2.38

#include<iostream>

#include<vector>

using namespace std;

const int maxlen = 5;

vector<int> minroad;

vector<int> maxroad;

vector<int> flag;

struct node

{

node \* next[maxlen];

int road[maxlen];

int num;

};

void init(node \*& x)

{

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

{

x->next[n] = NULL;

x->road[n] = 0;

}

}

void dfs(node \* x, int targal, int path)

{

if (x->num == (targal + 1))

{

if (path < minroad[targal])

{

minroad[targal] = path;

}

if (path > maxroad[targal])

{

maxroad[targal] = path;

}

return;

}

flag[x->num - 1] = 1;

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

{

if (x->next[n] != NULL)

{

if (flag[x->next[n]->num - 1] == 0)

{

dfs(x->next[n], targal, path + x->road[n]);

}

}

}

flag[x->num - 1] = 0;

return;

}

int main()

{

int n;

cin >> n;

vector<node\*> map;

for (int n1 = 0; n1 < n; n1++)

{

node \*mid = new(node);

init(mid);

mid->num = n1 + 1;

map.push\_back(mid);

flag.push\_back(0);

}

for (int n1 = 0; n1 < n; n1++)

{

int mid;

cin >> mid;

for (int n2 = 0; n2 < mid; n2++)

{

int midn, midr;

cin >> midn >> midr;

map[n1]->next[n2] = map[midn - 1];

map[n1]->road[n2] = midr;

}

}

for (int n1 = 0; n1 < n; n1++)

{

minroad.push\_back(999);

maxroad.push\_back(0);

}

for (int n1 = 1; n1 < n; n1++)

{

dfs(map[0], n1, 0);

}

for (int n1 = 1; n1 < n; n1++)

{

cout << n1 + 1 << " 最早开始时间：" << minroad[n1] << " 最晚开始时间：" << maxroad[n1] << "\n";

}

cout << "最早完成时间：" << minroad[n - 1];

system("pause");

return 0;

}

/\*

输入nr

10

2 2 5 3 6

1 4 3

2 4 6 5 3

3 6 4 7 4 5 7

2 7 1 8 4

1 10 4

1 9 5

1 9 2

1 10 2

0

输出

2 最早开始时间：5 最晚开始时间：5

3 最早开始时间：6 最晚开始时间：6

4 最早开始时间：8 最晚开始时间：12

5 最早开始时间：9 最晚开始时间：19

6 最早开始时间：12 最晚开始时间：16

7 最早开始时间：10 最晚开始时间：20

8 最早开始时间：13 最晚开始时间：23

9 最早开始时间：15 最晚开始时间：25

10 最早开始时间：16 最晚开始时间：27

最早完成时间：16

\*/

2.36

#include<iostream>

#include<vector>

using namespace std;

const int maxlen = 5;

vector<int> minroad;

vector<int> flag;

struct node

{

node \* next[maxlen];

int road[maxlen];

int num;

};

void init(node \*& x)

{

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

{

x->next[n] = NULL;

x->road[n] = 0;

}

}

void dfs(node \* x, int targal,int path)

{

if (x->num == (targal + 1))

{

if (path < minroad[targal])

{

minroad[targal] = path;

}

return;

}

flag[x->num - 1] = 1;

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

{

if (x->next[n] != NULL)

{

if (flag[x->next[n]->num - 1] == 0)

{

dfs(x->next[n],targal, path + x->road[n]);

}

}

}

flag[x->num - 1] = 0;

return;

}

int main()

{

int n;

cin >> n;

vector<node\*> map;

for (int n1 = 0; n1 < n; n1++)

{

node \*mid = new(node);

init(mid);

mid->num = n1 + 1;

map.push\_back(mid);

flag.push\_back(0);

}

for (int n1 = 0; n1 < n; n1++)

{

int mid;

cin >> mid;

for (int n2 = 0; n2 < mid; n2++)

{

int midn, midr;

cin >> midn >> midr;

map[n1]->next[n2] = map[midn - 1];

map[n1]->road[n2] = midr;

}

}

for (int n1 = 0; n1 < n; n1++)

{

minroad.push\_back(999);

}

for(int n1=1;n1<n;n1++)

{

dfs(map[0], n1,0);

}

for (int n1 = 1; n1 < n; n1++)

{

cout << "到" << n1 + 1 << "最小路径" << minroad[n1] << "\n";

}

system("pause");

return 0;

}

/\*

输入

8

2 2 2 3 3

2 1 2 4 2

2 1 3 4 1

4 2 2 5 2 6 4 3 1

3 7 2 6 1 4 2

4 4 4 5 1 7 2 8 1

3 5 2 6 2 8 3

2 6 1 7 3

输出

2-8 为 b-h

到2最小路径2

到3最小路径3

到4最小路径4

到5最小路径6

到6最小路径7

到7最小路径8

到8最小路径8

\*/

2.35

#include<vector>

#include<iostream>

using namespace std;

const int maxlen = 5;

struct node

{

int in, out;

int num;

node \* next[maxlen];

};

void init(node \* &x)

{

x->in = 0;

x->out = 0;

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

{

x->next[n] = NULL;

}

return;

}

int main()

{

vector<node \*> map;

int n;

cin >> n;

for (int n1 = 0; n1 < n; n1++)

{

node \* mid = new(node);

init(mid);

mid->num = n1 + 1;

map.push\_back(mid);

}

for (int n1 = 0; n1 < n; n1++)

{

int mid;

cin >> mid;

for (int n2 = 0; n2 < mid; n2++)

{

int mid1;

cin >> mid1;

map[n1]->next[n2] = map[mid1 - 1];

}

}

for (int n1 = 0; n1 < n; n1++)

{

for (int n2 = 0;n2<maxlen; n2++)

{

if (map[n1]->next[n2] != NULL)

{

map[n1]->next[n2]->in += 1;

map[n1]->out += 1;

}

}

}

for (int n1 = 0; n1 < n; n1++)

{

cout <<n1+1<< " 入度：" << map[n1]->in << " 出度：" << map[n1]->out << "\n";

}

system("pause");

return 0;

}

/\*

输入

6

0

2 1 4

2 2 6

2 5 3

1 1

4 1 2 5 4

输出

1 入度：3 出度：0

2 入度：2 出度：2

3 入度：1 出度：2

4 入度：2 出度：2

5 入度：2 出度：1

6 入度：1 出度：4

\*/

2.34

#include<iostream>

#include<vector>

#include<queue>

using namespace std;

const int maxlen = 5;

vector<int> flag;

struct node

{

int num;

node \* next[maxlen];

};

void init(node \* & x)

{

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

{

x->next[n] = NULL;

}

return;

}

void DFS(node \* x)

{

if (flag[x->num - 1] == 0)

{

cout << x->num << " ";

flag[x->num - 1] = 1;

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

{

if (x->next[n] != NULL)

{

if (flag[x->next[n]->num - 1] == 0)

{

DFS(x->next[n]);

}

}

}

return;

}

}

void BFS(node \* x)

{

queue<node\*> bfs;

bfs.push(x);

flag[x->num - 1] = 0;

while (!bfs.empty())

{

node \* mid = bfs.front();

bfs.pop();

cout << mid->num <<" ";

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

{

if (mid->next[n] != NULL)

{

if (flag[mid->next[n]->num - 1] == 1)

{

flag[mid->next[n]->num - 1] = 0;

bfs.push(mid->next[n]);

}

}

}

}

return;

}

int main()

{

int n;

cin >> n;

vector<node \*> map;

for (int n1 = 0; n1 < n; n1++)

{

node \* mid = new (node);

init(mid);

mid->num = n1 + 1;

map.push\_back(mid);

flag.push\_back(0);

}

for (int n1 = 0; n1 < n; n1++)

{

int n2;

cin >> n2;

for (int n3 = 0; n3 < n2; n3++)

{

int mid;

cin >> mid;

map[n1]->next[n3] = map[mid - 1];

}

}

DFS(map[0]);//v1=1

cout << "\n";

BFS(map[0]);//v1=1

system("pause");

return 0;

}

/\*

输入

20

3 2 8 5

3 1 10 3

3 2 12 4

3 3 5 14

3 1 6 4

3 5 7 15

3 6 17 8

3 7 9 1

3 10 8 18

3 11 2 9

3 12 19 10

3 11 13 3

3 12 20 14

3 13 15 4

3 16 14 6

3 20 17 15

3 18 16 7

3 9 19 17

3 11 20 18

3 19 13 16

输出

dfs 1 2 10 11 12 13 20 19 18 9 8 7 6 5 4 3 14 15 16 17

bfs 1 2 8 5 10 3 7 9 6 4 11 12 17 18 15 14 19 13 16 20

\*/

2.33

#include<iostream>

#include<vector>

#include<algorithm>

#include<queue>

using namespace std;

struct node

{

node \* left,\* right;

int weight;

int p;

};

int n\_weight(node \* x,int y)

{

if (x->left == NULL && x->right == NULL)

{

return y \* x->weight;

}

return n\_weight(x->right, y + 1)+ n\_weight(x->left, y + 1);

}

bool cmp(node \* a, node \* b)

{

if (a->p == b->p)

{

return a->weight< b->weight;

}

else

{

return a->p < b->p;

}

}

void levelorder(node \* x)

{

queue<node \*> level;

level.push(x);

int n = level.size();

while (!level.empty())

{

node \* mid = level.front();

level.pop();

cout << mid->weight << " ";

n--;

if (mid->left != NULL)

{

level.push(mid->left);

}

if (mid->right!= NULL)

{

level.push(mid->right);

}

if (n == 0)

{

cout << "\n";

n = level.size();

}

}

return;

}

int main()

{

int n;

cin >> n;

vector<node \*> hfm;

for (int n1 = 0; n1 < n; n1++)

{

int mid;

cin >> mid;

node \* mid2 = new(node);

mid2->left = NULL;

mid2->right = NULL;

mid2->weight = mid;

mid2->p = 0;

hfm.push\_back(mid2);

}

for (int n1 = 0; n1 < n-1; n1++)

{

sort(hfm.begin(), hfm.end(), cmp);

hfm[0]->p = 1;

hfm[1]->p = 1;

node \* mid = new(node);

mid->left = hfm[0];

mid->right = hfm[1];

mid->p = 0;

mid->weight = n\_weight(mid,1);

hfm.push\_back(mid);

}

for (int n1 = 0; n1 < hfm.size(); n1++)

{

if (hfm[n1]->p == 0)

{

levelorder(hfm[n1]);

break;

}

}

system("pause");

return 0;

}

/\*

输入

7

8 2 5 3 2 17 4

输出

147

33 73

8 14 17 26

2 2 3 4 5 8

\*/

2.31

#include<vector>

#include<algorithm>

#include<iostream>

#include<queue>

using namespace std;

const int n = 10;//node数

vector<int> init\_val;

struct node

{

node \* left;

node \* right;

int val;

};

vector<int> preorder, inorder, postorder,levelorder;

int n1, n2;//树order长度

int index = 0;

int leave\_count = 0;//叶子节点个数

int maxdeep = 0;//最大深度

void order\_c(node \* root,int deep)///带计算各参数的order

{

if (root == NULL)

{

return;

}

preorder.push\_back(root->val);

order\_c(root->left,deep);

inorder.push\_back(root->val);

order\_c(root->right,deep);

postorder.push\_back(root->val);

if (root->right == NULL && root->left == NULL)

{

leave\_count += 1;

}

}

void order(node \* root)///不带计算各参数的order

{

if (root == NULL)

{

return;

}

preorder.push\_back(root->val);

order(root->left);

inorder.push\_back(root->val);

order(root->right);

postorder.push\_back(root->val);

}

void bfs(node \* root)

{

cout << root ->val << "\n";

queue<node \*> mid2;

mid2.push(root);

int n;

while (!mid2.empty())

{

n = mid2.size();

while (n--)

{

node \* mid3 = mid2.front();

mid2.pop();

if (mid3->left != NULL)

{

levelorder.push\_back(mid3->left->val);

cout << mid3->left->val<<" ";

mid2.push(mid3->left);

}

if (mid3->right != NULL)

{

levelorder.push\_back(mid3->right->val);

cout << mid3->right->val<<" ";

mid2.push(mid3->right);

}

}

cout << "\n";

maxdeep += 1;

}

}

void copy(node \* & root\_c, node \* root)

{

if (root == NULL)

{

root\_c = NULL;

return;

}

root\_c = new (node);

root\_c->val = root->val;

copy(root\_c->left, root->left);

copy(root\_c->right, root->right);

}

bool j\_eq()

{

if (n1 != n2)

{

return false;

}

for (int n3 = 0; n3 < n1; n3++)

{

if (inorder[n3] != inorder[n3 + n1])

{

return false;

}

if (postorder[n3] != postorder[n3 + n1])

{

return false;

}

if (preorder[n3] != preorder[n3 + n1])

{

return false;

}

}

return true;

}

void build(node \* & root, int n1)

{

if (n1 > n)

{

root = NULL;

return;

}

root = new(node);

build(root->left, n1 \* 2);

root->val = init\_val[index++];

build(root->right, n1 \* 2 + 1);

}

int main()

{

for (int n1 = 0; n1 < n; n1++)

{

int mid;

cin >> mid;

init\_val.push\_back(mid);

}

sort(init\_val.begin(), init\_val.end());//建bst

node \* root;

build(root, 1);

node \* root\_copy;

copy(root\_copy, root);

order\_c(root,1);

n1 = inorder.size();

order(root\_copy);

n2 = inorder.size() - n1;

if (j\_eq())

{

cout << "相等" << "\n";

}

else

{

cout << "不相等" << "\n";

}

bfs(root);

cout << "叶子数目：" << leave\_count << "\n" << "最大深度：" << maxdeep << "\n";

for (const int a : inorder)

{

cout << a << " ";

}

system("pause");

return 0;

}

/\*

输入

10

1 5 4 8 7 9 0 2 6 3

输出

10

1 5 4 8 7 9 0 2 6 3

相等

7

4 9

1 6 8 10

0 2 5

叶子数目：5

最大深度：4

0 1 2 4 5 6 7 8 9 10 0 1 2 4 5 6 7 8 9 10

\*/

2.24

#include<vector>

#include<iostream>

using namespace std;

const int m = 10;//数组长度

int main()

{

vector<int> a(m, 0);

int top1 = -1, top2 = a.size();

while (top1 != top2 - 1)

{

int mid;

cin >> mid;

if (mid % 2 == 1)

{

a[top2] = mid;

top2--;

}

else

{

a[top1] = mid;

top1++;

}

}

return 0;

}

2.23

#include<iostream>

#include<stack>

#include<string>

#include<map>

using namespace std;

map<char, int> level;

string in;//\*\*变为~

int n = 0;

bool slt(char a, char b)

{

return level[a] > level[b];

}

int compute()

{

stack<int> ns;

stack<char> os;//os运算符 ns操作数

os.push(';');

for (int n1 = n; n1 < in.size(); n1++)

{

if (in[n1] >= 48 && in[n1] <= 57)

{

ns.push(in[n1] - 48);

}

else

{

if (in[n1] == '(')

{

n = n1 + 1;

ns.push(compute());

n1 = n;

continue;

}

if (in[n1] == ')'&&os.top()==';')

{

n = n1 + 1;

while (os.top() != ';')

{

int mid1 = ns.top();

ns.pop();

int mid2 = ns.top();

ns.pop();

char mid3 = os.top();

os.pop();

switch (mid3)

{

case ';':break;

case'+':ns.push(mid2 + mid1); break;

case'-':ns.push(mid2 - mid1); break;

case'\*':ns.push(mid2 \* mid1); break;

case'/':ns.push(mid2 / mid1); break;

case'`':ns.push(pow(mid2, mid1)); break;

}

}

return ns.top();

}

if (slt(in[n1], os.top()))

{

os.push(in[n1]);

}

else

{

int mid1 = ns.top();

ns.pop();

int mid2 = ns.top();

ns.pop();

char mid3 = os.top();

os.pop();

switch (mid3)

{

case ';':break;

case'+':ns.push(mid2 + mid1); break;

case'-':ns.push(mid2 - mid1); break;

case'\*':ns.push(mid2 \* mid1); break;

case'/':ns.push(mid2 / mid1); break;

case'`':ns.push(pow(mid2, mid1)); break;

}

n1 -= 1;

}

}

}

while (os.top() != ';')

{

int mid1 = ns.top();

ns.pop();

int mid2 = ns.top();

ns.pop();

char mid3 = os.top();

os.pop();

switch (mid3)

{

case ';':break;

case'+':ns.push(mid2 + mid1); break;

case'-':ns.push(mid2 - mid1); break;

case'\*':ns.push(mid2 \* mid1); break;

case'/':ns.push(mid2 / mid1); break;

case'`':ns.push(pow(mid2, mid1)); break;

}

}

return ns.top();

}

int main()

{

level[')'] = 0;

level[';'] = 0;

level['+'] = 1;

level['-'] = 1;

level['\*'] = 2;

level['/'] = 2;

level['`'] = 3;

cin >> in;

cout<<compute();

system("pause");

}

/\*

输入

A\*(B-D)/D+C`(E\*F)

5\*(3-4)/2+2`(3\*2)

输出

62

\*/

2.22

#include<iostream>

#include<vector>

using namespace std;

int main()

{

vector<char> queue;

int n;

cin >> n;

int front = 1, rear = 1;

for (int n1 = 0; n1 < n; n1++)

{

char mid;

cin >> mid;

if (mid == 'r')

{

int midn;

cin >> midn;

for (int n2 = 0; n2 < midn; n2++)

{

char midr;

cin >> midr;

queue.push\_back(midr);

rear += 1;

}

cout <<front << " " << rear<<"\n";

}

else

{

int midn;

cin >> midn;

for (int n2 = 0; n2 < midn; n2++)

{

char midr;

cin >> midr;

front += 1;

}

cout << front << " " << rear << "\n";

}

}

system("pause");

return 0;

}

/\*

输入

5

r 5 d e b g h

c 2 d e

r 5 i j k m l

c 1 b

r 5 n o p q r

\*/

/\*

输出

5

r 5 d e b g h

1 6

c 2 d e

3 6

r 5 i j k m l

3 11

c 1 b

4 11

r 5 n o p q r

4 16

\*/