

Задача 3: Шестнадцатеричный максимум

Северин сегодня в школе познакомился с шестнадцатеричной системой. Теперь он использует короткие палочки для еды, чтобы представить шестнадцать цифр, как на семисегментном дисплее. Таким образом, для представления цифры доступно семь позиций; каждая позиция либо занята палочкой, либо свободна.

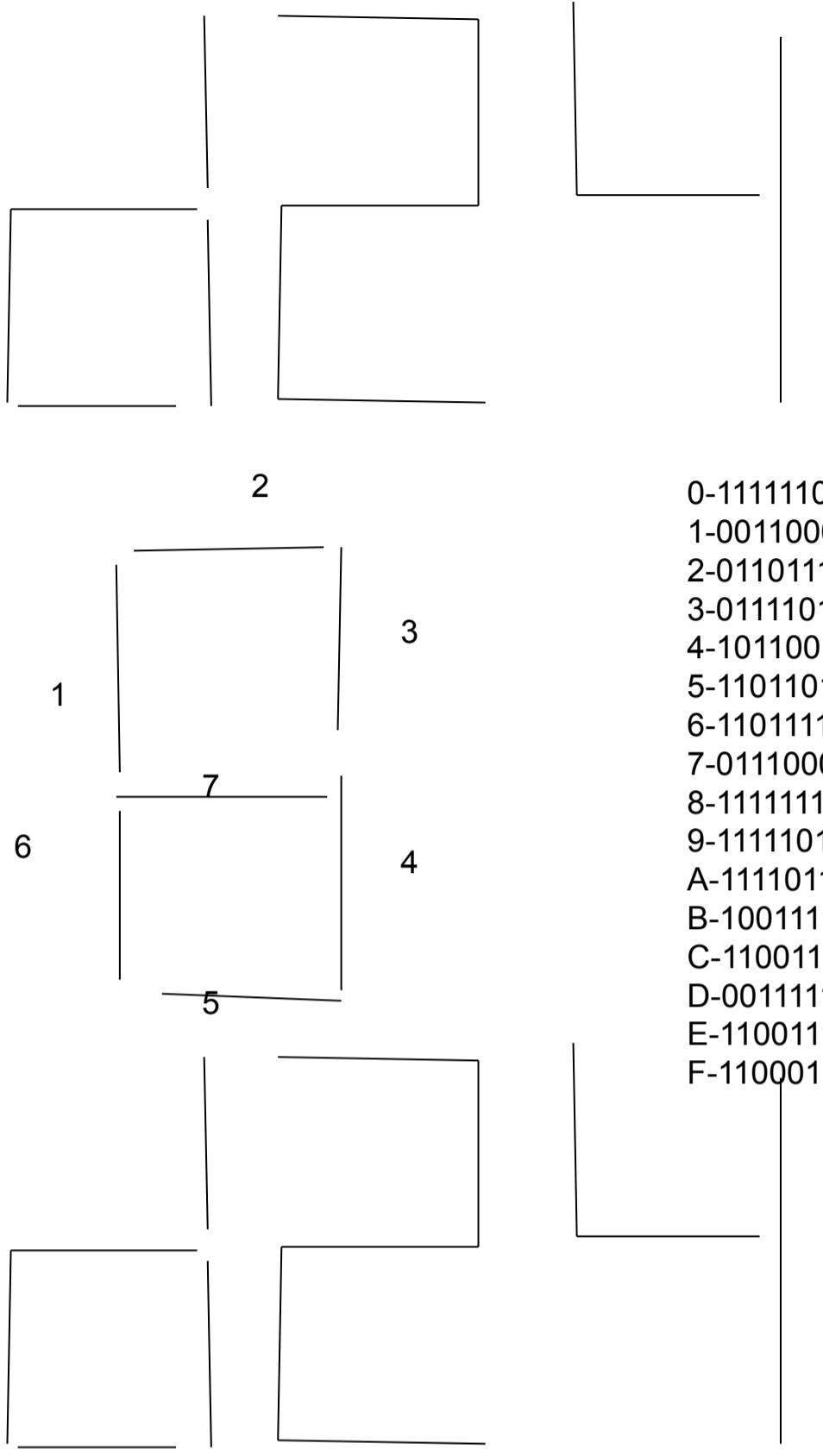
Теперь Северин хочет решить следующую загадку, заданную учителем: дано шестнадцатеричное десятичное число (сокращенно: шестнадцатеричное число) с n цифрами. Искомое-это наибольшее шестнадцатеричное число с тем же количеством цифр, которое может быть получено из данного числа путем дополнительного заданного максимального количества преобразований. „Перемещение " означает перемещение палочки из ее предыдущего положения в другое, доселе свободное положение.

Одно преобразование еще не должно давать допустимое шестнадцатеричное число, но результат после всех преобразований должен быть допустимым шестнадцатеричным числом в семисегментном представлении. Ни в коем случае нельзя полностью „опустошать " представление цифры. Заданное максимальное количество преобразований не должно быть исчерпано.

Вот пример: приведено число D24 и максимальное количество преобразований 3. Шестнадцатеричное число, которое вы ищете,-EE4.

Задание

Напишите программу, которая считывает шестнадцатеричное число, а также максимальное число m преобразований и определяет наибольшее шестнадцатеричное число, которое может быть сгенерировано с максимальным числом m преобразований. Программа должна выводить промежуточное положение, то есть текущую занятость позиций, после каждого перевода.



| | | | | |
|------|------|------|------|----|
| 01 | 1111 | 1100 | 0110 | 00 |
| 57 | 1101 | 1010 | 1110 | 00 |
| 1111 | 1100 | 0110 | 00 | |
| 1101 | 1010 | 1110 | 00 | |
| 1111 | 1100 | 0110 | 00 | |
| 1101 | 1110 | 0110 | 00 | |
| 1101 | 1010 | 1110 | 00 | |

```

void print_ar_hex(int* ar, int n)
{
    for(int i=0;i<n;i++)
    {
        cout<<hex<<ar[i]<<" ";
    }
    cout<<endl;
}

int proverka(int* ar_orig, int* end_ar, int n, int m)
{
    int ar_size=0,end_ar_size=0;
    for(int i=0;i<n;i++)
    {
        if(ar_orig[i]==1)
        {
            ar_size++;
        }
        if(end_ar[i]==1)
        {
            end_ar_size++;
        }
    }
    if(ar_size==end_ar_size)
    {
        int* ar=new int[n];
        int counter=0;
        for(int i=0;i<n;i++)
        {
            ar[i]=ar_orig[i];
        }
        for(int k=0;k<=m;k++)
        {
            for(int i=0;i<n;i++)
            {
                if(ar[i]==end_ar[i] && end_ar[i]==0)
                {
                    ar[i]=0;
                    for(int u=0;u<n;u++)
                    {
                        if(ar[u]==end_ar[u] && end_ar[u]==1)
                        {
                            ar[u]=1;
                            counter++;
                            break;
                        }
                    }
                }
            }
            if(counter==m)
            {
                for(int i=0;i<n;i++)
                {
                    if(ar[i]==end_ar[i])
                    {
                        return 1;
                    }
                }
            }
            return 0;
        }
        return 0;
    }
    return 0;
}

void sticks()
{
    int bank[16][7]={
        {1,1,1,1,0,0},
        {0,0,1,1,0,0,0},
        {0,1,0,1,1,1},
        {0,1,1,1,0,0,1},
        {1,0,1,1,0,0,1},
        {1,1,0,1,1,0,1},
        {1,1,0,1,1,1,1},
        {0,1,1,1,0,0,0},
        {1,1,1,1,1,1,1},
        {1,1,1,1,1,0,1},
        {1,1,1,1,0,1,1},
        {1,0,0,1,1,1,1},
        {1,1,0,0,1,1,0},
        {0,0,1,1,1,1,1},
        {1,1,0,0,1,1,1},
        {1,1,0,0,0,1,1}
    };
    int bank_size[16]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
    for(int k=0;k<16;k++)
    {
        for(int j=0;j<7;j++)
        {
            if(bank[k][j]==1)
            {
                bank_size[k]++;
            }
        }
    }
}

int num_size_of_num=0;
cin>>hex>>num;
//cin>>hex;
//cin>>k;
//num=0x24;
//M=3;
//cout<<hex<<num<<endl;
//cout<<M<<endl;
int temp_num=num;
while(num>0)
{
    num/=16;
    size_of_num++;
}
int* current_num=new int[size_of_num];
int* end_num=new int[size_of_num];
int* current_num_bin=new int[size_of_num*7];
int* end_num_bin=new int[size_of_num*7];
int* size_of_num_1;
while(temp_num>0)
{
    current_num[temp_num%16];
    end_num[temp_num%16];
    for(int u=0;u<7;u++)
    {
        current_num_bin[7*u+bank[current_num][u]];
        end_num_bin[7*u+bank[end_num][u]];
    }
    temp_num/=16;
}

//print_ar(current_num,size_of_num);
//print_ar(end_num,size_of_num);
//print_ar(current_num_bin,size_of_num*7);
//print_ar(end_num_bin,size_of_num*7);

print_ar_hex(end_num,size_of_num);
cout<<"---"<<endl;
cout<<"---"<<endl;
int k=size_of_num-1, flag_result;
flag_result=proverka(current_num_bin,end_num_bin,(size_of_num*7),M);
//print_ar(end_num,size_of_num);
//print_ar(end_num_bin,size_of_num*7);
{
    else
    {
        k--;
    }
    if(flag_result==1)
    {
        print_ar_hex(end_num,size_of_num);
        cout<<"finsh"<<endl;
        break;
    }
}
else
{
    print_ar_hex(end_num,size_of_num);
    cout<<"qasy finsh"<<endl;
}
}
}

```