学习情况表

|  |  |  |  |
| --- | --- | --- | --- |
| **姓名** | 王一博 | **学号** | 2021902655 |
| **学院** | 信息工程学院 | **专业** | 计算机科学与技术 |

（请在下面表格本周学习情况）

|  |
| --- |
| 学习情况简述 |
| 根据网站ACWing和书籍《大话数据结构》进行学习，学习内容包括：堆，栈，队列，哈希表，深度优先搜索（DFS），宽度优先搜索（BFS）等 |
| 本周练习过的代码（例） |
| （本周练习写过的代码，如实填写，疑似抄袭会单独测试）  *代码1：*  *#include<stdio.h>*  *int main()*  *{*  *printf("HelloWorld\n");*  *return 0;*  *}*  *代码2：*  …….. |
| 本周练习过的代码 |

代码1：

#include <iostream>

using namespace std;

const int N = 1E+5 + 10;

int heap[N], n, m, \_size;

void down(int u)

{

int t = u; //t存储当前三项中的最小值，u是根

if (2 \* u <= \_size && heap[2 \* u] < heap[t]) t = 2 \* u;

if (2 \* u + 1 <= \_size && heap[2 \* u + 1] < heap[t]) t = 2 \* u + 1;

if (u != t) //当当前数需要向下移动时

{

swap (heap[t], heap[u]);

down (t); //u==t则说明当前结点已移到合适位置

}

}

int main()

{

cin >> n >> m;

for (int i = 1; i <= n; i ++ ) cin >> heap[i];

\_size = n;

for (int i = n / 2; i >= 1; -- i) down(i); //使输入的数构成小根堆（第n/2个结点是最后一个非叶结点）

while (m -- )

{

cout << heap[1] << " "; //输出根结点，即当前最小的数

//删除最小数

heap[1] = heap[\_size];

\_size --;

down(1);

}

}

代码2：

//因为第4，5个操作，因此需要一个映射关系存储第k个插入的数 在堆中是什么

#include <iostream>

#include <cstring>

using namespace std;

const int N = 1E+5 + 10;

int h[N], hp[N], ph[N], \_size;

void heap\_swap(int a, int b) //交换堆中的两个数，同时交换对应的两个指针指向的区域

{

swap(ph[hp[a]], ph[hp[b]]);

swap(hp[a], hp[b]);

swap(h[a], h[b]);

}

void down(int u)

{

int t = u;

if (2 \* u <= \_size && h[2 \* u] < h[t]) t = 2 \* u;

if (2 \* u + 1 <= \_size && h[2 \* u + 1] < h[t]) t = 2 \* u + 1;

if (t != u)

{

heap\_swap (t, u);

down(t);

}

}

void up(int u)

{

while (u / 2 >= 1 && h[u / 2] > h[u]) // u/2和u在一个分支上，若h[u/2]更大，则将对应元素上移

{

heap\_swap(u / 2, u);

u /= 2;

}

}

int main()

{

int n, m = 0; //m用于表示第几个插入的数

cin >> n;

while (n -- )

{

char op[5];

int k, x;

cin >> op;

if (!strcmp(op, "I")) //插入一个数(插到当前堆的最后一位)

{

cin >> x;

\_size ++;

m ++;

ph[m] = \_size, hp[\_size] = m;

h[\_size] = x;

up(\_size);

}

else if(!strcmp(op, "PM")) cout << h[1] << endl; //输出最小值

else if(!strcmp(op, "DM")) //删除最小值

{

heap\_swap(1, \_size);

\_size --;

down(1);

}

else if(!strcmp(op, "D")) //删除第k个插入的数

{

cin >> k;

k = ph[k];

heap\_swap(k, \_size);

\_size --;

up (k), down(k); //二者执行其一

}

else if(!strcmp(op,"C")) //修改第k个插入的数

{

cin >> k >> x;

k = ph[k];

h[k] = x;

up(k), down(k);

}

}

}

代码3：

//拉链法

#include <iostream>

#include <cstring>

using namespace std;

const int N = 100003; //100003是大于10000的最小质数，此时冲突的数最少

int e[N], ne[N], h[N], idx; //h[]表示的是一个槽

void insert(int x) //向哈希表中插入数x及对应的哈希值h(x)

{

int k = (x % N + N) % N; //k为x对应的哈希值（保证了k是正数，其值等于 x % N）

e[idx] = x;

ne[idx] = h[k]; //h[N]相当于每个单链表的head指针

h[k] = idx ++;

}

bool find(int x)

{

int k = (x % N + N) % N;

for (int i = h[k]; i != -1; i = ne[i])

{

if(e[i] == x) return true;

}

return false;

}

int main()

{

int n;

cin >> n;

memset(h, -1, sizeof h); //将槽先清空，把h[]数组的数全部设为-1（空指针一般用 -1 来表示）

while (n --)

{

char op[2];

int x;

scanf("%s%d", op, &x);

if(!strcmp(op,"I")) insert(x);

else

{

if (find(x)) puts("Yes");

else puts("No");

}

}

}

代码4：

#include <iostream>

using namespace std;

typedef unsigned long long ULL; //ull最大值正好是2的64次方-1，如果溢出则相当于mod(2^64)

const int N = 1E5 + 10, P = 131;

char str[N];

ULL h[N], p[N]; //p[i]存P的i次方

ULL get(int l, int r)

{

return h[r] - h[l - 1] \* p[r - l + 1]; //求子段hash值的公式

}

int main()

{

int n, m;

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

scanf("%s", str + 1);

p[0] = 1; //给p[0]初始化，即P的0次方是1

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

{

h[i] = h[i - 1] \* P + str[i];

p[i] = p[i - 1] \* P;

}

while (m -- )

{

int l1, r1, l2, r2;

scanf("%d%d%d%d", &l1, &r1, &l2, &r2);

if (get(l1,r1) == get(l2,r2)) puts("Yes");

else puts("No");

}

}

代码5：

#include <iostream>

using namespace std;

const int N = 10;

int n, path[N], state[N]; //state[i] ==1表示该《数字》已用过，==0表示没用过

void dfs(int u) //u表示的是《位置》

{

if (u == n)

{

for (int i = 0; i < n; i ++ ) printf("%d ", path[i]);

puts("");

return;

}

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

{

if (!state[i]) //当前数i尚未被用过

{

path[u] = i;

state[i] = 1;

dfs(u + 1); //递归，填下一个位置

state[i] = 0; //回溯，重置i使其变成没用过（递归返回后，i从大数开始往小数回，将对应state置为0）

}

}

}

int main()

{

scanf("%d", &n);

dfs(0);

}

代码6：

#include <iostream>

using namespace std;

const int N = 10;

int n;

char g[N][N];

bool col[N], dg[2 \* N], udg[2 \* N];

void dfs(int u)

{

if (u == n)

{

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

puts( g[i] );

puts("");

return;

}

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

{

if (!col[i] && !dg[n + u - i] && !udg[u + i]) //此三项都是0，即三条线上没有Queen

{

g[u][i] = 'Q';

col[i] = dg[n + u - i] = udg[u + i] = true;

dfs (u + 1);

col[i] = dg[n + u - i] = udg[u + i] = false; //恢复现场，回溯后将原先用过的列、正反对角线置为0

g[u][i] = '.';

}

}

}

int main()

{

scanf("%d", &n);

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

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

g[i][j] = '.' ;

dfs(0);

}

代码7：

#include <iostream>

using namespace std;

const int N = 10;

int n;

char g[N][N];

bool col[N], dg[2 \* N], udg[2 \* N];

void dfs(int u)

{

if (u == n)

{

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

puts( g[i] );

puts("");

return;

}

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

{

if (!col[i] && !dg[n + u - i] && !udg[u + i]) //此三项都是0，即三条线上没有Queen

{

g[u][i] = 'Q';

col[i] = dg[n + u - i] = udg[u + i] = true;

dfs (u + 1);

col[i] = dg[n + u - i] = udg[u + i] = false; //恢复现场，回溯后将原先用过的列、正反对角线置为0

g[u][i] = '.';

}

}

}

int main()

{

scanf("%d", &n);

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

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

g[i][j] = '.' ;

dfs(0);

}

代码8：

#include <cstring>

#include <iostream>

using namespace std;

const int N = 1e5 + 10;

const int M = 2 \* N; //以有向图的方式存储无向图，所以最多有2n-2条边

int h[N]; //存所有的头结点

int e[M]; //存储元素（每个链表中至多会有M个数据）

int ne[M]; //存储链表的next值

int idx, n;

int ans = N;

bool st[N]; //记录结点是否被访问过

void add(int a, int b)

{

e[idx] = b;

ne[idx] = h[a];

h[a] = idx ++;

}

int dfs(int u) //返回以u为根的子树中，结点的个数（包含u）

{

st[u] = true;

int size = 0; //存删掉u结点后，剩余连通块中最大的节点数

int sum = 1; //存 以u为根的子树 的结点数，u为第一个

for(int i = h[u]; i != -1; i = ne[i]) //访问u的子节点

{

int j = e[i];

if(!st[j]) //若结点没被访问过

{

int s = dfs(j); //递归处理其子结点，并用s存储子树的结点数

size = max(size, s); //记录最大连通子图的节点数

sum += s;

}

}

size = max(size, n - sum);

ans = min(size, ans);

return sum ;

}

int main()

{

cin >> n;

memset(h, -1, sizeof h);

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

{

int a, b;

cin >> a >> b;

add(a, b), add(b, a); //无向图

}

dfs(1);

cout << ans;

}

1. 该表信息将会作为你报名申请的重要依据，请认真仔细填写。
2. 培训班有严格的制度，请认真阅读规则并结合自身情况填写该表。
3. 完成该表填写后以“xxx学习情况”命名，及时上传作业。

最后希望大家能够加入我们，一起努力，共同进步！