数据结构课程设计



班级： 1618403

学号： 161840227

姓名： 韦 鑫

指导教师：孙 涵

目录

1.采用的数据结构 ………………………………………… 3

2.算法设计思想 …………………………………………… 3

3.关键代码 ………………………………………………… 3

4.测试数据和结果 ………………………………………… 6

5.算法的时间复杂度及其改进方法 ……………………… 7

6.结束语 …………………………………………………… 7

1. 采用的数据结构

//URL匹配规则的结构

struct Rule

{

vector<string>p; //URL匹配的规则

char name[64]; //URL匹配的名字

bool flag; //标记该规则最后是否有"/"

}a[maxn];

使用vector存储URL规则的各个部分

二、算法设计思想

①分割并存储URL匹配规则：通过调用strtok（）函数将输入的匹配规则字符串以”/”为分隔符分割成多个部分以便匹配。

②分割并判断URL地址：同样通过调用strtok（）函数将输入的地址分割成多个部分，与n条规则一一匹配。

③特别需要注意的就是URL匹配规则与URL地址在最后是否有“/“，若有，需要在结构题中使用flag = 1进行标记，否则会出错。

三、关键代码

//URL匹配规则的结构

struct Rule

{

vector<string>p; //URL匹配的规则

char name[64]; //URL匹配的名字

bool flag; //标记该规则最后是否有"/"

}a[maxn];

//字符串分割，将字符串s以"/"为分隔符分割，并存放于向量v中

void split(vector<string> &v, char s[])

{

char \*temp = strtok(s, "/");

while (temp)

{

v.push\_back(temp);

temp = strtok(NULL, "/");

}

}

//判断字符串s是否都是数字

bool isNum(string s)

{

for (int i = 0; i < s.length(); i++)

if (!(s[i] >= '0' && s[i] <= '9')) return false;

return true;

}

int n, m;//规则和查询的数量

//处理URL地址，flag标记此URL地址最后是否有"/"

void solve(vector<string>URL, bool flag)

{

int i;

vector<string>ans;//存放参数

//顺序遍历n条规则

for (i = 0; i < n; i++)

{

ans.clear();//先清空

int j = 0, k = 0;

vector<string>t = a[i].p;//第i条规则

//查看URL是否和此规则匹配

while (j < t.size() && k < URL.size())

{

//情况1: <int> 需要考虑所输入的URL地址第k部分是否为数字，若是需要考虑消除前导零

if (t[j] == "<int>")

{

if (isNum(URL[k]))//如果都是数字

{

int sub;

for (sub = 0; sub < URL[k].size() - 1 && URL[k][sub] == '0'; sub++);//消除前导零

ans.push\_back(URL[k].substr(sub));//记录

j++, k++;//匹配下一部分

continue;

}

}

//情况2: <str> 因为题设保证了字符串的规范，所以直接记录，向下继续进行匹配

else if (t[j] == "<str>")

{

ans.push\_back(URL[k]);//直接记录即可

j++, k++;

continue;

}

//情况3: <path> 因为预处理时将地址以"/"为分隔符分隔，所以在这里需要拼接起来且去掉第一个"/"

else if (t[j] == "<path>")

{

string s;

while (k < URL.size()) s = s + "/" + URL[k++];

ans.push\_back(s.substr(1));//去掉第一个"/"，记录

j++;//实际上k已经到最后了，不需要在此自增也会跳出循环

continue;

}

//情况4: 匹配 URL匹配规则中的固定字符串

else if (t[j] == URL[k])

{

j++, k++;

continue;

}

//若出现了以上所述的其他情况，说明URL地址非法，直接跳出循环

break;

}

//若匹配成功就跳出(注意URL匹配规则最后和URL地址最后是否都有"/")

if (j == t.size() && k == URL.size() && flag == a[i].flag) break;

}

//URL与所有匹配规则失配,输出"404"

if (i == n) printf("404\n");

else

{

printf("%s", a[i].name);//输出匹配的规则的名字

for (int w = 0; w < ans.size(); w++)//输出各个参数

{

cout << " " << ans[w];

}

printf("\n");

}

}

int main()

{

scanf("%d%d", &n, &m);

string rule;

//输入URL匹配规则

for (int i = 0; i < n; i++)

{

char temp[128];

scanf("%s", temp); //URL匹配规则

scanf("%s", a[i].name); //匹配规则的名字

//记录规则的最后是否有"/"

if (temp[strlen(temp) - 1] == '/') a[i].flag = 1;

else a[i].flag = 0;

split(a[i].p, temp);//分割，让匹配规则以"/"为分隔符分割成多个部分以便匹配

}

for (int i = 0; i < m; i++)

{

vector<string>URL; //用向量装载URL地址的各个符分

char temp[128]; //输入URL地址

scanf("%s", temp);

//记录规则的最后是否有"/"

bool flag = 0;

if (temp[strlen(temp) - 1] == '/') flag = 1;

split(URL, temp);//分割

solve(URL, flag);//判断处理

}

system("pause");

return 0;

}

四、测试数据和结果

样例输入

5 4

/articles/2003/ special\_case\_2003

/articles/<int>/ year\_archive

/articles/<int>/<int>/ month\_archive

/articles/<int>/<int>/<str>/ article\_detail

/static/<path> static\_serve

/articles/2004/

/articles/1985/09/aloha/

/articles/hello/

/static/js/jquery.js

样例输出

year\_archive 2004

article\_detail 1985 9 aloha

404

static\_serve js/jquery.js



(输入一个地址便进行一次判断，所以输入输出是间隔着的)

五、算法的时间复杂度即改进方法

嵌套了两层循环，时间复杂度与匹配规则条数n、匹配规则、地址长度有关。假设最坏情况最大长度为M，则最坏时间复杂度为O(n\*M)

六、结束语

代码共139行

在本题实验中我学习使用了strtok（）以具体分隔符对字符串进行分割的操作。并学习了STL库中vector向量的具体操作,把重心更多的放在了如何解题而不是在存储上，进一步让我体会到函数封装的重要性。

本题为CSP模拟题，难度较大，需要足够的耐心以及对问题的抽象，我在今后会更多的练习该类型的题目，争取能下一次CSP认证中取得300分突破本次的220分