2016级数据结构第八次上机解题报告

A

建树，递归判断即可，详情请见代码。

参考代码：

/\*

Author: 公绪民(59)

Result: AC Submission\_id: 284144

Created at: Wed May 03 2017 13:35:46 GMT+0800 (CST)

Problem: 762 Time: 10 Memory: 2644

\*/

#include <iostream>

#include <string>

#include <cstdio>

#define MaxSize 11000

using namespace std;

typedef string ElemType;

int ans = 0;

typedef struct node

{

ElemType data;

struct node \*lchild;

struct node \*rchild;

}Tree;

bool JudgeMirror1(Tree \*left, Tree \*right)

{

if (left != NULL&&right != NULL)

if (JudgeMirror1(left->lchild, right->rchild) && JudgeMirror1(left->rchild, right->lchild))

if (left->data == right->data)

return true;

else

return false;

else

return false;

else if (left == NULL&&right == NULL)

return true;

else

return false;

}

bool JudgeMirror2(Tree \*left, Tree \*right)

{

if (left != NULL&&right != NULL)

if (JudgeMirror2(left->lchild, right->lchild) && JudgeMirror2(left->rchild, right->rchild))

return true;

else

return false;

else if (left == NULL&&right == NULL)

return true;

else

return false;

}

void Create(Tree\* &b, string str)

{

Tree \*St[MaxSize], \*p;

p = NULL;

int top, k, j;

top = -1;

j = 0;

char ch;

b = NULL;

ch = str[j];

while (ch != '\0')

{

switch (ch)

{

case '(':

top++;

St[top] = p;

k = 1;

break;

case ')':

top--;

break;

case ',':

k = 2;

break;

default:

p = new Tree;

int cur = 0;

while (str[j + 1] != '('&&str[j + 1] != ')'&&str[j + 1] != ','&&str[j + 1] != '\0')

p->data += str[j++];

p->data += str[j];

p->lchild = p->rchild = NULL;

if (b == NULL)

b = p;

else

{

switch (k)

{

case 1:

St[top]->lchild = p;

break;

case 2:

St[top]->rchild = p;

break;

}

}

}

j++;

ch = str[j];

}

}

int main()

{

int n;

cin >> n;

while (n--)

{

string s;

while (cin >> s)

{

Tree \*b;

b = NULL;

Create(b, s);

if (JudgeMirror1(b->lchild, b->rchild) && JudgeMirror2(b->lchild, b->rchild))

cout << "GzhIsSoSoHandsome" << endl;

else if (JudgeMirror1(b->lchild, b->rchild))

cout << "GzhIsHandsome" << endl;

else if (JudgeMirror2(b->lchild, b->rchild))

cout << "GzhIsSoHandsome" << endl;

else

cout << "GzhIsSoSoSoHandsome" << endl;

}

}

return 0;

}

B

这道题的考点是哈夫曼编码，然后输出编码之后的长度，一种方法是按照书上的方法建立哈夫曼树，然后计算编码长度。不过还有一种简便的方法，我们可以维护一个堆，然后每次从堆中取出两个权值最小的节点，计算他们的和，并放入堆中，这样直到堆中只有一个节点的时候，我们便得到了答案。

参考代码：

#include<cstdio>

#include <queue>

int main() {

int c, ans = 0;

int weight[128] = {0};

std::priority\_queue<int, std::vector<int>, std::greater<int>> q;

while ((c = getchar()) != EOF) {

weight[c]++;

}

for (int i = 0; i < 128; ++i) {

if (weight[i]) {

q.push(weight[i]);

}

}

while (q.size() > 1) {

int x1 = q.top();

q.pop();

int x2 = q.top();

q.pop();

ans += x1 + x2;

q.push(x1 + x2);

}

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

}

C

比较容易发现将从上往下算同一级的人分在一组是最优的，所以最后的答案就是最深的一颗树的深度，那么dfs一下就好了

参考代码：

#include <cstdio>

int father[2001];

int vis[2001];

int ans;

void dfs(int x,int h)

{

vis[x]=1;

if(father[x]==-1)

{

if(h>ans)

ans=h;

return;

}

return dfs(father[x],h+1);

}

int main()

{

int n;

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

{

ans=0;

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

scanf("%d",&father[i]);

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

if(!vis[i])

dfs(i,1);

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

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

vis[i]=0;

}

return 0;

}

D

我们可以记

G(n)为长度为n的二叉搜索树的数目

F(i, n), 1 <= i <= n, 根节点为i, 长度为n的二叉搜索树的数目

显然 G(n) = F(1, n) + F(2, n) + ... + F(n, n), G(0) = 1, G(1) = 1

而F(i, n) = G(i - 1) \* G(n - i), 也就是根节点为i时，左子树的长度为i – 1, 而右子树的长度即为n – i, 根据我们刚刚的定义便可以推出

G(n) = G(0) \* G(n-1) + G(1) \* G(n-2) + … + G(n-1) \* G(0)

参考代码：

#include <iostream>

#include <cstring>

using namespace std;

int main() {

int g[16] = {0};

g[0] = g[1] = 1;

for (int i = 2; i <= 16; ++i) {

for (int j = 1; j <= i; ++j) {

g[i] += g[j - 1] \* g[i - j];

}

}

int x;

while (cin >> x) {

cout << g[x] << "\n";

}

}

E

二分，详情请见代码。

参考代码：

#include <iostream>

#include <cstdio>

#include <algorithm>

using namespace std;

int L, N, M;

int d[80000];

bool cmp(int a, int b)

{

return a < b;

}

int BSearch(int l, int h, int k)

{

int m, last, cnt;

while (l <= h)

{

m = (l + h) >> 1;

last = cnt = 0;

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

{

if (m >= d[i] - d[last]) cnt++;

else last = i;

}

if (cnt > k)

h = m - 1;

else

l = m + 1;

}

return l;

}

int main()

{

while (scanf("%d%d%d", &L, &N, &M) != EOF)

{

d[0] = 0;

d[N + 1] = L;

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

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

sort(d + 1, d + 1 + N, cmp);

printf("%d\n", BSearch(0, L, M));

}

return 0;

}

F

dfs时记录一下连续经过权值为1的数量，如果超过就return，否则继续往下，到叶节点计数

参考代码：

#include <cstdio>

#include <vector>

using namespace std;

vector<int> mp[100001];

int a[100001];

int ans;

int n,m;

void dfs(int x,int cnt,int from)

{

if(a[x]==0)

cnt=0;

else cnt++;

if(cnt>m)

return;

if(mp[x].size()==1&&mp[x][0]==from)

{

ans++;

return;

}

for(int i=0;i<mp[x].size();i++)

if(mp[x][i]!=from)

dfs(mp[x][i],cnt,x);

return;

}

int main()

{

int x,y;

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

{

ans=0;

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

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

for(int i=0;i<n-1;i++)

{

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

mp[x].push\_back(y);

mp[y].push\_back(x);

}

dfs(1,0,0);

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

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

mp[i].clear();

}

return 0;

}