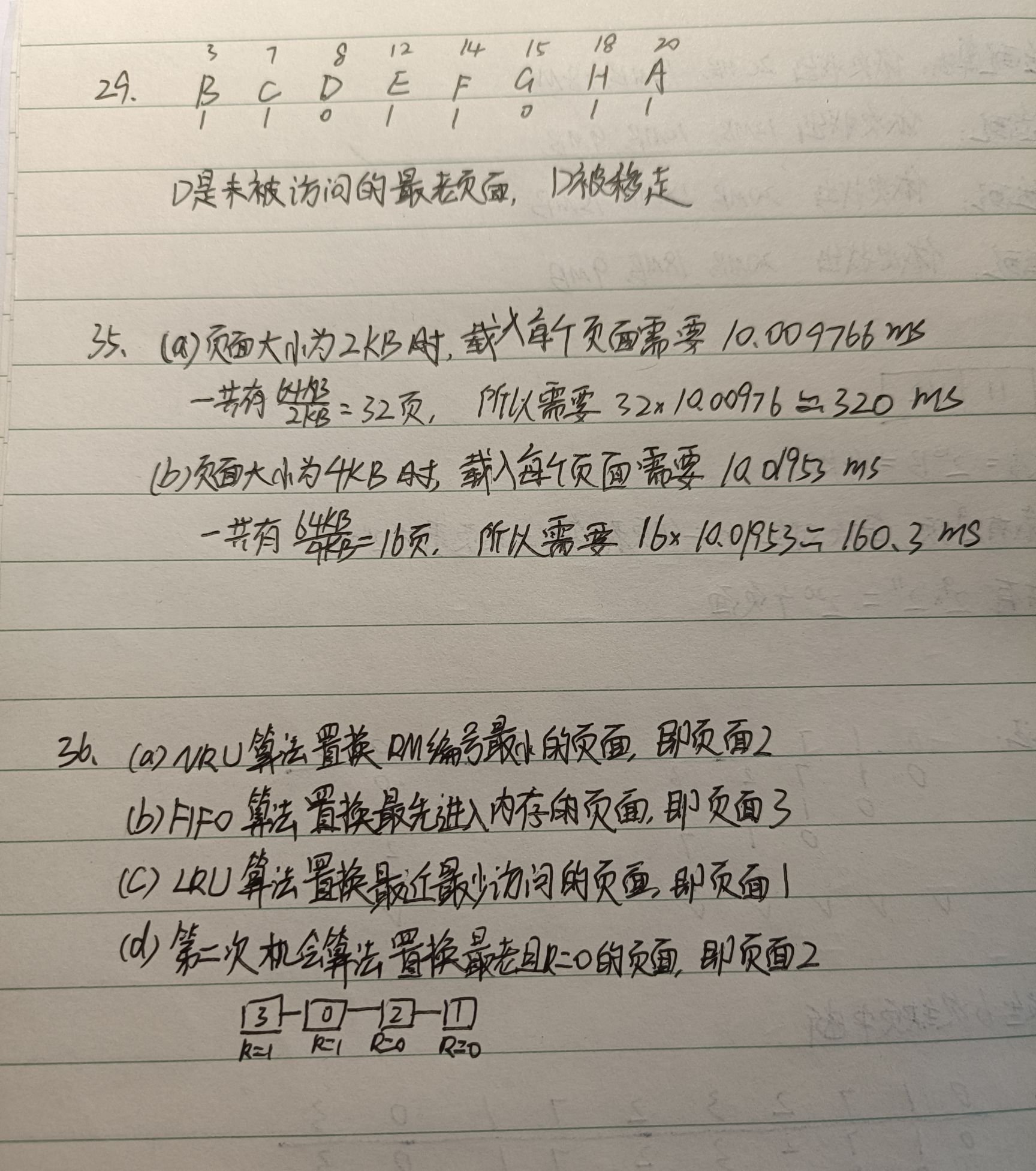
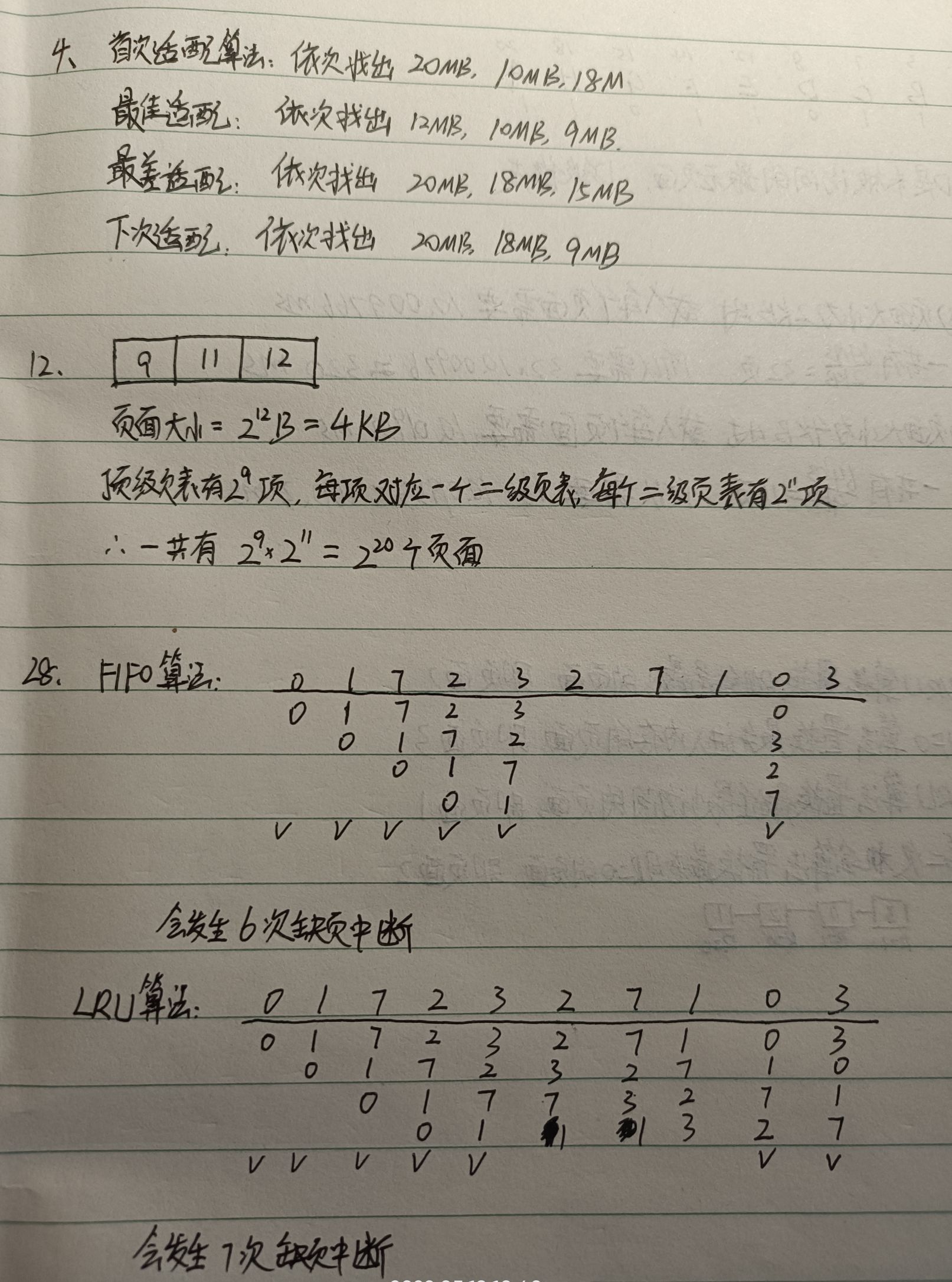
**手写题：**





**代码题：**

源码：

#include<iostream>

#include<vector>

#include<queue>

#include<string>

#include<list>

#include<algorithm>

using namespace std;

#define N 10

class Page{

public:

    Page(int n):n(n) { R=M=0; }

    int R, M;

    int n;

    bool operator==(const Page& p)

    {

        return n == p.n;

    }

};

vector<int> lists;

list<Page> pages;

string s;

int n;

bool Find(const Page& p)

{

    for(auto &ch:pages)

    {

        if(ch.n == p.n)

            return true;

    }

    return false;

}

void OPT()

{

    int m=0;

    for(auto& ch: s)

        lists.push\_back(ch-'0');

    int k=0;

    for(auto& c:lists)

    {

        cout << "当前访问 页面" << c << " ";

        if(Find(Page(c)))

        {

            cout << endl;

            continue;

        }

        if(pages.size() == n)

        {

            m++;

            vector<bool> flag(N, false);

            int a=0;

            int xxxxx=0;

            for(int i=k+1;i<lists.size();++i)

            {

                for(auto& ch:pages)

                {

                    if(ch.n == lists[i])

                    {

                        a++;

                        flag[ch.n] = true;

                        break;

                    }

                }

                if(a == n-1)

                    break;

            }

            for(auto& ch:pages)

            {

                if(flag[ch.n] == false)

                {

                    cout << "页面" << ch.n << "换出内存   ";

                    cout << "页面" << c << "换入内存";

                    pages.remove(Page(ch.n));

                    pages.push\_back(Page(ch.n));

                    break;

                }

            }

        }

        pages.push\_back(Page(c));

        cout << endl;

        k++;

    }

    cout << "一共发生" << m+n << "次缺页, " << m << "次置换" << endl;

}

void NRU()

{

    int m=0;

    for(auto& ch: s)

        lists.push\_back(ch-'0');

    for(auto& c:lists)

    {

        cout << "当前访问 页面" << c << "   ";

        if(Find(Page(c)))

        {

            cout << endl;

            continue;

        }

        pages.push\_back(Page(c));

        if(pages.size() > n)

        {

            int r=1, m=1, i=0, k=0;

            for(auto& ch:pages)

            {

                if(ch.R<r || (ch.R==r && ch.M<m))

                { r=ch.R; m=ch.M; k=i;}

                ++i;

            }

        }

        cout << endl;

    }

    cout << "一共发生" << m+n << "次缺页, " << m << "次置换" << endl;

}

void FIFO()

{

    int m=0;

    for(auto& ch: s)

        lists.push\_back(ch-'0');

    for(auto& c:lists)

    {

        cout << "当前访问 页面" << c << "   ";

        if(Find(Page(c)))

        {

            cout << endl;

            continue;

        }

        pages.push\_back(Page(c));

        if(pages.size() > n)

        {

            cout << "页面" << pages.front().n << "换出内存   ";

            cout << "页面" << c << "换入内存";

            pages.pop\_front();

            m++;

        }

        cout << endl;

    }

    cout << "一共发生" << m+n << "次缺页, " << m << "次置换" << endl;

}

void LRU()

{

    int m=0;

    for(auto& ch: s)

        lists.push\_back(ch-'0');

    for(auto& c:lists)

    {

        cout << "当前访问 页面" << c << "   ";

        if(Find(Page(c)))

        {

            pages.remove(Page(c));

        }

        pages.push\_back(Page(c));

        if(pages.size() > n)

        {

            cout << "页面" << pages.front().n << "换出内存   ";

            cout << "页面" << c << "换入内存";

            pages.pop\_front();

            m++;

        }

        cout << endl;

    }

    cout << "一共发生" << m+n << "次缺页, " << m << "次置换" << endl;

}

int main()

{

    char ch;

    cout << "请输入要访问的页面列表:";

    cin >> s;

    cout << "请输入页框大小：";

    cin >> n;

    cout << "请选择页面置换算法：";

    cout << "1.OPT  2.NRU  3.FIFO  4.LRU" << endl;

    cin >> ch;

    switch (ch)

    {

    case '1':

        OPT();

        break;

    case '2':

        NRU();

        break;

    case '3':

        FIFO();

        break;

    case '4':

        LRU();

        break;

    default:

        cout << "输入错误" << endl;

        break;

    }

}

运行结果：