构造题杂

CF468C.Hack it!

令f(x)表示x在十进制下各位数字之和,给定一整数a 构造l,r,使得$\sum_{i=l}^{r} f(i) \equiv 0 \mod a$

数据范围$1 \leq a \leq 10^{18}, 1 \leq l, r \leq 10^{200}$

首先问题转化为找到两个前缀使得答案 % a 相等，根据抽屉原理问题一定有解。

容易注意到 $f(x+y) = f(x) + f(y)$ 当且仅当 $x + y$ 没有进位。

容易想到把其中一个后面若干位设置成 0，这样可以任意放。

那么考虑求出 $\sum_{i=0}^{10^k-1} f(i) \equiv p \pmod a$
对于 $x < 9 \times 10^k, f(x + 10^k) = f(x) + 1$，于是不难发现，当 $a - p \leq 9ek$的时候，
$\sum_{i=a-p}^{10^k+a-p-1} f(i) \equiv 0 \pmod a$，该做法在 $a \leq 9 \times 10^k$的时候必然出解。

当然还可以枚举上标为 $t \times 10^k$，则 $a - p \leq (10 - t) \times 10^k$ 的时候可以找到一组解，不过原题限制很宽松，你甚至可以直接考虑 k = 18。

```
int a,pw[20];
void Solve(){
    cin>>a;
    pw[0]=1;
    for(int i=1;i<=18;++i) pw[i]=pw[i-1]*10;
    for(int k=1;k<=18;++k){
        ll p=pw[k-1]%a*k%a*45%a;
        ll l=a-p,r=pw[k]+a-p-1;
        if(l>=0&&l/9<=pw[k]){
            cout<<l<<" "<<r<<"\n";
            return;
        }
    }
}
```

CF356D.Bags and Coins

有s个硬币，n个包，一个包可以放在其他包里面，可以多层嵌套，如果拿出某个硬币必须要打开第i个包我们就说这个硬币在第i个包里，现已知第i个包里总共有ai个硬币，构造一种满足上述条件的方案，或确定问题无解。

$1 \leq n, s, a_i \leq 70000$

构造+bitset优化DP+分段查找优化空间。

思路：我们可以这样构造，最大的那个肯定是作为以一个树根，所以我们只要找到一个序列a1 + a2 + a3 .... + ak 并且ak为所有点中最大的那个，那么我们a1, a2, a3..., ak-1 作为单独的点，其他没有涉及到的点套在ak的里面。 现在问题变成了找出a1, a2, a3, a4, ... , ak。 可以用bitset优化普通dp，因为要找路径，空间开不下，所以需要分段。

```
#include<bits/stdc++.h>
#define fi first
#define se second
#define mk make_pair
#define pii pair<int,int>
using namespace std;
using LL=long long;
const int N=7e4+7;
const int inf=0x3f3f3f3f;

int n,s,c[N];
bool in[N];
pii a[N];
bitset<N>dp[N/10],dp2,tmp;
vector<int>edge[N];
vector<pii>dol;

bool check(int x,int y){ //背包,是否能够凑出硬币y个
    if(y<0) return false;
    dp2=dp[x/10]; //分段优化空间
    for(int i=x/10*10+1;i<=x;i++)
        dp2|=(dp2<<a[i].fi);
    return dp2[y];
}

signed main(){
    scanf("%d%d",&n,&s);
    for(int i=1;i<=n;++i){
        scanf("%d",&a[i].fi);
        a[i].se=i;
    }
    sort(a+1,a+1+n); //按容量从小到大排序
    int tar=s-a[n].fi; //减去树根的硬币数量
    tmp[0]=1;dp[0]=tmp;
    for(int i=1;i<n;++i){
        tmp|=(tmp<<a[i].fi);
        if(i%10==0) dp[i/10]=tmp;
    }
    if(check(n-1,tar)){ //能否用剩下的n-1个结点存tar的权值
        for(int i=n-1,now=tar;i>=1;--i){
            if(now>=a[i].fi&&check(i-1,now-a[i].fi)){ //作为单独的点拿出来
                now=now-a[i].fi;
                in[a[i].se]=true;
            }
        }
        for(int i=1;i<n;++i){
            if(in[a[i].se]) c[a[i].se]=a[i].fi;
            else dol.push_back(a[i]); //没有涉及的点
        }
        dol.push_back(a[n]);
```

```
        c[dol[0].se]=dol[0].fi;
        for(int i=1;i<dol.size();++i){ //将剩下的点放入树中
            c[dol[i].se]=dol[i].fi-dol[i-1].fi;
            edge[dol[i].se].push_back(dol[i-1].se);
        }
        for(int i=1;i<=n;++i){ //输出每个包的银币数量和嵌套的包
            printf("%d %d ",c[i],edge[i].size());
            for(int j:edge[i]) printf("%d ",j);
            puts("");
        }
    }else puts("-1"); //方案不存在
    return 0;
}
```

POI2013 Tower Defense game

一个国家有 n 个城市和 m 条长度均为 1 的无向道路。这个国家 k 个城市中建有防御塔，一座防御塔可以守护和它所在城市最短距离小于等于 1 的所有城市，这 k 座塔守护了所有城市。因为一次意外，这 k 座塔全部被摧毁了，国王下令重新修建防御塔。因为科技进步，新的防御塔可以守护和它所在的城市最短距离小于等于 2 的所有城市。请您找出一种方案可以用至多 k 座新型防御塔守护这个国家（保证给定的无向图可以由 k 座老的防御塔守护所有的城市）$n, m \le 500000$

思路：原来保证了k座塔可以覆盖，那么升级之后的每一座塔的覆盖范围都是原来的两倍，只要中心保证一半以上不重叠，无论怎么放，都是可行的……

```
//#pragma GCC optimize("Ofast", "inline", "-ffast-math")
//#pragma GCC target("avx,sse2,sse3,sse4,mmx")
#include<bits/stdc++.h>
using namespace std;
const int N=5e5+7;
const int mod=1e9+7;

int read(){ int x=0,f=1;char ch=getchar();while(ch<'0'||ch>'9'){if(ch=='-')
f=f*-1;ch=getchar();}while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}return
x*f;}
void write(int x){if(x>9) write(x/10);putchar(x%10+'0');}

int n,m,k,idx=0,pu[N],vis[N];
vector<int>G[N];

void dfs(int x,int f,int step){
    vis[x]=1;
    if(step>=2) return;
    for(auto to:G[x]){
        if(to==f) continue;
        dfs(to,x,step+1);
    }
}

void Solve(){
    n=read();m=read();k=read();
    for(int i=1,u,v;i<=m;++i){
        u=read();v=read();
        G[u].push_back(v);
        G[v].push_back(u);
```

```
    }
    for(int i=1;i<=n;++i){
        if(vis[i]) continue;
        dfs(i,0,0);
        ++idx;
        pu[i]=1;
    }
    write(idx);putchar('\n');
    for(int i=1;i<=n;++i) if(pu[i]) write(i),putchar(' ');
}

signed main(){
//  freopen("in.cpp","r",stdin);
//  freopen("out.cpp","w",stdout);
    int T=1;
//  cin>>T;
//  clock_t start,finish;
//  start=clock();
    while(T--){
        Solve();
    }
//  finish=clock();
//  cerr<<((double)finish-start)/CLOCKS_PER_SEC<<endl;  return 0;
}
```

参考资料

《构造》——沈洋

《几道构造相关的题目》吉如一

https://www.cnblogs.com/CJLHY/p/10098141.html

https://blog.csdn.net/zxyoi_dreamer/article/details/105558526