计算概论机考题目备选

幂法求解矩阵绝对值最大的特征值

矩阵的特征值在理论和实践中均有重要作用, 所以如何矩阵的特征值, 是大家所关心的一个问题. 对于求解矩阵绝对值最大的特征值, 有如下的简单方法:

1. 随机选择初始向量 `v`
2. 不断地做矩阵向量乘法 `v=Av`

可以证明, 在一定条件下, `v` 会逐渐收敛到 `A` 的绝对值最大的特征值所对应的特征向量. 所以, 请用C语言实现如下算法, 计算矩阵绝对值最大的特征值及其对应的特征向量.

1. 选择初始向量 `v`;
2. 做矩阵向量乘法 `v=Av`;
3. 将 `v` 归一化(目的是防止溢出), 即`v`除以`v`的二范数;
4. 做收敛性判断; 如果不收敛, 返回第2步, 否则输出`v`做为近似特征向量.

附加说明:

1. 初始向量 `v` 不妨选为分量全为1的向量;
2. 向量 `v` 的二范数定义为, `v`的各分量的平方和, 再开根号.
3. 收敛的准则: 前后两次 `v` 的差向量的二范数小于1E-8;
4. 输入时, 首先输入一个整数 `N` 代表矩阵的规模, 然后输入 `N*N` 个双精度浮点数, 做为矩阵元素. 输出时为了统一, 请输出二范数为1, 且第一个分量非负的那个特征向量.
5. 一些简化: 输入矩阵为实对称矩阵, 且特征值互异.

样例输入一:

```
2
2 1
1 2
```

样例输出一:

```
Eigenvalue:
3.0000
Eigenvector:
0.7071
0.7071
```

样例输入二:

```
3
2 1 1
1 2 1
1 1 2
```

样例输出二:

```
Eigenvalue:
4.0000
Eigenvector:
0.5774
0.5774
0.5774
```

参考代码

```
#include<stdio.h>
#include<math.h>

double A[100][100];

// multiplication between matrix and vector
void mv(double A[][100], double *x, double* res, int N)
{
        for(int i=0;i<N;i++)
        {
                res[i]=0;
                for(int j=0;j<N;j++)
                {
                        res[i]+=A[i][j]*x[j];
                }
        }
}

double norm2(double *x, int N)
{
        double res=0;
        for(int i=0;i<N;i++)
        {
                res+=x[i]*x[i];
        }
        return sqrt(res);
}

void pow_method(double A[][100], int N)
{
        double tmp;
        double x_new[100],x_old[100],diff[100];
        for(int i=0;i<N;i++) x_new[i]=1;
        do{
                // x_old=x_new;
                for(int i=0;i<N;i++)
                {
                        x_old[i]=x_new[i];
                }
                mv(A,x_old,x_new,N);
                // Normalize x_new
```

```
                tmp=norm2(x_new,N);
                for(int i=0;i<N;i++)
                {
                        x_new[i]=x_new[i]/tmp;
                }
                // diff=x_new-x_old;
                for(int i=0;i<N;i++)
                {
                        diff[i]=x_new[i]-x_old[i];
                }
        }while(norm2(diff,N)>1e-8);
        //Output
        printf("Eigenvalue:\n%.4f\n",tmp);
        printf("Eigenvector:\n");
        double sign=1.0;
        if(x_new[0]<0) sign=-1.0;
        for(int i=0;i<N;i++)
        {
                printf("%.4f\n",sign*x_new[i]);
        }
}

int main()
{
        int N;
        scanf("%d",&N);
        for(int i=0;i<N;i++)
        {
                for(int j=0;j<N;j++)
                {
                        scanf("%lf",&A[i][j]);
                }
        }
        pow_method(A,N);
        return 0;
}
```

最大的数

输入 `N` 个正整数(`1<=N<100`), 输出这 `N` 个整数通过改变次序, 所能组成的最大整数. 例如, 输入 `12` 和 `3` , 结果为 `312` , 因为 `123<312` .

样例输入一:

```
3
87 86 8
```

样例输出一:

```
88786
```

样例输入二:

```
3
89 86 8
```

样例输出二:

```
89886
```

思路提示: 可以通过对输入的整数进行降序排列来实现上述功能, 但是需要合理定义新的"大小"关系. 这种具有自反性和传递性的二元关系, 在数学上被称为拟序关系. 其他不同思路实现功能, 也可.

样例代码

```
#include <stdio.h>

// if a>b in the new order relation, return 1
int newOrder(int a,int b)
{
        int tmp_a=a, tmp_b=b;
        int len_a=0, len_b=0;
        // a,b length
        while(tmp_a>0)
        {
                tmp_a=tmp_a/10;
                len_a++;
        }
        while(tmp_b>0)
        {
                tmp_b=tmp_b/10;
                len_b++;
        }
        int ab=a, ba=b;
        for(int i=0;i<len_a;i++)
                ba=ba*10;
        for(int i=0;i<len_b;i++)
                ab=ab*10;
        ab=ab+b, ba=ba+a;
        if(ab>ba)
                return 1;
        else
                return 0;
}

void largestNumber(int *A, int N)
{
        // Any sort algorithm for the new order
        // In descending order
        // Here is the select sort
        for(int i=0;i<N;i++)
        {
                int max_index=i;
                for(int j=i+1;j<N;j++)
                {
```

```
                        if(newOrder(A[j],A[max_index])==1)
                        {
                                max_index=j;
                        }
                }
                // A[max_index] is maximum in i,i+1,...N-1
                // Exchange A[i] and A[max_index]
                int tmp=A[i];
                A[i]=A[max_index];
                A[max_index]=tmp;
        }
}

int main()
{
        int N; int A[100];
        scanf("%d",&N);
        for(int i=0;i<N;i++)
                scanf("%d",&A[i]);
        largestNumber(A,N);
        for(int i=0;i<N;i++)
        {
                printf("%d",A[i]);
        }
        printf("\n");
        return 0;
}
```