МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М.В. ЛОМОНОСОВА

ОЛИМПИАДНАЯ РАБОТА

Наименование олимпиады школьников: **«Ломоносов»**

Профиль олимпиады: **Информатика**

ФИО участника олимпиады: **Ефремов Алексей Александрович**

Класс: **11 класс**

Технический балл: **70**

Дата проведения: **17 марта 2022 г.**

Результаты проверки:

Оценка участника строится из 3 частей:

1. оценка за задание - рассчитывается путем запуска тестов и определения правильности работы программы на тестах, до 100 баллов по каждой задаче;
2. дополнительные баллы за полностью правильное решение задания со 2 по 5 - в случае прохождения всех тестов по заданию к оценке прибавляется 55 баллов;
3. нормализация оценки - если полученная из пунктов 1 и 2 сумма баллов превышает 500, то итоговая оценка - 100, если не превышает 500, но превышает 400 - 99 баллов, если не превышает 400 - делится на 3.9 и округляется до целого.

Оценки за задания:

№	1	2	3	4	5
Оценка	43	100	0	76	0

Дополнительный балл: 55

Задание 1. Попытка 1.

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <bits/stdc++.h>


using namespace std;

typedef long long ll;
#define INFLL 0x2aaaaaaaaaaaaaaa

/*
+7 (925) 970-73-55

pk@cs.msu.ru
*/


#define N 61

int convert_c(char a){
    int res = -1;
    if(('0'<=a)&&(a<='9')){
        res= a-'0';
    }
```

```
    else if(('a'<=a)&&(a<='z')){
        res = a-'a'+10;
    }
    else if(('A'<=a)&&(a<='Z')){
        res = a-'A'+36;
    }

    return res;
}



void remove_zeroes(char **s){
    char *b = *s;
    //printf("char = %hhd\n", *b);
    while((*b)=='0'){

        b++;
    }

    if((*b)==0){
        b--;
        (*b)='0';
    }
```

```
    s[0]=b;
}



int comp(char *a, char *b){
    char *f = a;
    char *s = b;

    int res = 0;

    if(strlen(f)==strlen(s)){
        while((*f)!=0){
            if(convert_c(*f)==convert_c(*s)){
                f++;
                s++;
            }
            else{
                if(convert_c(*f)>convert_c(*s))
                    res=1;
                else
                    res=-1;

                break;
            }
```

```
        }
    }
    else{
        if(strlen(f)>strlen(s))
            res=1;
        else
            res=-1;
    }


    return res;

}




int main(){
    int k;
    int n;
    scanf("%d%d", &k, &n);
```

```
char* arr_s[n];
vector<int> vres;

int res =-1;
for (int i=0; i<n; i++){
    arr_s[i]=new char[N+0x10];
    scanf("%s", arr_s[i]);
    remove_zeroes(&arr_s[i]);
    //printf("new str: %s\n", arr_s[i]);
    if((strlen(arr_s[i])>=k)||(arr_s[i][0]=='0')){
        if(res==-1){
            res= i;
            vres.push_back(i+1);
        }
        else{
            int cc= comp(arr_s[i], arr_s[res]);
            if(cc>=0){
                if(cc==1)
                    vres.clear();
                res=i;
                vres.push_back(i+1);
            }

        }
    }
```

```
    }



    if(res!=-1){
        printf("%s\n", arr_s[res]);


        for(auto v: vres)
            printf("%d\n", v);
    }
    else{
        printf("-1\n");
    }




}
```

Задание 2. Попытка 1.

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <bits/stdc++.h>


using namespace std;

typedef long long ll;
#define INFLL 0x2aaaaaaaaaaaaaa

/*
+7 (925) 970-73-55

pk@cs.msu.ru
*/


#define N 62

int convert_c(char a){
    int res = -1;
    if(('0'<=a)&&(a<='9')){
        res= a-'0';
    }
```

```
    else if(('a'<=a)&&(a<='z')){
        res = a-'a'+10;
    }
    else if(('A'<=a)&&(a<='Z')){
        res = a-'A'+36;
    }

    return res;
}

char deconvert_c(int a){
    char res = '!';
    if((0<=a)&&(a<=9)){
        res= a+'0';
    }
    else if((10<=a)&&(a<=35)){
        res = a-10+'a';
    }
    else if((36<=a)&&(a<=61)){
        res = a-36+'A';
    }

    return res;
}
```

```
void remove_zeroes(char **s){
    char *b = *s;
    //printf("char = %hhd\n", *b);
    while((*b)=='0'){

        b++;
    }

    if((*b)==0){
        b--;
        (*b)='0';
    }


    s[0]=b;
}









int main(){
```

```
int n;
scanf("%d", &n);

char s[n];

char res[N];
res[N-1]=0;

int cur = N-1;

int fin = N-1;

scanf("%s", s);


int hh[N];

for (int i=0; i<N; i++)
    hh[i]=0;


for(int i=0; i<n; i++){
    int k = convert_c(s[i]);

    if(k!=-1){
```

```
        hh[k]++;
    }

}



int pref[N];
pref[0]=hh[0];
for(int i=1; i<N; i++){
    pref[i]=pref[i-1]+hh[i];
}




while(cur!=0){
    int buf = pref[cur-1];
    int c = cur;
    while((hh[c]<=0)&&(c>=0)){
        --c;
    }

    if(c!=cur){
        buf--;
    }
```

```
    if(buf>=cur-1){
        //printf("try %d\n", cur);
        hh[c]--;
        for(int gg = c; gg<=cur; gg++){
            --pref[gg];
        }



        res[N-1-cur]=deconvert_c(c);
    }
    else{
        fin = cur-1;
    }



    --cur;
}
fin = N-1-fin;
if(fin!=N-1){
    char *buf = res+fin;
    remove_zeroes(&buf);
    printf("%s\n", buf);
}
else{
    printf("-1\n");
```

```
    }



}
```

Задание 4. Попытка 1.

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <bits/stdc++.h>


using namespace std;

typedef long long ll;
typedef pair<int, int> pi;


#define INFLL 0x2aaaaaaaaaaaaaaa
#define INF 1000000009

/*
+7 (925) 970-73-55

pk@cs.msu.ru
*/

 int n, m;


int fin = INF;
```

```
vector<pi> vres;

struct Edg{

    int from;
    int to;
    int h;
    friend bool operator< (const Edg &a1, const Edg &a2){
        return a1.h<a2.h;
    };

    Edg(int fr, int t, int hh){
        from =fr;
        to=t;
        h=hh;
    };
};


struct Vert{
    int deep;
    vector<int> tos;
    multiset<Edg> endpoints;

    Vert(){
```

```
        deep=0;
        tos = vector<int>();
        endpoints=multiset<Edg>();
    }
};




int dfs1( map<int, int>* graph, Vert* graph2, bool* used, int cur, int p){

    int res=1;
    used[cur]=1;
    if(p==-1){
        graph2[cur].deep=0;
    }
    else{
        graph2[cur].deep= graph2[p].deep+1;
        graph2[cur].endpoints.insert(Edg(p, cur, graph2[p].deep));
        graph[cur][p]--;

    }


    for(auto &v: graph[cur]){
```

```
    if(v.second!=0){
        if(used[v.first]==0){
            graph2[cur].tos.push_back(v.first);
            v.second--;
            res+=dfs1(graph, graph2, used, v.first, cur);

            for(auto v: graph2[v.first].endpoints)
                graph2[cur].endpoints.insert(v);
        }
        else{
            while(v.second!=0){
                v.second--;
                graph[v.first][cur]--;
                graph2[cur].endpoints.insert(Edg(cur, v.first, graph2[v.first].deep));
            }
        }
    }




}

if((graph2[cur].endpoints.size()<fin)&&(graph2[cur].endpoints.size()>0)){
```

```
        fin = graph2[cur].endpoints.size();
        vres.clear();
        for(auto v: graph2[cur].endpoints){
            int a = v.from;
            int b = v.to;
            if(a>b)
                swap(a, b);

            vres.push_back({a, b});
        }
    }


    graph2[cur].endpoints.erase(Edg(0, 0, graph2[cur].deep-1));


    return res;

};



int main(){

    scanf("%d%d", &n, &m);
```

```
bool used[n];
for (int i=0; i<n; i++)
    used[i]=0;

map<int, int> graph[n];  // map<to, used> graph[from];
Vert graph2[n];
for(int i=0; i<m; i++){
    int from, to;
    scanf("%d%d", &from, &to);

    from--; to--;
    graph[from][to]++;
    graph[to][from]++;

}


int cc = dfs1(graph, graph2, used, 0, -1);

//printf("find : %d\n", cc);
if(cc!=n){
    printf("0\n");
}
```

```
    else{
        printf("%d\n", fin);
        for(auto v: vres){
            printf("%d %d\n", v.first+1, v.second+1);
        }
    }
}
```

Задание 4. Попытка 2.

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <bits/stdc++.h>


using namespace std;

typedef long long ll;
typedef pair<int, int> pi;


#define INFLL 0x2aaaaaaaaaaaaaaa
#define INF 1000000009

/*
+7 (925) 970-73-55

pk@cs.msu.ru
*/

 int n, m;


int fin = INF;
```

```
set<pi> vres;

struct Edg{

    int from;
    int to;
    int h;
    friend bool operator< (const Edg &a1, const Edg &a2){
        return a1.h<a2.h;
    };

    Edg(int fr, int t, int hh){
        from =fr;
        to=t;
        h=hh;
    };
};


struct Vert{
    int deep;
    vector<int> tos;
    multiset<Edg> endpoints;

    Vert(){
```

```
        deep=0;
        tos = vector<int>();
        endpoints=multiset<Edg>();
    }
};




int dfs1( map<int, int>* graph, Vert* graph2, bool* used, int cur, int p){

    int res=1;
    used[cur]=1;
    if(p==-1){
        graph2[cur].deep=0;
    }
    else{
        graph2[cur].deep= graph2[p].deep+1;
        graph2[cur].endpoints.insert(Edg(p, cur, graph2[p].deep));
        graph[cur][p]--;

    }


    for(auto &v: graph[cur]){
```

```
    if(v.second!=0){
        if(used[v.first]==0){
            graph2[cur].tos.push_back(v.first);
            v.second--;
            res+=dfs1(graph, graph2, used, v.first, cur);

            for(auto v: graph2[v.first].endpoints)
                graph2[cur].endpoints.insert(v);
        }
        else{
            while(v.second!=0){
                v.second--;
                graph[v.first][cur]--;
                graph2[cur].endpoints.insert(Edg(cur, v.first, graph2[v.first].deep));
            }
        }
    }



}

if((graph2[cur].endpoints.size()<fin)&&(graph2[cur].endpoints.size()>0)){
```

```
        fin = graph2[cur].endpoints.size();
        vres.clear();
        for(auto v: graph2[cur].endpoints){
            int a = v.from;
            int b = v.to;
            if(a>b)
                swap(a, b);

            vres.insert({a, b});
        }
    }

    graph2[cur].endpoints.erase(Edg(0, 0, graph2[cur].deep-1));

    return res;

};

int main(){

    scanf("%d%d", &n, &m);
```

```
bool used[n];
for (int i=0; i<n; i++)
    used[i]=0;

map<int, int> graph[n];  // map<to, used> graph[from];
Vert graph2[n];
for(int i=0; i<m; i++){
    int from, to;
    scanf("%d%d", &from, &to);

    from--; to--;
    graph[from][to]++;
    graph[to][from]++;

}



int cc = dfs1(graph, graph2, used, 0, -1);

//printf("find : %d\n", cc);
if(cc!=n){
    printf("0\n");
}
```

```
    else{
        printf("%d\n", fin);
        for(auto v: vres){
            printf("%d %d\n", v.first+1, v.second+1);
        }
    }
}
```

Задание 4. Попытка 3.

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <bits/stdc++.h>


using namespace std;

typedef long long ll;
typedef pair<int, int> pi;


#define INFLL 0x2aaaaaaaaaaaaaaa
#define INF 1000000009

/*
+7 (925) 970-73-55

pk@cs.msu.ru
*/

 int n, m;


int fin = INF;
```

```
set<pi> vres;

struct Edg{

    int from;
    int to;
    int h;
    friend bool operator< (const Edg &a1, const Edg &a2){
        return a1.h<a2.h;
    };

    Edg(int fr, int t, int hh){
        from =fr;
        to=t;
        h=hh;
    };
};


struct Vert{
    int deep;
    //vector<int> tos;
    multiset<Edg> endpoints;

    Vert(){
```

```
        deep=0;
        //tos = vector<int>();
        endpoints=multiset<Edg>();
    }
};




int dfs1( map<int, int>* graph, Vert* graph2, bool* used, int cur, int p){

    int res=1;
    used[cur]=1;
    if(p==-1){
        graph2[cur].deep=0;
    }
    else{
        graph2[cur].deep= graph2[p].deep+1;
        graph2[cur].endpoints.insert(Edg(p, cur, graph2[p].deep));
        graph[cur][p]--;

    }


    for(auto &v: graph[cur]){
```

```
    if(v.second!=0){
        if(used[v.first]==0){
            //graph2[cur].tos.push_back(v.first);
            v.second--;
            res+=dfs1(graph, graph2, used, v.first, cur);

            for(auto v: graph2[v.first].endpoints)
                graph2[cur].endpoints.insert(v);
        }
        else{
            while(v.second!=0){
                --v.second;
                --graph[v.first][cur];
                graph2[cur].endpoints.insert(Edg(cur, v.first, graph2[v.first].deep));
            }
        }
    }



}

if((graph2[cur].endpoints.size()<fin)&&(graph2[cur].endpoints.size()>0)){
```

```
        fin = graph2[cur].endpoints.size();
        vres.clear();
        for(auto v: graph2[cur].endpoints){
            int a = v.from;
            int b = v.to;
            if(a>b)
                swap(a, b);

            vres.insert({a, b});
        }
    }

    graph2[cur].endpoints.erase(Edg(0, 0, graph2[cur].deep-1));

    return res;

};

int main(){

    scanf("%d%d", &n, &m);
```

```
bool used[n];


map<int, int> mgraph[n];  // map<to, used> graph[from];
map<int, int> graph[n];  // map<to, used> graph[from];
Vert graph2[n];
for(int i=0; i<m; i++){
   int from, to;
   scanf("%d%d", &from, &to);


   from--; to--;
   mgraph[from][to]++;
   mgraph[to][from]++;


}
int cc =-1;
for(int i=0; i<n; ++i){
   for (int b=0; b<n; ++b){
      used[b]=0;
      graph[b]= mgraph[b];
      graph2[b].endpoints.clear();
   }
```

```
        cc = dfs1(graph, graph2, used, i, -1);
    }

    //printf("find : %d\n", cc);
    if(cc!=n){
        printf("0\n");
    }
    else{
        printf("%d\n", fin);
        for(auto v: vres){
            printf("%d %d\n", v.first+1, v.second+1);
        }
    }
}
```