

洛谷 / 题目列表 / 题目详情 / 查看题解

应用 »

题库

题单

比赛

记录

讨论

P1019 [NOIP2000 提高组] 单词接龙 题解

返回题目

本题目已不接受新题解提交

当题目的题解数量、做题思路已足够丰富，题目过于简单，或处于月赛保护期时，题解提交入口会被关闭。

不要把题解发布在题目讨论区。

题解仅供学习参考使用

抄袭、复制题解，以达到刷 AC 率/AC 数量或其他目的的行为，在洛谷是严格禁止的。

洛谷非常重视学术诚信。此类行为将会导致您成为**作弊者**。具体细则请查看洛谷社区规则。

提交题解前请务必阅读《洛谷主题库题解规范》。

洛谷网校

2023暑假

J组高分与S组/NOIP课程

授课计划

基础提高
衔接计划

进阶计划
前/后期

省选计划
前/后期

目标

CSP-J高分

CSP-S获奖

NOIP高分

冲省队/NOI

刷题巩固

NOIP
冲刺计划

洛谷推荐

2023年
秋季开设



68 篇题解

 **Hinanawi_Feng** 创建时间：2017-10-27 11:42:19

首先在题意上可能有些误解。

- 两个单词合并时，合并部分取的是**最小重叠部分**
- **相邻的两部分不能存在包含关系**就是说如果存在包含关系，就不能标记为使用过。
- 每个单词最多出现两次。

(其实也就是读题问题。这些都是我所犯的错误，希望大家能注意一下)

好了。然后是解题思路。

首先是预处理，用yc[i][j]来存储 第i个单词 后连接 第j个单词 的 最小重叠部分 (mt函数)

后来预处理完了之后就是深搜：

先从第一个到最后一个单词看一看哪个单词是指定字母为开头的，作为深搜的第一个单词，同时标记使用过一次 (vis[i]++)

然后继续搜吧。

以下是代码。mt函数可能有点难理解。拿草纸模拟一下就能很直观知道在干什么了。

```
#include<cstdio>
#include<iostream>
#include<string>
#include<cmath>
using namespace std;
int n;//单词数
string tr[30];//存储字符串
int yc[30][30];//两个字母的最小重叠部分
int vis[30];//判断单词使用频率。
int mt(int x, int y) { //mt函数，返回x单词后连接一个y单词的最小重叠部分
    bool pp=true;
    int ky=0;
    for(int k=tr[x].size()-1;k>=0;k--) { //从x单词尾部向前看看最小重叠部分是从哪里开始的，以为因为是倒着来，所以保证是最小的
        for(int kx=k;kx<tr[x].size();kx++) {
            if(tr[x][kx]!=tr[y][ky++]) {
                pp=false;
                break;
            }
        }
        if(pp==true) { //如果说当前以k为开头的前一个单词后缀，是后面单词的前缀，就马上返回重叠部分。（tr[x].size()-k是找出来的规律）
            return tr[x].size()-k;
        }
        ky=0;
        pp=true; //不行就继续
    }
    return 0;
} //可能这里有点难理解。可以手动模拟一下
char ch;//开头字母
int ans=-1;//答案
int an=0;//每次搜到的当前最长串
void dfs(int p) { //p为尾部单词编号(p的后缀就是“龙”的后缀，因为p已经连接到”龙“后面了)
    bool jx=false;
    for(int j=1;j<=n;j++) {
        if(vis[j]>=2) continue; //使用了两次就跳过
        if(yc[p][j]==0) continue; //两单词之间没有重合部分就跳过
        if(yc[p][j]==tr[p].size() || yc[p][j]==tr[j].size()) continue; //两者存在包含关系就跳过
        an+=tr[j].size()-yc[p][j]; //两单词合并再减去最小重合部分
        vis[j]++; //使用了一次
        jx=true; //标记一下当前已经成功匹配到一个可以连接的部分
        dfs(j); //接上去
        an=tr[j].size()-yc[p][j]; //回溯，就要再减回去那一部分长度
        vis[j]--; //回溯，使用--
    }
    if(jx==false) { //jx==false说明不能再找到任何一个单词可以相连了
        ans=max(ans, an); //更新ans
    }
    return;
}
```

应用 >>

题库

题单

比赛

记录

讨论

```
}
int main() {
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
        cin>>tr[i];
    cin>>ch;
    for(int i=1;i<=n;i++){
        for(int j=1;j<=n;j++){
            yc[i][j]=mt(i,j);
        }
    }
    //预处理yc数组。yc[i][j]就表示，i单词后连接一个j单词的最小重叠部分
    //比如 i表示at,j表示att. yc[i][j]就为2 但是yc[j][i]就为0.
    //预处理是一个关键

    for(int i=1;i<=n;i++){//从头到尾看一下有没有以指定开头字母为开头的单词
        if(tr[i][0]==ch){//如果有，就以当前单词为基准进行搜索。
            vis[i]++;//使用过一次
            an=tr[i].size();//更新当前串长度
            dfs(i);//接上
            vis[i]=0;//消除影响
        }
    }
    printf("%d",ans);
    return 0;
}
```

另外思路是zcy dalao所提供的。在做题的时候又发现了许多新的问题，所以贴出来既是给自己警示也分享给大家。

ps:本人第一次发题解，有什么错误请指出来。谢谢QAQ

👍 710 💬 265 条评论

 Chardo 创建时间：2018-02-10 21:54:29

首先明确几个要点：

- 一个单词最多用两次
- 单词可以不全用完
- 不可以包含：一旦包含了和不用岂不是一样
- 按照贪心原则，重叠部分应该越少越好

Talk is cheap, show me the code.

代码结构非常简明，canlink()返回最小重叠部分的长度，没有返回0。solve()进行dfs搜索。主函数读取、调用和输出。

希望大家能养成简洁清晰的代码风格，非常有利于互相的review和自己查错

```
C++

#include<bits/stdc++.h>
using namespace std;
string str[20];
int use[20], length = 0, n;
int canlink(string str1, string str2) {
    for(int i = 1; i < min(str1.length(), str2.length()); i++) { //重叠长度从1开始，直到最短的字符串长度-1（因为不能包含）
        int flag = 1;
        for(int j = 0; j < i; j++)
            if(str1[str1.length() - i + j] != str2[j]) flag = 0; //逐个检测是否相等
        if(flag) return i; //检测完毕相等则立即return
    }
    return 0; //无重叠部分，返回0
}

void solve(string strnow, int lengthnow) {
    length = max(lengthnow, length); //更新最大长度
    for(int i = 0; i < n; i++) {
        if(use[i] >= 2) continue; //该字符串使用次数需要小于2
        int c = canlink(strnow, str[i]); //获取重叠长度
        if(c > 0) { //有重叠部分就开始dfs
            use[i]++;
```

应用 >>

题库

题单

比赛

记录

讨论

```
        solve(str[i], lengthnow + str[i].length() - c);
        use[i]--;
    }
}

main() {
    cin >> n;
    for(int i = 0; i <= n; i++) use[i] = 0, cin >> str[i]; //str[n]为开始字符
    solve(''+str[n], 1); //有必要解释一下开始阶段。为了指定第一个字符，而且因为canlink需要重叠部分小于最短长度-1，所以要从前面添加一个无意义字符
    cout << length ;
}
```

467

152 条评论



ShawnZhou 创建时间：2017-09-05 23:54:14

安利一发自己的博客：<http://www.cnblogs.com/OierShawnZhou/>

（我平常写的题解都会往博客里发，欢迎各位大佬前来拍砖）

谨以此题解纪念我的洛谷橙名，我会继续努力。

基本思路是搜索。

处理的难点在于对重叠部分的处理。

单词的使用次数很好判断，开一个数组即可，和正常向dfs的vis数组差不多。

但对于重叠部分的处理，我想细说一下。

因为连接起来的单词要最长，所以对比是选择从上一个单词的末尾与当前单词的开头进行比对，如果发现不符那就不能匹配。

这里我借鉴了一位大神的思路，使用一个check函数，用来比较两个串s和m的长度为k的接口能不能匹配。

判断方式有两种，第一种就是我用的这样按字符比较，如果发现某处不匹配立即返回false。还有一个方法是用string里面的substr，我就不以试一下。

我们假设接口长度为k，串s的长度为lens，然后我们从0到k枚举，判断s[lens-k+i]是不是等于m[i]。

这个式子怎么来的呢？接口前面的串是s，后面的串是m，那么很显然，s串的接口最开始应该是lens-k处，然后在后面加上一个枚举的i就可匹配（我们的i是从0开始枚举的）。

还有一些细节问题。如果我们在接龙的时候发现我们现在要接的龙还不如之前某一次接过的长，那么这个接龙方案肯定不是最优的，所以要剪枝。

（想一下深搜时的遍历过程，这句话便不难理解）

使用我这个方法，可能有一个奇怪的想法有同学没想到，那就是在进行拼接操作时，要注意使用给定的串的副本（即复制一份原来的串）进行拼接。这是因为，如果你把原串改变了，而且这个串还不是最优的，那就完了，回溯不回去了。

具体到操作上，首先，我们从1到n枚举每个短字符串，如果它已经被用了两次则continue，然后我们求出当前短串的长度，从1到这个长度枚举k（自然是接口越短融合串越长嘛）然后执行拼接操作，记录一下最大长度，再加上回溯就好了。

拼接操作和check函数很像，具体到代码上大家就能看明白了。

```
#include <iostream>
#include <cstdio>
#include <cstring>
#include <string>
#define maxn 100
using namespace std;
int n;
int ans = 0;
string word[maxn]; // 字符串数组，用来存储单词
string beginn; // 用来存储开头字符
int used[maxn]; // 这个就是用来记录dfs时候每一个单词被使用了几次的数组

bool check(string s, string m, int k) { // 重点一，check函数判断接口可行性，k代表接口长度，以下同
    int lens = s.length();
    for (int i=0; i<k; i++) {
        if (s[lens-k+i] != m[i]) // 我讲过了
            return false;
    }
    return true;
}
```

应用 »

题库

题单

比赛

记录

讨论

```
bool check(string s, string m, int k) {
    return false;
}

return true;
}

void add(string &s, string m, int k) { // 拼接操作，因为要把m接到s上，所以对于s串不可以传参，因为我们要试图改变这个串
    int lenm = m.length();
    for (int i=k; i<lenm; i++)
        s+=m[i]; // C++字符串类型特性操作，支持+=符号直接拼接
}

void dfs(string now) { // 这只是一个看似平淡无奇的dfs
    int x = now.length();
    ans = max(ans, x); // 每次拼接之后更新一下答案
    for (int i=1; i<=n; i++) {
        if (used[i]>=2) // 如果有一个单词用完了，那这个单词就不能选了
            continue;
        int maxk = word[i].length();
        for (int j=1; j<=maxk; j++) { // 枚举接口长度
            if (check(now, word[i], j)) {
                string temp = now; // 重点二，使用字符串副本进行拼接
                add(temp, word[i], j);
                if (temp==now) // 拼完之后如果发现长度没增加，也就是和原串一样，那这次拼接没有意义，剪掉
                    continue;
                used[i]++;
                dfs(temp);
                used[i]--; // 这只是一个看似平淡无奇的回溯
            }
        }
    }
}

int main() {
    cin >> n;
    for (int i=1; i<=n; i++)
        cin >> word[i];
    cin >> beginn;
    dfs(beginn);

    cout << ans << endl;
    return 0;
}
```

👍 207 💬 83 条评论

 **Strong_Jelly** 创建时间：2019-04-29 11:40:44

单词接龙

题目描述

单词接龙是一个与我们经常玩的成语接龙相类似的游戏，现在我们已知一组单词，且给定一个开头的字母，要求出以这个字母开头的最长的“龙”（出现两次），在两个单词相连时，**其重合部分合为一部分**，例如 `beast`和`astonish`，如果接成一条龙则变为`beastonish`，另外相邻的词系统，例如`at`和 `atide`间不能相连。

这道题需要注意三点：

- 1. 每个单词最多在龙里出现两次
- 2. 不能在龙里出现包含关系
- 3. 可以有重合部分

重点来了（敲黑板）

求重合部分可以从已经在龙里的最后一个单词的最后面开始枚举（从后向前枚举）直到找到最后一个单词的一个字母和想要接龙的单词的单词的字母的位置记录为`k`，然后从已经在龙里的那个单词从`k`到尾进行循环枚举，同时想要接龙的单词从头到`k`进行枚举，如果有不相等的一个回那个单词的长度 - (`k + 1`)



应用 »



题库



题单



比赛



记录



讨论

code~:

```

#include <bits/stdc++.h>
using namespace std;
int n, vis[100001], ans, maxn, l[100001], p;
string s[10001];
char ch;
int find(int x, int y)//是否能合并
{
    int lx = l[x];
    int ly = l[y];
    for(int i = lx - 1; i >= 1; i--)
    {
        if(s[x][i] == s[y][0])//发现字母相等!
        {
            int k = 0;
            for(int j = i + 1; j <= lx - 1; j++)
            {
                k++;//同时枚举
                if(s[x][j] != s[y][k])//有一个字母不相等就返回0
                {
                    return 0;
                }
            }
            return ly - (k + 1);//全相等返回长度
        }
    }
    return 0;
}
void dfs(int x)//搜索单词
{
    for(int i = 1; i <= n; i++)
    {
        if(vis[i] < 2 && find(x, i))//不能用两次且可以接龙
        {
            vis[i]++;//用的次数++
            ans += find(x, i);//加上新单词的长度
            dfs(i);//接着这个单词继续搜
            vis[i]--;//回溯
            ans -= find(x, i);//回溯
        }
    }
    if(ans > p)//求最大
    {
        p = ans;
    }
}
int main()
{
    scanf("%d", &n);
    for(int i = 1; i <= n; i++)
    {
        cin>>s[i];
        l[i] = s[i].length();//存下所有单词的长度
    }
    cin>>ch;
    for(int i = 1; i <= n; i++)
    {
        if(s[i][0] == ch)//得等于首字母才能接
        {
            ans = l[i];//先加上首单词的长度
            vis[i]++;//首单词用过了
            dfs(i);
            vis[i]--;//回溯
            if(p > maxn)//求最大
            {
                maxn = p;
            }
        }
    }
    printf("%d", maxn);
    return 0;
}

```

应用 >>

题库

题单

比赛

记录

讨论

已AC

👍 131

💬 32 条评论

👤 xiejinhao  创建时间：2019-04-22 21:15:29

P1019 单词接龙 题解

Dalao的思路没看懂，于是有了这篇题解

其实这题不难，但是要把思路理清。

这里要纠正一下题目，并不是包含关系就不能连接，比如 ata 和 atatata 是可以连接的。因此，题目应理解为：**若不能使单词长度增加，则不能连接**。

思路如下：

1. 先输入，然后直接进入DFS深搜，在深搜里判断是这两个单词是否能连接；
2. 判断两个单词是否能接，需要保证这两个数都没被使用过两次，那么需要用数组存一下每个数使用的次数；
3. 若两个数可以连接，进入判断。为了保证龙的长度最大，我们可以从龙头的尾巴开始搜索，如果在该字符串的的不是第1位字符搜索到了i 字符可能可以拼接，从这一次的位数再正过来搜索，搜到有一位不同就跳出，否则让重复部分 +1，这样，当前的上一个单词和下一个单词了；
4. 如果两个单词可以连接，以新连接的单词作为开头，搜索，长度+这个单词长度-重复部分长度；
5. 记得回溯；

- 这里有个问题，如果我在struct里面定义一个

```
int left=2;
```

然后神奇的Devc++就给了我一串警告；不管了，反正A了；

然后重叠部分的判断就成为了难点，下面我给出代码以及解释，耐心看下去：

```
struct word{
    string line;
    int left=2;
}w[25];
inline int find(int last,int next)
//last表示上一个单词序号，next表示要连接的单词序号
//inline 用于加速，写不写无所谓
{
    int lenl=w[last].line.length(),lenn=w[next].line.length();
//这个长度可以在输入的时候就处理好，这么写麻烦了，大家自行优化；
    for(int i=lenl-1;i>=1;--i)//倒着搜
    {
        if(w[last].line[i]==w[next].line[0])
        {
            //搜到了有一位相同，可能可以连接，“正”过来搜
            int k=i,j=0,c1=0;
            while(j<lenn && k<lenl)
            //要保证搜索的长度小于前一个单词和后一个单词的长度的长度，有一个不成立都代表搜索结束了；
            {
                if(w[last].line[k]==w[next].line[j])++c1;
                else return 0;
                ++k;++j;
            }
            if(k==lenl && j<lenn)return c1;
            //如果前一个单词搜到了最后一个，并且搜到的小于后面那个单词的长度（不然就包含在这个单词里面，不能使龙长度增加，无效，不能连接）
            else return 0;
            //否则搜到一个不一样，返回0，重叠部分为0；
        }
    }
    return 0;
//把前一个单词搜完了还没找到，也代表不能连接，重叠部分0；
}
```

- 然后DFS不用讲了，判断返回值是否为真，为真就继续搜索这个单词的下一个，否则继续循环，定义一个标志变量为 false，能继续搜就

来的 false，代表搜完了，没有继续能连接的数，最大值取上一次最大值和这次长度中的最大值。

给出程序完整代码：

```
//认真看，杜绝抄袭
#include<stdio.h>
#include<iostream>
#include<string>
using namespace std;
struct word{
    string line;
    int left=2;//定义每个单词的初始次数（不知道为什么有警告）；
}w[25];
int n,maxA;
inline int find(int last,int next)//解释过不解释了；
{
    int lenl=w[last].line.length(),lenn=w[next].line.length();
    for(int i=lenl-1;i>=1;--i)
    {
        if(w[last].line[i]==w[next].line[0])
        {
            int k=i,j=0,c1=0;
            while(j<lenn && k<lenl)
            {
                if(w[last].line[k]==w[next].line[j])++c1;
                else return 0;
                ++k;++j;
            }
            if(k==lenl && j<lenn)return c1;
            else return 0;
        }
    }
    return 0;
}
inline void dfs(int now,int length)
{
    bool flag=false;//所谓的标志变量
    for(int i=1;i<=n;i++)
    {
        if(!w[i].left)continue;//一个单词只能用两次；
        else
        {
            int x=find(now,i);
            if(x)
            {
                flag=true;//搜到变成真
                --w[i].left;
                length+=(w[i].line.length()-x);
                dfs(i,length);
                ++w[i].left;
                length-=(w[i].line.length()-x);
                //这边不仅编号要回溯，长度也要回溯（卡了我很久）
                //至于为什么，就是因为dfs结束，返回到了这里，那么dfs这一趟完了，这个长度和次数还要用到下一次搜索当中去。
            }
            else flag=false;//标志变量为假就是搜完了，没有可连接单词；
        }
    }
    if(!flag) maxA=max(maxA,length);
}
int main()
{
    scanf("%d",&n);
    //for(int i=1;i<=n;i++)w[i].left=2;
    for(int i=1;i<=n;i++)cin>>w[i].line;
    char p=getchar();cin>>p;
    //输入完成；
    for(int i=1;i<=n;i++)
    {
        if(w[i].line[0]==p)
        {
            --w[i].left;
            dfs(i,w[i].line.length());
            ++w[i].left;
        }
    }
}
```


应用 >>

题库

题单

比赛

记录

讨论

```
        //这边也要回溯，同上。
    }
}
printf("%d", maxA); //最后输出最大值;
return 0;
}
```

我觉得我写的程序可读性还是不错的，不妨点个赞？

👍 107 💬 21 条评论

 Mickey_snow 创建时间：2018-01-02 21:48:02

重点：单词重叠的部分不好操作

在搜索的基础上代码需要有所修改。

首先，题目包含的接龙要求大致有以下几点：1.同一个单词最多出现2次 2.相邻的两部分不能存在包含关系 3.在两个单词相连时，其重合部分我们很容易就可以想到，使用一个结构体用于存储各个单词的数据，包括单词本身，它的实际长度以及可供使用的剩余次数（初始值为2）。如

```
struct words
{
    int left = 2, tail = 0;           //剩余的使用次数 单词实际长度
    char word[101];                  //内容
}words[22];
```

我们可以使用一个字符变量start用来存储开头的字母，然后即可开始搜索，由于start当中的字符并不会被计入总长度，所以可以先通过一个单词，然后再以这个单词作为开头，进行深搜。

```
cin >> start;
for (int i = 0; i < wordTot; i++)
    if (words[i].word[0] == start)
    {
        words[i].left--;
        dfs(words[i].word, words[i].tail); //目前龙中最后一个单词 长度
        words[i].left++;
    }
```

由于单词接龙中相邻的两部分不能存在包含关系，所以无论龙有多长，只需要考虑最后一个加入的单词能够和那些单词拼接即可。在这里，juj来判断两个单词之间重合部分长度的最小值。

```
int juj(char a[], char b[])
{
    int al = strlen(a), bl = strlen(b), j;
    bool flag = true;
    for (int i = 1; i < al && i < bl; i++)
    {
        flag = true;
        for (j = 0; j < i; j++)
            if (a[al - i + j] != b[j]) { flag = false; break; }
        if (flag)
            return i;
    }
    return 0;
}
```

这个函数主要的原理是：在a，b两个字符串当中一个个地尝试可能重合部分的长度，并把这个长度作为一个数值返回。如果这个长度已经不小于长度，那么说明这两个字符串没有符合题意的重合部分，juj将返回0。

处理好了单词重合部分，剩下的就是搜索了。这部分比较简单，直接照着深搜的伪代码模板进行填充相应的代码即可。

```
void dfs(char tail[], int num)
{
    _max = max(_max, num); //更新最大值
    int a;                 //两个单词重合部分的长度
    for (int i = 0; i < wordTot; i++)
        if (words[i].left > 0)
        {
```



应用 >>



题库



题单



比赛



记录



讨论

```

        a = juj(tail, words[i].word);
        if (a != 0)           //判断是否符合题意
        {
            words[i].left--;
            dfs(words[i].word, num + words[i].tail - a);    //下一步搜索
            words[i].left++;
        }
    }
}

```

这样，程序就基本完成了，下面是完整的代码：

```

#include<iostream>
#include<cstring>
#include<algorithm>
using namespace std;
int wordTot, _max = 0;
struct words
{
    int left = 2, tail = 0;           //剩余的使用次数 单词实际长度
    char word[101];                 //内容
}words[22];
int juj(char a[], char b[])
{
    int al = strlen(a), bl = strlen(b), j;
    bool flag = true;
    for (int i = 1; i < al && i < bl; i++)
    {
        flag = true;
        for (j = 0; j < i; j++)
            if (a[al - i + j] != b[j]) { flag = false; break; }
        if (flag)
            return i;
    }
    return 0;
}
void dfs(char tail[], int num)
{
    _max = max(_max, num);           //更新最大值
    int a;                           //两个单词重合部分的长度
    for (int i = 0; i < wordTot; i++)
        if (words[i].left > 0)
        {
            a = juj(tail, words[i].word);
            if (a != 0)           //判断是否符合题意
            {
                words[i].left--;
                dfs(words[i].word, num + words[i].tail - a);    //下一步搜索
                words[i].left++;
            }
        }
}
int main()
{
    char start;
    cin >> wordTot;
    for (int i = 0; i < wordTot; i++)
    {
        cin >> words[i].word;
        words[i].tail = strlen(words[i].word);
    }
    cin >> start;
    for (int i = 0; i < wordTot; i++)
        if (words[i].word[0] == start)
        {
            words[i].left--;
            dfs(words[i].word, words[i].tail);    //目前龙中最后一个单词 长度
            words[i].left++;
        }
    cout << _max << endl;
    return 0;
}

```



应用 >>



题库



题单



比赛



记录



讨论



小小怪下士_



创建时间：2019-11-08 22:37:04

这应该是退役前最后一篇题解了吧！（真好）

思路：dfs暴搜 考虑一下最优的情况 比如acac和acacb 这时我们可以有两个选择：

第一个ac相连或者和第二个ac想连。显而易见尽量让长度越长越好，所以如果遇见这样的情况就和最后一个想连。

然后在考虑如何去搜：

我们1到n循环能连的连起来在dfs一下如果不能连了自然会回溯到上一步来。我们只需在过程中找个最大长度即可。

最麻烦的就是然后判断这是不是合法的和重叠的长度是多少

比如 acac 和acbc来说我们用a出现的位置来和acbc的头来比如果相同就++；直到找到不相同或者全部找完。如果停止的位置是acac长度最后选择一个最优的。当前重叠长度就是acac的长度减去我们刚开始的位置。

思路大概就是这样，最后把当前的总长度减去总重叠的长度取个max就OK啦！

代码如下：

```

#include<bits/stdc++.h>
using namespace std;
const int N=30;
int chu[N];
string t[N],str;
int n,m,maxx;
int gao(string x,string q)//找是否可以相连
{
    for(int i=x.size()-1;i>=0;i--)//倒的找找到一个就退出这样肯定是最优的
    {
        if(x[i]==q[0])//如果和匹配开头一样就开始找是否合法
        {
            int l=i;//当前开始的位置
            for(int j=0;j<q.size();j++){if(x[l]==q[j])l++;else break;}//如果一样的话就一直找否则就退出来
            if(l==x.size())return l-i;//如果找完了就算出有多少个是匹配的这个就是重复的值
        }
    }
    return 0;//没有就返回0回去;
}
void dfs(int fu,string x)
{
    maxx=max(maxx,(int)x.size()-fu);//用字符的长度减去重复的长度
    for(int i=1;i<=n;i++)
    {
        int duan=0;
        if(chu[i]==2)continue;//如果用过两次就直接跳
        duan=gao(x,t[i]);//duan为他们重叠的长度
        if(!duan)continue;//如果为0没重叠就直接跳过
        chu[i]++;//这个位置标记一下表示用过
        dfs(fu+duan,x+t[i]);//重叠的部分相加字符串更新
        chu[i]--;//回溯
    }
    return ;//没找到一个匹配就返回
}
int main()
{
    cin>>n;
    for(int i=1;i<=n;i++)cin>>t[i];//把每个都存起来
    cin>>str;
    dfs(0,str);
    cout<<maxx;//最大长度
    return 0;
}

```

56



22 条评论



MorsLin



创建时间：2018-03-24 11:40:16

当时一看到这题，蒟蒻的我还以为是DP，结果发现标签是搜索.....



应用 >>



题库



题单



比赛



记录



讨论

这道题的难点在于思路和预处理，真正的搜索实现起来并不难。我们可以用一个贪心的思路，开一个dic数组记录每个单词的最小重复部分，很方便地查阅dic数组，而不是每次再计算一遍。

预处理是长这样的：

```
void f(string a,string b,int x,int y)
{
    int a1=a.size()-1,b1=b.size()-1;
    for(int i=0;i<=b1;i++) //从第一个开始枚举
    {
        if(a[0]==b[i]) //如果a的首字母和b中间的字母相同，则判断它们能不能接在一起
        {
            int pos=0,tot=0; //pos是当前a的第几个字母，tot是a和b的重合部分长度
            for(int j=i;j<=b1;j++)
            {
                if(a[pos]==b[j])
                {
                    tot++;
                    pos++;
                    if(j==b1&&tot!=min(a1,b1)+1) //如果枚举到了最后，并且a和b没有包含关系，说明可以这么接
                        dic[x][y]=tot; //记录最小重叠部分的长度
                    //之所以不break，是因为后面可能还会枚举到更小的接法
                    //比如 chsese 和 sesettt 显然 chsesesettt 要比chseseettt更优
                }
            }
            else break;
        }
    }
}
```

这样就把每个单词的相互重叠部分全记录下来了,最后的处理出来的dic[x][y]是把x接在y后面的重复部分长度

之后我们就可以愉快的搜索了，搜索本身并不难，只需要注意每个单词可以用两次，以及接上新单词的“龙”的长度就可以了。

完整代码：

```
#include<iostream>
using namespace std;
int n,dic[21][21],vis[21],ans;
string words[21];
char s;
void f(string a,string b,int x,int y)
{
    int a1=a.size()-1,b1=b.size()-1;
    for(int i=0;i<=b1;i++) //从第一个开始枚举
    {
        if(a[0]==b[i]) //如果a的首字母和b中间的字母相同，则判断它们能不能接在一起
        {
            int pos=0,tot=0; //pos是当前a的第几个字母，tot是a和b的重合部分长度
            for(int j=i;j<=b1;j++)
            {
                if(a[pos]==b[j])
                {
                    tot++;
                    pos++;
                    if(j==b1&&tot!=min(a1,b1)+1) //如果枚举到了最后，并且a和b没有包含关系，说明可以这么接
                        dic[x][y]=tot; //记录最小重叠部分的长度
                    //之所以不break，是因为后面可能还会枚举到更小的接法
                    //比如 chsese 和 sesettt 显然 chsesesettt 要比chseseettt更优
                }
            }
            else break;
        }
    }
}

void dfs(int pos,int sum)
{
    ans=max(ans,sum); //实时更新ans
    for(int i=1;i<=n;i++)
    {
        if(dic[i][pos]&&vis[i])
        {
            vis[i]--;
        }
    }
}
```



应用 >>

 题库

 题单

 比赛

 记录

 讨论

```
int suml=sum+words[i].size()-dic[i][pos]; //接上新单词“龙”的长度为=旧的长度+新单词长度-重复部分长度
dfs(i, suml); //接上新单词继续搜索
vis[i]++;

    }
}
}
int main()
{
    cin>>n;
    for(int i=1;i<=n;i++)
    {
        cin>>words[i];
        vis[i]=2; //初始化vis数组，每个单词能用两次
    }
    cin>>s;
    for(int i=1;i<=n;i++)
        for(int j=1;j<=n;j++)
            f(words[i],words[j], i, j); //预处理dic数组


    for(int i=1;i<=n;i++)
    {
        if(words[i][0]==s) //找到开始部分
        {
            vis[i]--;
            dfs(i, words[i].size()); //深搜
            vis[i]++;
        }
    }
    cout<<ans;

    return 0;
}
```

- 广告时间

在下的[洛谷博客](#)和[博客园博客](#)

👍 37 🗨 7 条评论



Pjone 创建时间：2018-10-24 21:20:53

本题是一道很考验人的细节题

对于这一道题，建议大家借鉴思路后坚持调出来。

思路

- 1 .深搜枚举每一个单词，筛选出正确答案。 2 .找到某一个单词中的某一个点，符合需要接上的单词龙， 以此为基点相后扫，筛选出最大值

注意

- 1 .每个单词最多出现两次. 2 .相邻的两部分不能存在包含关系. (这将成为我们剪枝，排错的依据)

请各位神犇不耻下看

```
#include<bits/stdc++.h>
using namespace std;
int ans=0,n;//记录答案与单词串数。
int maxn=0;//用于筛选最大值。
int len[25];//记录单词串长度。
char a[25][25];//记录单词矩阵。
int t[25];//记录单词出现次数，t<=2。
void dfs(int x)//话不多说，开搜！
{
    int y=0;//这个是用来判断是否能够在找到符合条件的要求(若y未改动，则说明搜索到了边界。)
    for(int i=1;i<=n;i++)
        //遍历每一个单词串。
        if(t[i]<2)//保证次数未超，注意是小于2而不是小于等于（若等于则在之后再被用上使用数变为3，不符合要求。）
            for(int j=0;j<len[x];j++)
```



应用 >>



题库



题单



比赛



记录



讨论

```

    //从0开始遍历每一个字母。
    if(a[i][0]==a[x][j])
    {
        //找到第一位与待接单词长龙某一位相等的单词，记下位置，扫。
        if(j==0)
        {
            if(len[i]!=len[x]) continue;
        }
        //判断一下，是否会重合。（其实有点小问题。）
        int l=1;
        bool b=0;
        for(int h=j+1;h<len[x];h++)
            if(a[x][h]!=a[i][l++])
            {
                b=1;
                break;
            }
        //若扫到一位不相同，则不合法。
        if(b==1) continue;
        else
        {
            y++;ans+=len[i]-1;t[i]++;
            //改变y值，表示能找到，注意ans+的是len[i]-j单词的第一位是从0开始的。
            dfs(i);
            y--;ans-=len[i]-1;t[i]--;//回溯一波。
        }
    }
}

}

if(y==0)
{
    if(ans>maxn) maxn=ans;
    return;//若搜完，取最大值。
}
}

int main()
{
    //freopen("dragon.in","r",stdin);
    //freopen("dragon.out","w",stdout);
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
    {
        scanf("%s",a[i]);
        len[i]=strlen(a[i]);
    }
    scanf("%s",a[n+1]);
    for(int i=1;i<=n;i++)
    {
        if(a[i][0]==a[n+1][0])
        {
            //注意题目条件，找到符合条件的龙头。
            ans=len[i];t[i]++;
            dfs(i);
            ans=0;t[i]--;
        }
    }
    printf("%d",maxn);
    return 0;
}

```

👍 31



💬 13 条评论



紫翠麒麟

创建时间：2019-02-10 19:57:11

注：本题不一样的解法

其实这道题可以状压dp的

我们先谈一下怎么状压dp：

状态定义：

$dp[i][j]$ 表示情况 i 下以 j 号单词结尾所能达到的最大长度

我们可以用 1 个长度为 n 的 01 串（即二进制）表示单词使用情况，1 表示使用过，0 表示未使用。



应用 >>

题库

题单

比赛

记录

讨论

举几个栗子：

- $n=4$ 时，1101表示只使用过1，3和4号单词；
- $n=5$ 时，10010表示只使用过5号和2号单词。

注意：在我写的（不包括你）01串中，第*i*位表示的是第*n-i+1*号单词的情况原因后面会提到

状态转移方程：

$$dp[next][j] = \max\{dp[next][j], dp[now][k1...kn] + len[k1...kn][j]\}$$

其中 $k1...kn$ 表示枚举*n*个单词， $len[k][j]$ 表示将*j*号单词接在*k*号后面长度的增加量（可预处理）


dp部分代码

```
for(int i=1;i<(1<<n);i++){\\总共有2^n种使用情况
    for(int j=1;j<=n;j++){
        if(i&(1<<(j-1)))continue;\\判断j在i情况下有没有被使用过
        for(int k=1;k<=n;k++){
            if((i&(1<<(k-1)))&&len[k][j]!=-1&&dp[i][k]!=-1){\\判断能否转移：1.k在情况i下使用了 2.j能接在k后面
                dp[i|(1<<(j-1))][j]=max(dp[i|(1<<(j-1))][j],dp[i][k]+len[k][j]);\\转移：next=i|(1<<(j-1)),now=i;
                ans=max(ans,dp[i|(1<<(j-1))][j]);\\注意不断更新最终结果
            }
        }
    }
}
```


还有一点，本题中由于么个单词可以最多用两次，所以应该开个 $2 * n$ 长度的数组，将每个单词再复制一遍

其他没什么好说的了，详见代码。。。


```
#include<bits/stdc++.h>
using namespace std;
int n,len[1001][1001],dp[1<<16][17],ans;
string s[1001];
char tou;
void pre(){\\预处理len数组，有点小麻烦
    for(int i=1;i<=n;i++){
        for(int j=1;j<=n;j++){
            if(i==j)continue;
            int lenans=0;
            for(int lenth=1;lenth<min(s[i].size(),s[j].size());lenth++){
                bool flag=true;
                for(int il=s[i].size()-lenth,jl=0;il<=s[i].size()-1,jl<=lenth-1;il++,jl++){
                    if(s[i][il]!=s[j][jl]){
                        flag=false;
                        break;
                    }
                }
                if(flag==true)lenans=lenth;
                if(lenans!=0)break;
            }
            if(lenans==0)len[i][j]=-1;
            else len[i][j]=s[j].size()-lenans;
        }
    }
}
int main(){
    memset(dp,-1,sizeof(dp));
    scanf("%d",&n);
    for(int i=1;i<=n;i++)cin>>s[i];
    for(int i=n+1;i<=2*n;i++)s[i]=s[i-n];\\复制
    n=n*2;
    pre();\\预处理
    scanf("%n%c",&tou);\\这里得加个"\\n",因为数字读入之后后面还有个换行符
    for(int i=1;i<=n;i++){\\特别处理开头单词的情况
        if(s[i][0]==tou){
            dp[1<<(i-1)][i]=s[i].size();
            ans=max(ans,dp[1<<(i-1)][i]);\\注意这里也要更新ans(我被卡过一次)
        }
    }
    for(int i=1;i<(1<<n);i++){\\开始dp
```




应用 >>




题库




题单



比赛





记录




讨论

```
for(int j=1;j<=n;j++){
    if(i&(1<<(j-1)))continue;
    for(int k=1;k<=n;k++){
        if((i&(1<<(k-1)))&&len[k][j]!=-1&&dp[i][k]!=-1){
            dp[i|(1<<(j-1))][j]=max(dp[i|(1<<(j-1))][j],dp[i][k]+len[k][j]);
            ans=max(ans,dp[i|(1<<(j-1))][j]);
        }
    }
}
printf("%d",ans);
return 0;\\大功告成
}
```

 29



 36 条评论

共 7 页

K

<

1

2

3

4



在洛谷，
享受 Coding 的快乐



关于洛谷 | 帮助
小黑屋 | 陶
Developers

增值电信业务经营许可证
沪ICP备18012868号