#include<iostream>

#include<string>

#include<algorithm>

#include<queue>

#include<vector>

#include<sstream>

#include<stack>

#define MAX 1005

using namespace std;

int store[MAX], num;

int index[11] = { 1,2,4,8,16,32,64,128,256,512,1024 };

//int index\_add[11] = { 1,3,7,15,31,63,127,255,511,1023 };

struct Bitree {

int key;

Bitree\* LeftC;

Bitree\* RightC;

};

Bitree\* dfs(int start\_pos,int stop\_pos)

{

Bitree\* root;

int size = stop\_pos - start\_pos + 1;

if (size == 1)

{

root = (Bitree\*)malloc(sizeof(Bitree));

root->key = store[start\_pos];

root->LeftC = NULL;

root->RightC = NULL;

return root;

}

if (size <= 0)

{

return NULL;

}

int total = 0,sub,flag;

for (int i = 0; i < 11; i++)

{

total += index[i];

if (total >size)

{

total -= index[i];

sub = size - total;

if (sub < index[i - 1])

{

int rootpos = stop\_pos-(size - 1 - sub) / 2;

root = (Bitree\*)malloc(sizeof(Bitree));

root->key = store[rootpos];

root->LeftC = dfs(start\_pos, rootpos - 1);

root->RightC = dfs(rootpos+1,stop\_pos);

return root;

}

else

{

int rootpos = start\_pos + (total - 1) / 2 + index[i - 1];

root = (Bitree\*)malloc(sizeof(Bitree));

root->key = store[rootpos];

root->LeftC = dfs(start\_pos, rootpos - 1);

root->RightC = dfs(rootpos + 1, stop\_pos);

return root;

}

}

}

}

queue<Bitree\*>waitlist;

int main()

{

cin >> num;

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

cin >> store[i];

sort(store, store + num);

Bitree\* head=dfs(0, num - 1);

cout << head->key;

if (head->LeftC != NULL)

waitlist.push(head->LeftC);

if (head->RightC != NULL)

waitlist.push(head->RightC);

while (waitlist.size())

{

Bitree\* temp = waitlist.front();

waitlist.pop();

cout << " " << temp->key;

if (temp->LeftC != NULL)

waitlist.push(temp->LeftC);

if (temp->RightC != NULL)

waitlist.push(temp->RightC);

}

return 0;

}

//第一次自己写完了一根树，感动

//后面还有一个简便算法，应该去研究一下。

1. #include <cstdio>
2. #include <cstdlib>

5. const int maxx = 1005;
6. int node[maxx];
7. int tree[maxx];
8. int pos,n;
10. int cmp(const void \*a,const void \*b){
11. int \*pa = (int \*)a;
12. int \*pb = (int \*)b;
13. return \*pa-\*pb;
14. }
16. void build(int root){
17. if(root>n)return;
18. int lson = root<<1,rson = (root<<1)+1;
19. build(lson);
20. tree[root] = node[pos++];
21. build(rson);
22. }
24. void print(int \*a,int n){
25. int i;
26. for(i=0;i<n;++i){
27. printf("%d ",a[i]);
28. }
29. printf("\n");
30. }
32. int main()
33. {
34. int i;
35. scanf("%d",&n);
36. for(i=0;i<n;++i){
37. scanf("%d",&node[i]);
38. }
40. qsort(node,n,sizeof(int),cmp);
42. // print(node,n);
44. pos = 0;
45. build(1);
47. printf("%d",tree[1]);
48. for(i=2;i<=n;++i){
49. printf(" %d",tree[i]);
50. }
51. printf("\n");
52. return 0;
53. } }

懒得写了，把这道代码的原理给分析一下：

完全二叉树的特性：

对于任何一个序列为i的子节点，它的左子节点序号为2i,右子节点的序号为2i+1；

则，根据这个性质，能得出结点为n时的中序序列的节点输出顺序：

其原理为：

递归算法：

根节点序号为1；

将2\*1放入1的左边；

对于2进行追溯；

若任何子节点数列大于节点个数，则退出；

否则把左节点放在父节点左边

右节点放在父节点右边