此题需要求解最少的成语数，因此可以很容易联想到广度优先搜索。对于此题，如果直接使用原有数据进行搜索，每次都从第一个开始搜索，那么数据就很容易超时，因此需要借助其它操作对其进行优化。我的优化是先将输入的成语按照第一个数字进行排序，然后开辟一个数组，将成语的第一个数字所在的位置记录下来并找到开始词语所在的位置

```
long add=0;
nb[word[1][1]] =1;
for(i=2;i<=m;i++)
{
    if(word[i][1]!=word[i-1][1]) nb[word[i][1]]=i;
    if(word[i][1]==a&&word[i][2]==b&&word[i][3]==c&&word[i][4]==d) add=i;
}
```

接下来的操作就是进行搜索了，此时我们并不是从第一个搜到第一个而是从第一个成语的第四个字所在的位置开始搜索，当搜到比第四个字大的位置时，就退出此轮循环，如果第四个字没有对应的位置，那么说明这个成语不能继续接龙，因此搜索的循环语句可以这么写

```
long y1=word[step[head]][4];
for(i=(nb[word[step[head]][4]]==0?m+1:nb[word[step[head]][4]]);i<=m;i++)
{
    long x1=word[i][1];
    if(x1>y1) break;
```

y1为成语的尾巴，nb数组存的是数的位置，x1为成语的开头，这样搜索就减少了对数据的重复访问，最大的访问次数就是m,从而就不会超时了。