2016级数据结构第五次上机解题报告

A: ModricWang的瑞士轮

这道题介绍了竞技比赛中的一种赛制，大体思想是先建立排名，然后每一轮比赛都在排名相邻的两名选手间完成，通过一定轮数的比赛得到最终的排名。实现时只需用数组存储每个排名对应的选手，每一轮对相邻排名的选手的实力值进行对比即可。

参考代码：

#include <cstdio>

#include <cstdlib>

#include <algorithm>

using namespace std;

int n,r,q;

struct node

{

int score,strength,id;

}a[300000];

int cmp(node x,node y)

{

if(x.score!=y.score)

return x.score>y.score;

return x.id<y.id;

}

int main()

{

scanf("%d%d%d",&n,&r,&q);

for(int i=1;i<=2\*n;i++)

a[i].id=i;

for(int i=1;i<=2\*n;i++)

scanf("%d",&a[i].score);

for(int i=1;i<=2\*n;i++)

scanf("%d",&a[i].strength);

sort(a+1,a+2\*n+1,cmp);

while(r)

{

for(int i=1;i<=n;i++)

if(a[i\*2-1].strength>a[i\*2].strength)

a[i\*2-1].score++;

else

a[i\*2].score++;

sort(a+1,a+2\*n+1,cmp);

r--;

}

printf("%d",a[q].id);

return 0;

}

B ModricWang's Marshaling

通过进栈顺序和出栈顺序求出操作顺序，初始时栈为空，根据出栈顺序逐个进行推理，如果下一个出栈的元素在栈顶，就让它出栈，否则不断进栈，直到当前元素在栈内为止。

参考代码：

#include <iostream>

#include <stack>

using namespace std;

int n, now, tot;

stack<int> s;

void work() {

tot = 0;

while (!s.empty()) s.pop();

cin >> n;

while (n--) {

cin >> now;

while (tot < now) s.push(++tot), cout << 'A';

while (s.top() != now) s.pop(), cout << 'B';

s.pop(), cout << 'B';

}

cout << '\n';

}

int main() {

ios::sync\_with\_stdio(false);

cin.tie(0);

cout.tie(0);

work();

return 0;

}

C:Mdd的回文串

这道题要求判断一个字符串是否为回文串，考察的是对字符串的一些简单操作，由于加入了通配符，所以需要自己建立一个匹配函数。

参考代码：

#include <iostream>

#include <string>

#include <fstream>

#define in std::cin

#define out std::cout

using namespace std;

bool match(char p, char q) {

if (p == '?' or q == '?') {

return true;

}

return p == q;

}

bool isPalindrome(string str) {

int s = 0, e = str.length() - 1;

while (s < e) {

if (!match(str[s++], str[e--])) {

return false;

}

}

return true;

}

int main() {

string str;

while (in >> str) {

out << (isPalindrome(str)? "Yes\n" : "No\n");

}

return 0;

}

D：括号匹配

这是一道非常经典的题，方法是利用栈来判断给的括号是否匹配。如果一个当前字符是左括号，就将这个字符进栈。如果是右括号，就需要检查是否能够形成一个完整的括号，如果当前栈顶的元素和这个右括号匹配，就说明形成了一个完整的括号，然后将栈顶元素出栈，否则说明有一个括号不能够匹配。

参考代码：

#include <iostream>

#include <string>

#include <fstream>

#define in std::cin

#define out std::cout

using namespace std;

const int MAX\_SIZE = 1000;

struct MyStack {

int \_top;

int \_size;

int \*\_data;

MyStack(int size) {

\_top = -1;

\_size = size;

\_data = new int[\_size];

}

void init() {

\_top = -1;

}

bool push(int x) {

if (\_top == \_size - 1) {

return false;

}

\_data[++\_top] = x;

return true;

}

void pop() {

\_top--;

}

int top() {

return \_data[\_top];

}

bool empty() {

return \_top == -1;

}

~MyStack() {

delete \_data;

}

};

bool match(MyStack &stack, char p) {

bool ret = false;

switch (p) {

case ')':

if (stack.top() == '(')

ret = true;

break;

case ']':

if (stack.top() == '[')

ret = true;

break;

case '}':

if (stack.top() == '{')

ret = true;

break;

case '>':

if (stack.top() == '<')

ret = true;

break;

default:

break;

}

return ret;

}

int main() {

string input;

MyStack stack(MAX\_SIZE);

while (in >> input) {

stack.init();

int len = input.length();

bool ok = true;

for (int i = 0; i < len and ok; ++i) {

char p = input[i];

switch (p) {

case ')':

case ']':

case '}':

case '>':

if (stack.empty() or !match(stack, p)) {

ok = false;

}

stack.pop();

break;

default:

stack.push(p);

}

}

if (!stack.empty()) {

ok = false;

}

out << (ok ? "Yes\n" : "No\n");

}

return 0;

}

E:Mdd学数学

这道题用到了括号匹配的知识，同时又加入了一些限制条件:大括号只能包含大括号和中括号，中括号只能够包含小括号，小括号里面不能有任何括号。我们可以采取与括号匹配同样的方法，只不过在括号进栈与出栈时，需要加上一些判断条件。

#include <iostream>

#include <string>

#include <fstream>

#define in std::cin

#define out std::cout

using namespace std;

const int MAX\_SIZE = 1000;

struct MyStack {

int \_top;

int \_size;

int \*\_data;

MyStack(int size) {

\_top = -1;

\_size = size;

\_data = new int[\_size];

}

void init() {

\_top = -1;

}

bool push(int x) {

if (\_top == \_size - 1) {

return false;

}

\_data[++\_top] = x;

return true;

}

void pop() {

\_top--;

}

int top() {

return \_data[\_top];

}

bool empty() {

return \_top == -1;

}

~MyStack() {

delete \_data;

}

};

bool match(MyStack &stack, char p) {

if (stack.empty()) {

return false;

}

bool ret = false;

switch (p) {

case ')':

if (stack.top() == '(')

ret = true;

break;

case ']':

if (stack.top() == '[')

ret = true;

break;

case '}':

if (stack.top() == '{')

ret = true;

break;

default:

break;

}

return ret;

}

int main() {

string str;

MyStack stack(MAX\_SIZE);

while (in >> str) {

stack.init();

//round, square, curly分别代表栈内是否有小括号，中括号，大括号

bool round, square, curly, ok = true;

round = square = curly = false;

int len = str.length();

for (int i = 0; i < len and ok; ++i) {

char p = str[i];

switch (p) {

case '(':

if (round or curly) {

ok = false;

}

round = true;

stack.push(p);

break;

case ')':

if (!match(stack, p)) {

ok = false;

}

stack.pop();

round = false;

break;

case '[':

if (round or square) {

ok = false;

}

square = true;

stack.push(p);

break;

case ']':

if (!match(stack, p)) {

ok = false;

}

stack.pop();

square = false;

break;

case '{':

if (round or square) {

ok = false;

}

curly = true;

stack.push(p);

break;

case '}':

if (!match(stack, p)) {

ok = false;

}

stack.pop();

curly = false;

break;

default:

break;

}

}

if (!stack.empty()) {

ok = false;

}

out << (ok ? "Yes\n" : "No\n");

}

return 0;

}

F: DH的新手机Ⅱ

采取队列的思想，一个队列记录所有的信息，另使用一个数组记录每个app读到了第几个信息。随着操作更改未读信息的数量。

参考代码：

#include <cstdio>

#include <vector>

using namespace std;

int app[300001],endd[300001];

vector<int> message;

int visited[300001];

int main()

{

int n,q,type,x,ans,e;

while(scanf("%d%d",&n,&q)!=EOF)

{

ans=0;

e=0;

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

{

scanf("%d%d",&type,&x);

if(type==1)

{

app[x]++;

endd[x]=message.size()+1;

message.push\_back(x);

ans++;

}

else if(type==2)

{

ans-=app[x];

visited[x]=endd[x];

app[x]=0;

}

else

{

for(int j=e;j<x;j++)

if(visited[message[j]]<j+1)

{

visited[message[j]]=j+1;

app[message[j]]--;

ans--;

}

if(e<x)

e=x;

}

printf("%d\n",ans);

}

for(int i=1;i<=n;i++)

app[i]=0;

for(int i=1;i<=n;i++)

endd[i]=0;

for(int i=1;i<=n;i++)

visited[i]=0;

message.clear();

}

return 0;

}

G: ModricWang的星灵棋

此题属于NP问题，需要用枚举的方法解决。由于是求出最少步骤数，适合使用宽度优先搜索，用队列作为存储状态的数据结构。对于每一种状态，将它的下一步可能的状态存入队列中。对于已经进入过队列的状态，需要予以剔除，否则会超时。

#include<cstdio>

#include<queue>

#include<map>

using namespace std;

struct data {

char ch[4][4];

bool operator<(const data &a) const {

for (int i = 0; i < 4; i++) {

for (int j = 0; j < 4; j++) {

if (ch[i][j] != a.ch[i][j])return ch[i][j] < a.ch[i][j];

}

}

return ch[3][3] < a.ch[3][3];

}

bool operator==(const data &a) const {

for (int i = 0; i < 4; i++) {

for (int j = 0; j < 4; j++) {

if (ch[i][j] != a.ch[i][j])return false;

}

}

return true;

}

};

bool is(const char ch[4][4]) {

for (int i = 0; i < 4; i++) {

int j;

for (j = 1; j < 4; j++) {

if (ch[i][j] != ch[i][0])break;

}

if (j == 4)return true;

for (j = 1; j < 4; j++) {

if (ch[j][i] != ch[0][i])break;

}

if (j == 4)return true;

}

int i;

for (i = 1; i < 4; i++) {

if (ch[i][i] != ch[0][0])break;

}

if (i == 4)return true;

for (i = 1; i < 4; i++) {

if (ch[i][3 - i] != ch[0][3])break;

}

if (i == 4)return true;

return false;

}

char ch[4][5];

const int dir[4][2] = {{-1, 0},

{0, 1},

{1, 0},

{0, -1}};

queue<int> step;

queue<data> que;

map<data, bool> mp;

queue<char> color;

map<char, char> rev;

int bfs() {

while (!que.empty())que.pop();

while (!step.empty())step.pop();

while (!color.empty())color.pop();

data p;

for (int i = 0; i < 4; i++) {

for (int j = 0; j < 4; j++) {

p.ch[i][j] = ch[i][j];

}

}

que.push(p);

step.push(0);

color.push('B');

que.push(p);

step.push(0);

color.push('W');

mp[p] = true;

while (!que.empty()) {

data d = que.front();

que.pop();

int stp = step.front();

step.pop();

if (is(d.ch)) {

return stp;

}

char cl = color.front();

color.pop();

char cc[4][4];

for (int i = 0; i < 4; i++) {

for (int j = 0; j < 4; j++) {

if (d.ch[i][j] == 'O') {

for (int k = 0; k < 4; k++) {

int xx = i + dir[k][0], yy = j + dir[k][1];

if (xx < 0 || xx > 3 || yy < 0 || yy > 3)continue;

if (d.ch[xx][yy] == cl) {

d.ch[i][j] = cl;

d.ch[xx][yy] = 'O';

if (!mp[d]) {

que.push(d);

step.push(stp + 1);

color.push(rev[cl]);

mp[d] = true;

}

d.ch[i][j] = 'O';

d.ch[xx][yy] = cl;

}

}

}

}

}

}

return -1;

}

int main() {

rev.clear();

rev['W'] = 'B';

rev['B'] = 'W';

mp.clear();

for (int i = 0; i < 4; i++) {

scanf("%s", ch[i]);

}

int ans = bfs();

printf("%d\n", ans);

return 0;

}