CSP-J 解题报告

[CSP-J 2024] 扑克牌

基本思路

本题要求出在给定的扑克牌的基础上，还需要多少张牌可以让扑克牌凑成一整套。

试题中读入的字符串每个都代表一张合法的扑克牌，从而可以使用 C++ STL 中的 set（集合）完成本题。set 可以自动去重，去除重复的牌（字符串）后，剩下的字符串就是实际拥有的不同的牌。而一副扑克牌有 52 张牌，使用 52 减去该集合的大小即可求出答案。

代码实现

```
#include <bits/stdc++.h>
using namespace std;
set <string> st;
int main() {
    int n;
    cin >> n;
    for (int i=1;i<=n;i++) {
        string s;
        cin >> s;
        st.insert(s);
    }
    cout << 52-st.size() << endl;
    return 0;
}
```

[CSP-J 2024] 地图探险

基本思路

本题考虑用模拟，递归会有爆栈风险。

设 `dx[]={0,1,0,-1},dy[]={1,0,-1,0}`，代表 d 为东南西北时坐标分别如何变化。我们先把新的坐标 `fx,fy` 借助 `dx,dy` 算出来，再判断是不是合法的点。如果是，更新坐标，否则转向。

至于记录走了几个点，我们可以记录 `vis` 数组。定义 `vis[i][j]` 为 (i,j) 点的状态，如果为 1，则走过，否则没走过。最后遍历一遍 `vis` 数组看有几个点标记为 1 即可。

代码实现

```
#include <bits/stdc++.h>
using namespace std;
bool vis[1005][1005];
char ch[1005][1005];
int dx[]={0,1,0,-1},dy[]={1,0,-1,0};
void solve() {
    int n,m,k,x,y,d;
    memset(vis,0,sizeof(vis));
```

```
    cin >> n >> m >> k;
    cin >> x >> y >> d;
    for (int i=1;i<=n;i++) {
        char s[1005];
        cin >> s;
        for (int j=1;j<=m;j++)
            ch[i][j]=s[j-1];
    }
    vis[x][y]=true;
    for (int i=1;i<=k;i++) {
        int fx=x+dx[d],fy=y+dy[d];
        if (1<=fx && fx<=n && 1<=fy && fy<=m && ch[fx][fy]=='.') {
            x=fx;
            y=fy;
        }
        else
            d=(d+1)%4;
        vis[x][y]=true;
    }
    int ans=0;
    for (int i=1;i<=n;i++)
        for (int j=1;j<=m;j++)
            ans+=vis[i][j];
    cout << ans << endl;
}
int main() {
    int T;
    cin >> T;
    while (T--)
        solve();
}
```

[CSP-J 2024] 小木棍

基本思路

由图可知：

数字	0	1	2	3	4	5	6	7	8	9
需要的木棒数	6	2	5	5	4	5	6	3	7	6

消耗木棒最多的数是 8，消耗 7 根木棒，所以尽量多摆 8。

根据特性 A 和 B，可得出：

当 $n \bmod 7$ 等于 0 时，摆 $\frac{n}{7}$ 个 8 数位最小，最优；

当 $n \bmod 7$ 等于 1 时，把第一个 8 改为 0，多出的 1 根与 $n \bmod 7$ 余下的 1 在首位摆一个数字 1，再摆 $(\lfloor \frac{n}{7} \rfloor - 1)$ 个 8，数位最小，最优。

以此类推：

当 $n \bmod 7$ 等于 2 时，在首位摆一个 1，再摆 $\lfloor \frac{n}{7} \rfloor$ 个 8，数位最小，最优。

当 $n \bmod 7$ 等于 3 时，注意，当 $n < 17$ 时，把第一个 8 改为 2，多出的 2 根与 $n \bmod 7$ 余下的 3 在首位摆一个数字 2，再摆 $(\lfloor \frac{n}{7} \rfloor - 1)$ 个 8，数位最小，最优。否则，把前两个 8 改为两个 0，多出的 2 根与 $n \bmod 7$ 余下的 3 在首位摆一个数字 2，数位最小，最优。

当 $n \bmod 7$ 等于 4 时，把第一个 8 改为 0，多出的 1 根与 $n \bmod 7$ 余下的 4 在首位摆一个数字 2，再摆 $(\lfloor \frac{n}{7} \rfloor - 1)$ 个 8，数位最小，最优。

当 $n \bmod 7$ 等于 5 时，多出的 5 根在首位摆一个数字 2，再摆 $\lfloor \frac{n}{7} \rfloor$ 个 8 数位最小，最优。

当 $n \bmod 7$ 等于 6 时，用 $n \bmod 7$ 余下的 6 在首位摆一个数字 6，再摆 $\lfloor \frac{n}{7} \rfloor$ 个 8 数位最小，最优。

代码实现

```
#include<bits/stdc++.h>
using namespace std;
const int f[10]={0,-1,1,7,4,2,6,8,10};
int T,n,d,i;
int main(){
    cin >> T;
    while(T--){
        cin>>n;
        if(n<=8)cout<<f[n];
        else{
            d=n%7;
            if(!d)for(i=1;i<=n/7;i++)cout<<8;
            if(d==1){
                cout<<10;
                for(i=1;i<n/7;i++)cout<<8;
            }
            if(d==2){
                cout<<1;
                for(i=1;i<=n/7;i++)cout<<8;
            }
            if(d==3){
                if(n==10)cout<<22;
                else{
                    cout<<200;
                    for(i=1;i<=n/7-2;i++)cout<<8;
                }
            }
            if(d==4){
                cout<<20;
                for(i=1;i<n/7;i++)cout<<8;
            }
            if(d==5){
                cout<<2;
                for(i=1;i<=n/7;i++)cout<<8;
            }
            if(d==6){
                cout<<6;
                for(i=1;i<=n/7;i++)cout<<8;
            }
        }
        cout<<endl;
    }
}
```

[CSP-J 2024] 接龙

基本思路

考虑接龙能接上的条件：以上一轮接龙序列的**最后一个元素**开头。

那么我们可以用一个二维数组 $pos_{i,j}$ 记录在第 i 次接龙中，最后一个元素能否是数字 j。

如何转移呢？假设 $pos_{i,j}$ 存在，若当前词库里存在 j 这个数字，那么这个数字后面的 $1 \sim k-1$ 个数字**均可以**作为本次接龙的结尾。

但题目要求这次接龙的人不能与上一次相同，那么将 $pos_{i,j}$ 表示的状态改为：在第 i 次接龙中，最后一个元素能为数字 j 时接的人的**编号**。

转移时稍微改一下，如果枚举到的人的编号与上一轮的一样，说明不能接到。如果一个位置可以由超过两个人接，那么说明下一轮无论谁来接都可以。

最后注意多测清空即可。

代码实现

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
#define pb push_back
const ll N=1e5+10;
ll T,n,k,q,r,c,x,l[N],pos[110][N<<1];
vector<ll> s[N];//s 开数组开不下，要用 vector 存
bool ans;
void init(){
    for(int i=1;i<=n;i++) s[i].clear();
    for(int i=0;i<=100;i++)
        for(int j=0;j<=(N<<1)-20;j++)
            pos[i][j]=-1;//-1 表示第 i 轮不能以 j 为结尾
    pos[0][1]=0;//0 表示什么人都可以接
}
int main(){
    ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
    cin>>T;
    while(T--){
        cin>>n>>k>>q;init();
        for(int i=1;i<=n;i++){
            cin>>l[i];
            for(int j=0;j<l[i];j++) cin>>x,s[i].pb(x);
        }
        for(int i=1,j;i<=100;i++)
            for(int p=1;p<=n;p++)
                for(int x=0;x<s[p].size();x++){
                    j=s[p][x];
                    if(pos[i-1][j]!=-1&&pos[i-1][j]!=p){//上一轮 j 可以作为结尾
                        x++;
                        for(int q=1;q<k&&x<s[p].size();q++,x++){//x 后 1~k-1 个字符都可以
                            j=s[p][x];
                            if(pos[i][j]==-1) pos[i][j]=p;//编号为 p 的人接了
                            else if(pos[i][j]!=p) pos[i][j]=0;//有两个不同的人可以接，说明下一轮谁都可以接
```

```
                            if(pos[i-1][j]!=-1&&pos[i-1][j]!=p) q=0;//如果当前位置可以接，那么当前位置后面的 k-1 个位置也可以
                        }
                        x--;
                    }
                }
        while(q--) cin>>r>>c,cout<<(pos[r][c]!=-1)<<"\n";
    }
    return 0;
}
```