补题笔记 · ACWing 周赛 #11

唐冬 LaTeX

更新：August 10, 2021

1 A. 计算 abc

1.1 题意

原题链接

1.2 思路

在比赛的时候，由于我自己眼瞎，没有看到题目中说全部数据为正整数，于是写了一个整数范围的解 QAQ，正确的解法是先从小到大排序，最大的数一定是三个数的和，再分别做减法，就能求出 a,b,c 三个数的值了。
我的解法是，先将给出的四个数求和

$$\begin{aligned} Sum_2 &= a + b + c + Sum_1 \\ &= 3a + 3b + 3c \end{aligned} \tag{1}$$

再除以三，得到三个数之和，在进行匹配，找出题目给出的四个数中不是和的三个，再排序后分别做差，就能求出答案了。
下次一定认真读题-v-

1.3 代码

```
#include"bits/stdc++.h"

using namespace std;

int main(){
    freopen("input.txt","r",stdin);
    string s;
    cin>>s;
    int ans=-1;
    for(int i=1;i<=s.size();i++){
        string s2;
        for(int j=0;j<i;j++){
            int p=j;
            while(p<s.size()){s2+=s[p];p+=i;}
            int x=1,y=0,z=1;
            p=1;
            while(p<s2.size()){
                if(s2[p]==s2[p-1])z++;
                else {
                    y=max(y,z);
                    if(x<y)swap(x,y);
                    z=1;
                }
                p++;
            }
            y=max(y,z);
            if(x<y)swap(x,y);
            z=0;
            ans=max(ans,x*y);
        }
```

```
    }
    cout<<ans<<endl;
}
```

2 B. 凑平方

2.1 题意

原题链接

2.2 思路

题目中说每次操作可删除一位数字，所以主体思路是递归遍历模拟删除数字的每一位，类似于遍历连续子串，再判断是否为平方数，记录递归深度（删除的数的个数），最小的记录就是答案。
那么如何判断平方数呢？打表数组开不下，可以先开平方，再平方，判断是否和原数相等（利用 C++ 截断特性）；
如果遍历全部情况，会导致超时。需要三个剪枝：
· 找到平方数后直接返回，因为再次递归的答案一定不是最优解。
· 当前深度不小于记录值时直接返回。
· 当前数遍历完成后，当再次遇到时直接返回，因为我们已经遍历过它和它的全部子集了。
需要额外注意，题目中说需要删除无用的前导零（我因为没有在意这个，喜-2，血亏 QAQ）

2.3 代码

```cpp
#include"bits/stdc++.h"

using namespace std;

#define ll long long

map<int,bool>Hc;

const int nmax=1e9+1;
bool Hash(ll x){
    ll k=(ll)sqrt(x);
    if(x!=0&&k*k==x)return 1;
    return 0;
}

int ans;

void dfs(ll a,int u){
    if(a==0||Hc.count(a)>0)return;
    Hc[a]=1;
    u++;
    if(u>=ans)return;
    int n=0;
    for(ll i=1;i<=1e9;i*=10){
        if(a>(i-1))n++;
        else break;
    }
    if(n==1){if(Hash(a))ans=u;return;}

    bool ok=0;
    for(int i=0;i<n;i++){
        ll k=a%(ll)pow(10,i);
```

```
33          k+=(a/(ll)pow(10,i+1))*(ll)pow(10,i);
34          if(Hash(k)){ok=1;
35          int n2=0;
36          for(ll j=1;j<=1e9;j*=10){
37          if(k>(j-1))n2++;
38          else break;
39      }
40          ans=min(ans,u+n-n2-1);
41          return;}
42          else dfs(k,u);
43      }
44  }
45
46  int main(){
47      //freopen("input.txt","r",stdin);
48      int T;
49      cin>>T;
50      while(T--){
51          Hc.clear();
52          ans=100;
53          ll a;
54          cin>>a;
55
56          if(Hash(a))ans=0;
57          else dfs(a,0);
58
59          bool ok=0;
60          if(ans<100)cout<<ans<<endl;
61          else cout<<"-1\n";
62      }
63  }
```

3 C. 最大化最短路

3.1 题意

原题链接

3.2 思路

题中说固定起点终点，要求选点连接并求出最短路的最大值，那么最短路可能有三种情况：
情况 1: 从 1 点出发到 n 点并经过新路；
情况 2: 从 n 点出发到 1 点并经过新路；
情况 3: 不经过新路的最短路（1，n 出发都一样）。
考虑情况 1 和情况 2，用 BFS 分别求出从 1，n 出发到连接点的最短路，记做 $d_{0,p}, d_{1,p}$**，总路长即为：**

$$S = dist_{0,x} + dist_{1,y} + 1 \tag{2}$$

再通过遍历 x，y 求出最大值。在求最大值的过程中，遍历到 x，y 时会遇到两种情况，x->y 和 y->x；为减少遍历次数，可以先进行排序：

$$d_{0,x} + d_{1,y} < d_{1,x} + d_{0,y} \tag{3}$$

移项，得：

$$d_{0,x} - d_{1,x} < d_{0,y} - d1, y \tag{4}$$

排序后可顺序遍历一次在 O(n) 内求出最大值。

3.3 代码

```cpp
#include"bits/stdc++.h"

using namespace std;

const int nmax=2e6+10;

int n,m,k;
int Node[nmax],            //连接点
    dist[2][nmax];        //路径记录

vector<int>Link[nmax];    //邻接表

void bfs(int fst){
    queue<int>q;
        q.push(fst);
        int kk=fst==1?0:1;
        dist[kk][fst]=0;
        while(!q.empty()){
            int p=q.front();
            int v=dist[kk][p];
            q.pop();
            for(int i=0;i<Link[p].size();i++)if(dist[kk][Link
                [p][i]]<0){
                dist[kk][Link[p][i]]=v+1;
                //printf("dist[%d][%d]=%d\n",kk,Link[p][i],v
                    +1);
                q.push(Link[p][i]);
            }
        }
}

int main(){
```

```
31      //freopen("input.txt","r",stdin);
32      cin>>n>>m>>k;
33      for(int i=0;i<k;i++){
34          int a;
35          scanf("%d",&a);
36          Node[i]=a;
37      }
38      for(int i=0;i<m;i++){
39          int a,b;
40          scanf("%d%d",&a,&b);
41          Link[a].push_back(b);
42          Link[b].push_back(a);
43      }
44      memset(dist,-1,sizeof(dist));
45      bfs(1);
46      bfs(n);
47
48      sort(Node,Node+k,[&](int a,int b){//c++17新特性
49          return dist[0][a]-dist[1][a]<dist[0][b]-dist[1][b];
50      });
51
52      int res=0,fst=dist[0][Node[0]];
53      for(int i=1;i<k;i++){
54          int v=dist[1][Node[i]];
55          res=max(res,v+fst+1);
56          fst=max(fst,dist[0][Node[i]]);
57      }
58
59      cout<<min(dist[0][n],res)<<endl;
60  }
```

在 BFS 求最短路时我更习惯用迭代器，大雪菜老师用的是数组，可以学习一

下大佬的代码 (*f￣f*)。链接

```
#include <iostream>
lude <cstring>
lude <algorithm>

g namespace std;

t int N = 200010, M = 400010;

n, m, k;
h[N], e[M], ne[M], idx;
dist1[N], dist2[N];
q[N], p[N];

 add(int a, int b)  // 添加一条边a->b

e[idx] = b, ne[idx] = h[a], h[a] = idx ++ ;


 bfs(int start, int dist[])

memset(dist, 0x3f, N * 4);
dist[start] = 0;
int hh = 0, tt = 0;
q[0] = start;

while (hh <= tt)
{
    int t = q[hh ++ ];

    for (int i = h[t]; ~i; i = ne[i])
    {
```

```
        int j = e[i];
        if (dist[j] > dist[t] + 1)
        {
            dist[j] = dist[t] + 1;
            q[ ++ tt] = j;
        }
    }
}



main()

scanf("%d%d%d", &n, &m, &k);
memset(h, -1, sizeof h);
for (int i = 0; i < k; i ++ ) scanf("%d", &p[i]);
while (m -- )
{
    int a, b;
    scanf("%d%d", &a, &b);
    add(a, b), add(b, a);
}

bfs(1, dist1);
bfs(n, dist2);

sort(p, p + k, [&](int a, int b) {
    return dist1[a] - dist2[a] < dist1[b] - dist2[b];
});

int res = 0, x = dist1[p[0]];
for (int i = 1; i < k; i ++ )
```

```
{
    int t = p[i];
    res = max(res, dist2[t] + x + 1);
    x = max(x, dist1[t]);
}

printf("%d\n", min(res, dist1[n]));
return 0;
```