sorting\_winner\_tree

1. The algorithm of this program
2. 讀input.txt檔，並將讀到的資訊作處理跟儲存，讀到的第一行存成player，第二行作字串的切割，並用player的值去計算總共有多少節點(node\_number)，用node\_number動態起始陣列(array\_tree)，並將切割的值存入陣列array\_tree
3. 起始Winner Tree，寫一個函式min傳入兩個變數回傳較小的值，再將有同樣父節點的兩個子節點傳入min作判斷，再把min回傳的值存入父節點，如果父節點已經有值(值不為0)就將回傳的值跟父節點的值傳入min作判斷，回傳值再填入父節點，如果父節點沒有值(值為0)就直接存，用for(int i = getNode\_number() - 1; i >= 0; i = i -2)就可以起始整顆樹
4. 利用Winner Tree作排序，用動態起始一個大小為Player數的一維陣列(sorted\_array)，因為起始後的 Winner Tree的根為Player中的最小值，就將根的值存入sorted\_array，並將根的值設為10，在找到根這個值的起始位置，果根的值起初的位置是奇數，用for從a(Player在array\_tree中的第一個位置)跑到0，並每次都讓記數的變數i，減1再除2，並將array\_tree[i] 設成0，如果根的值起初的位置是偶數，則記數的變數i，先除2再減1。再重新判斷每個有子節點的父節點，將兩個子節點的值傳入min作判斷，再把min回傳的值存入父節點。將上述步驟跑過Player的值，就可以得到排序好的值。
5. The program

///////////////////////////////////////////////////////////////////////////////

// 程式: Homework chap05 - Winner Tree Sorting

// 版本: array(class)

// 日期: 2013/3/11

///////////////////////////////////////////////////////////////////////////////

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

class Wintree{ // 宣告 Wintree class

public:

Wintree(fstream &filei){

int n; // Player

char buffer[100];

filei >> n;

filei.getline(buffer,sizeof(buffer)); // 抓換行

filei.getline(buffer,sizeof(buffer)); // 抓player的值(一行)

setPlayer(n);

setNode\_number(n);

setSorted\_array(n);

setArray\_tree(getNode\_number(), buffer);

} // 建構子(Wintree)

void setPlayer(int in){player = in;} // set Player 的值

void setNode\_number(int n){ // 傳入 n(player)

int h = 0, in = 1; // h = height

while(n != 0){

n = n/2;

h++;

} // 計算樹高度

height = h;

for(int i = 0; i < h; i++) in\*=2;

node\_number = in - 1;

} // set Node\_number 的值(計算樹共有多少節點), 也計算 height 的值

void setArray\_tree(int n, char buffer[]){// (總共有多少節點, 讀的Player字串)

char \*pch; // 字元指標用來計被切割的值

int a = getNode\_number() - getPlayer(); // 從第幾個開始有值

array\_tree = new int[n]; // 動態起始陣列

for(int i = 0; i < n; i++) array\_tree[i] = 0; // 將 array\_tree 歸零

pch = strtok(buffer, " ");

while(pch != NULL){

array\_tree[a] = atoi(pch); // 將字元轉成整數

pch = strtok(NULL," :");

a++;

} // 切割字串並存入 array\_tree

} // set Array\_tree

void setSorted\_array(int n){

sorted\_array = new int[n]; // 動態起始陣列

for(int i = 0; i < n; i++) sorted\_array[i] = 0; // 將 sorted\_array 歸零

} // 起始 sorted\_array

int min(int a, int b){

if(a>b)return b;

else return a;

} // 判斷傳入的兩數誰較小 迴傳較小的值

void instar(){

int n;

for(int i = getNode\_number() - 1; i >= 0; i = i -2){ // getNode\_number() = 31

n = min(array\_tree[i], array\_tree[i-1]); // 判斷哪個子節點較小

if(0!=array\_tree[(i/2)-1]){ // 如果父節點已經有值(0)

array\_tree[(i/2)-1] = min(array\_tree[(i/2)-1], n);

} // 將 n 跟父節點的值丟到 min

else array\_tree[(i/2)-1] = n; // 如果父節點沒有值 存值

}

} // 初始整顆樹

void Retrieve(int n){ // n = 排序的位置

int a = getNode\_number() - getPlayer(), find = 0, b;

// Player在array中的第一個位置, 是否找到Winner的位置, 暫存min值

sorted\_array[n] = array\_tree[0];// 將Winner的值存到sorted\_array

array\_tree[0] = 10; // 將Winner用10取代

while(find != 1){

if(array\_tree[a] == sorted\_array[n]){ // 找Winner起初的位置

array\_tree[a] = 10;

if(a%2 != 0){ // 如果Winner起初的位置是奇數

for(int i = a; i>0; i = (i-1)/2)

array\_tree[i] = 10;

}

if(a%2 == 0){ // 如果Winner起初的位置是偶數

for(int i = a; i>0; i = (i/2)-1)

array\_tree[i] = 10;

}

find = 1;

}

else a++;

} // while, 找到Winner的位置就跳出

for(int i = getNode\_number() - 1; i >= 0; i = i -2){

b = min(array\_tree[i], array\_tree[i-1]); // 判斷哪個子節點較小

array\_tree[(i/2)-1] = b; // 存值

}

} // 排序

int getPlayer(){return player;} // 取 player 的值

int getNode\_number(){return node\_number;} // 取 node\_number 的值

int getHeight(){return height;} // 取 height 的值

void printf\_tree(fstream &fileo){

for(int i =0; i < getNode\_number(); i++){

if(array\_tree[i] == 10){

cout << "#";

fileo << "#";

}

else {

cout << array\_tree[i];

fileo << array\_tree[i];

}

cout << " ";

fileo << " ";

}

cout << endl;

fileo << endl;

} // 印出array\_tree的值

void printf\_sorted(fstream &fileo){

cout << "Sorted array: ";

fileo << "Sorted array: ";

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

if(sorted\_array[i]==0){

cout << "-";

fileo << "-";

}

else {

cout << sorted\_array[i];

fileo << sorted\_array[i];

}

cout << " ";

fileo << " ";

}

cout << endl;

fileo << endl;

} // 印出sorted\_array的值

private:

int player; // player

int node\_number; // 陣列的個數

int height; // 樹的高度(在setNode\_number中找值)

int \*array\_tree;

int \*sorted\_array; // 排序後的值

};

// Wintree class 宣告結束

void funtion(fstream &filei, fstream &fileo){ // funtion

class Wintree w(filei); // 宣告一個名為 w 的 Wintree class

cout << "Before tree initialization:\n";

fileo << "Before tree initialization:\n";

w.printf\_tree(fileo); // 只有 player 的狀態

cout << "After tree initialization:\n";

fileo << "After tree initialization:\n";

w.instar(); // 起始樹

w.printf\_tree(fileo);

for(int i = 0; i < w.getPlayer(); i++){ // 排序

w.Retrieve(i);

cout << "Retrieve " << i+1 << " number(s):\n";

fileo << "Retrieve " << i+1 << " number(s):\n";

cout << "tree: ";

fileo << "tree: ";

w.printf\_tree(fileo);

w.printf\_sorted(fileo);

}

}

///////////////////////////////////////////////////////////////////////////////

int main(){

fstream filei, fileo;

filei.open("input.txt", ios::in); // 讀input.txt

fileo.open("output.txt", ios::out); // 寫output.txt

funtion(filei, fileo); // 呼叫funtion

system("pause");

return 0;

}

1. The data used to test the program.

Input檔: k = 16

數字為: 4 3 6 8 1 5 7 3 2 6 9 4 5 2 5 8

e. The output file

Before tree initialization:

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 3 6 8 1 5 7 3 2 6 9 4 5 2 5 8

After tree initialization:

1 1 2 3 1 2 2 3 6 1 3 2 4 2 5 4 3 6 8 1 5 7 3 2 6 9 4 5 2 5 8

Retrieve 1 number(s):

tree: 2 3 2 3 3 2 2 3 6 5 3 2 4 2 5 4 3 6 8 # 5 7 3 2 6 9 4 5 2 5 8

Sorted array: 1 - - - - - - - - - - - - - - -

Retrieve 2 number(s):

tree: 2 3 2 3 3 4 2 3 6 5 3 6 4 2 5 4 3 6 8 # 5 7 3 # 6 9 4 5 2 5 8

Sorted array: 1 2 - - - - - - - - - - - - - -

Retrieve 3 number(s):

tree: 3 3 4 3 3 4 5 3 6 5 3 6 4 5 5 4 3 6 8 # 5 7 3 # 6 9 4 5 # 5 8

Sorted array: 1 2 2 - - - - - - - - - - - - -

Retrieve 4 number(s):

tree: 3 3 4 4 3 4 5 4 6 5 3 6 4 5 5 4 # 6 8 # 5 7 3 # 6 9 4 5 # 5 8

Sorted array: 1 2 2 3 - - - - - - - - - - - -

Retrieve 5 number(s):

tree: 4 4 4 4 5 4 5 4 6 5 7 6 4 5 5 4 # 6 8 # 5 7 # # 6 9 4 5 # 5 8

Sorted array: 1 2 2 3 3 - - - - - - - - - - -

Retrieve 6 number(s):

tree: 4 5 4 6 5 4 5 # 6 5 7 6 4 5 5 # # 6 8 # 5 7 # # 6 9 4 5 # 5 8

Sorted array: 1 2 2 3 3 4 - - - - - - - - - -

Retrieve 7 number(s):

tree: 5 5 5 6 5 6 5 # 6 5 7 6 9 5 5 # # 6 8 # 5 7 # # 6 9 # 5 # 5 8

Sorted array: 1 2 2 3 3 4 4 - - - - - - - - -

Retrieve 8 number(s):

tree: 5 6 5 6 7 6 5 # 6 # 7 6 9 5 5 # # 6 8 # # 7 # # 6 9 # 5 # 5 8

Sorted array: 1 2 2 3 3 4 4 5 - - - - - - - -

Retrieve 9 number(s):

tree: 5 6 5 6 7 6 5 # 6 # 7 6 9 # 5 # # 6 8 # # 7 # # 6 9 # # # 5 8

Sorted array: 1 2 2 3 3 4 4 5 5 - - - - - - -

Retrieve 10 number(s):

tree: 6 6 6 6 7 6 8 # 6 # 7 6 9 # 8 # # 6 8 # # 7 # # 6 9 # # # # 8

Sorted array: 1 2 2 3 3 4 4 5 5 5 - - - - - -

Retrieve 11 number(s):

tree: 6 7 6 8 7 6 8 # 8 # 7 6 9 # 8 # # # 8 # # 7 # # 6 9 # # # # 8

Sorted array: 1 2 2 3 3 4 4 5 5 5 6 - - - - -

Retrieve 12 number(s):

tree: 7 7 8 8 7 9 8 # 8 # 7 # 9 # 8 # # # 8 # # 7 # # # 9 # # # # 8

Sorted array: 1 2 2 3 3 4 4 5 5 5 6 6 - - - -

Retrieve 13 number(s):

tree: 8 8 8 8 # 9 8 # 8 # # # 9 # 8 # # # 8 # # # # # # 9 # # # # 8

Sorted array: 1 2 2 3 3 4 4 5 5 5 6 6 7 - - -

Retrieve 14 number(s):

tree: 8 # 8 # # 9 8 # # # # # 9 # 8 # # # # # # # # # # 9 # # # # 8

Sorted array: 1 2 2 3 3 4 4 5 5 5 6 6 7 8 - -

Retrieve 15 number(s):

tree: 9 # 9 # # 9 # # # # # # 9 # # # # # # # # # # # # 9 # # # # #

Sorted array: 1 2 2 3 3 4 4 5 5 5 6 6 7 8 8 -

Retrieve 16 number(s):

tree: # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #

Sorted array: 1 2 2 3 3 4 4 5 5 5 6 6 7 8 8 9

1. Any difficulties encountered (at least 50 words)
2. winner tree陣列總共的大小
3. 用起始後的winner tree來找出檔案中所有值的排序，找到值後，將值用一個很一個很大的值去取代，在重新起始樹找下一個值
4. How you resolved the problems (at least 50 words)
5. 找以前的講義，有關於完整二元樹的介紹，跟相關的特性。winner tree陣列的大小 = 2的 height(樹的高度)次方
6. 先在紙上畫出樹，並標記陣列的位置，再找出規則，我們程式根在陣列的位置是0，發現如果根的起初的位置是奇數，那個值的跟他上面節點的關係是(i-1)/2，如果是偶數關係是(i/2)-1，就可以找到應該被取代的位置。