循环把所有进程循环了一遍，但是这样相当于只能将这些进程按顺序遍历一遍，安全序列还没有装满。

所以应该以循环数 `i=(i+1)%m` 的方式来遍历进程，然后判断是否满足退出的条件。

当安全队列满了时就退出循环并判断是安全的，这很容易，但是要判断出一个不安全的队列，还要多定义两个变量，一个记录一共连续判断了几个work不满足need的进程，一个记录当前有几个进程的finish为flase。

由于循环数是轮转的，所以当上述两个变量相等时，循环即可退出并判断不安全。

最后的算法流程如图所示。



- 源程序如下：

```
//
//  main.cpp
//  banker
//
//  Created by Yiyi Deng on 11/2/16.
//  Copyright © 2016 邓依伊. All rights reserved.
//
#include <iostream>
```

```
#include <fstream>
#include <vector>

using namespace std;
#define MAXPROCESS 5                            /*最大进程数*/
#define MAXRESOURCE 3                           /*最大资源数*/

int AVAILABLE[MAXRESOURCE];                     /*可用资源数组*/
int MAX[MAXPROCESS][MAXRESOURCE];               /*最大需求矩阵*/
int ALLOCATION[MAXPROCESS][MAXRESOURCE];        /*分配矩阵*/
int NEED[MAXPROCESS][MAXRESOURCE];              /*需求矩阵*/
int REQUEST[MAXPROCESS][MAXRESOURCE];           /*进程需要资源数*/
bool FINISH[MAXPROCESS];                        /*系统是否有足够的资源分配*/
int p[MAXPROCESS];                              /*记录序列*/
int m,n;                                        /*m个进程,n类资源*/
void Init();
bool Safe();
void Bank();

int main()
{
    Init();
    Safe();
    Bank();
}

void Init()                 /*初始化算法*/
{
    int i,j;
    cout<<"请输入进程的数目:";
    cin>>m;
    cout<<"请输入资源的种类:";
    cin>>n;
    cout<<"请输入每个进程最多所需的各资源数,按照"<<m<<"x"<<n<<"矩阵输入"<<endl;
    for(i=0;i<m;i++)
        for(j=0;j<n;j++)
            cin>>MAX[i][j];
    cout<<"请输入每个进程已分配的各资源数,也按照"<<m<<"x"<<n<<"矩阵输入"<<endl;
    for(i=0;i<m;i++)
    {
        for(j=0;j<n;j++)
        {
            cin>>ALLOCATION[i][j];
            NEED[i][j]=MAX[i][j]-ALLOCATION[i][j];
            if(NEED[i][j]<0)
            {
```

```
                cout<<"输入的第"<<i+1<<"个进程所拥有的第"<<j+1<<"个资源数错误,请重新输入:"<<endl;
                j--;
                continue;
            }
        }
    }
    cout<<"请输入各个资源现有的数目:"<<endl;
    for(i=0;i<n;i++)
    {
        cin>>AVAILABLE[i];
        for(j=0;j<m;j++)
            AVAILABLE[i] -= ALLOCATION[j][i];
    }
}

void Bank()                 /*银行家算法*/
{
    int i,pid;
    char again;
    while(1)
    {
        cout<<"请输入要申请资源的进程号(注:第1个进程号为0,依次类推)"<<endl;
        cin>>pid;
        cout<<"请输入进程所请求的各资源的数量"<<endl;
        for(i=0;i<n;i++)
        {
            cin>>REQUEST[pid][i];
        }
        for(i=0;i<n;i++)
        {
            if(REQUEST[pid][i]>NEED[pid][i])
            {
                cout<<"输入的请求数超过进程的需求量!请重新输入!"<<endl;
                continue;
            }
            if(REQUEST[pid][i]>AVAILABLE[i])
            {
                cout<<"输入的请求数超过系统有的资源数!请重新输入!"<<endl;
                continue;
            }
        }
        for(i=0;i<n;i++)
        {
            AVAILABLE[i]-=REQUEST[pid][i];
            ALLOCATION[pid][i]+=REQUEST[pid][i];
```

```
            NEED[pid][i]-=REQUEST[pid][i];
        }
        if(Safe())
        {
            cout<<"同意分配请求!"<<endl;
        }
        else
        {
            cout<<"您的请求被拒绝!"<<endl;
            for(i=0;i<n;i++)
            {
                AVAILABLE[i]+=REQUEST[pid][i];
                ALLOCATION[pid][i]-=REQUEST[pid][i];
                NEED[pid][i]+=REQUEST[pid][i];
            }
        }
        cout<<"还需再次请求分配吗?是请按y/Y,否请按其它键"<<endl;
        cin>>again;
        if(again=='y'||again=='Y')
            continue;
        break;
    }
}

bool Safe ()
{
    // l：已确认安全的进程数；compcount：其need与当前work比较的连续进程数目，fcount：当前是flase状态的进程数目
    // 若连续比了fount个都是flase，则表示不安全。
    int i, j, l = 0, compcount = 0, fcount = 5;
    int Work[MAXRESOURCE];
    for (i = 0; i < n; i++)
        Work[i] = AVAILABLE[i];
    for (i = 0; i < m; i++)
        FINISH[i] = false;

    // 意为将当前work和每一个flase的进程逐一比较：
    for (i=0; ; i=(i+1)%m)
    {
        if (FINISH[i])
            continue;

        for (j = 0; j < n; j++)
        {
            if (Work[j] < NEED[i][j])
            {
```

```
                compcount++;
                break;
            }
        }
        // 上面这个比较每类资源的循环结束后:
        if (j == n) // 说明每一类资源均满足要求
        {
            FINISH[i] = true;
            fcount--;
            compcount=0;
            for (j = 0; j < n; j++)
                Work[j] += ALLOCATION[i][j];
            p[l++] = i; // 安全算法中的进程序列
        }
        if (l == m) {
            cout << "系统是安全的" << endl;
            cout << "安全序列:" << endl;
            for (i = 0; i < l; i++) {
                cout << p[i];
                if (i != l - 1)
                    cout << ", ";
            }
            cout << "" << endl;
            return true;
        }
        if(compcount==fcount)
            break;
    }
    cout << "系统是不安全的" << endl;
    return false;
}
```

- 程序运行情况:

输入的值为教材上的数据，P0 ~ P4四个进程，输入后初次判断如下:

```
请输入进程的数目:5
请输入资源的种类:3
请输入每个进程最多所需的各资源数,按照5x3矩阵输入
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
请输入每个进程已分配的各资源数,也按照5x3矩阵输入
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
请输入各个资源现有的数目:
10 5 7
系统是安全的
安全序列:
1, 3, 4, 0, 2
```

之后进程1、4分别请求资源，情况如下：

```
请输入要申请资源的进程号(注:第1个进程号为0,依次类推)
1
请输入进程所请求的各资源的数量
1 0 2
系统是安全的
安全序列:
1, 3, 4, 0, 2
同意分配请求!
您还想再次请求分配吗?是请按y/Y,否请按其它键
y
请输入要申请资源的进程号(注:第1个进程号为0,依次类推)
4
请输入进程所请求的各资源的数量
3 3 0
您输入的请求数超过系统有的资源数!请重新输入!
系统是不安全的
您的请求被拒绝!
您还想再次请求分配吗?是请按y/Y,否请按其它键
```

- 小结：

本次总结的经验是想不清楚代码实现的时候，把变量写清楚并画流程图很有帮助。