2.42

#include<iostream>

#include<vector>

using namespace std;

vector<int>biao;

vector<int> x;

int n;

void out()

{

for (const int & a : x)

{

cout << a << " ";

}

cout << "\n";

}

void insert()

{

x = biao;

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

{

int mid = n1;

int temp = x[n1];

while (mid > 0 && x[mid - 1] > temp)

{

x[mid] = x[mid - 1];

mid--;

}

x[mid] = temp;

out();

}

}

void simple()

{

x = biao;

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

{

int k=n1;

for (int n2 = n1+1; n2 < n; n2++)

{

if (x[n2] <x[k])

{

k = n2;

}

}

int mid = x[k];

x[k] = x[n1];

x[n1] = mid;

out();

}

}

void downadjust(int low, int high)

{

int i = low, j = low \* 2;

while (j < high)

{

if (j + 1 < high&&x[j + 1] > x[j])

{

j = j + 1;

}

if (x[j] > x[i])

{

int temp = x[j];

x[j] = x[i];

x[i] = temp;

i = j;

j = i \* 2;

}

else

{

break;

}

}

}

void creatheap()

{

for (int n1 = n / 2; n1 >= 0; n1--)

{

downadjust(n1, n);

}

out();

}

void heap()

{

x = biao;

creatheap();

for (int n1 = n - 1; n1 > 0; n1--)

{

int temp = x[n1];

x[n1] = x[0];

x[0] = temp;

downadjust(0, n1);

out();

}

}

void bob()

{

x = biao;

int k = n - 1;

for (int n1 = n - 1; n1 > 0;)

{

int mid = 0;

while (mid < n1)

{

if (x[mid] > x[mid + 1])

{

int temp = x[mid];

x[mid] = x[mid + 1];

x[mid + 1] = temp;

k = mid;

}

mid += 1;

}

if (n1 - 1 < k)

{

n1 = n1 - 1;

}

else

{

n1 = k;

}

out();

}

}

int part(int left, int right)

{

int mid = x[left];

while (left < right)

{

while (left<right&&x[right]>mid)

{

right--;

}

x[left] = x[right];

while (left < right&&x[left] <= mid)

{

left++;

}

x[right] = x[left];

}

x[left] = mid;

return left;

}

void quicksort(int left, int right)

{

if (left < right)

{

int pos = part(left, right);

quicksort(left, pos-1);

quicksort(pos+1, right);

}

out();

}

void quick()

{

x = biao;

quicksort(0, n - 1);

out();

}

int main()

{

cin >> n;

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

{

int mid;

cin >> mid;

biao.push\_back(mid);

}

cout << "插入排序\n";

insert();

cout << "简单排序\n";

simple();

cout << "堆排序\n";

heap();

cout << "起泡排序\n";

bob();

cout << "快排\n";

quick();

system("pause");

return 0;

}

/\*

输入

12

41 62 13 84 35 96 57 39 79 61 15 83

输出

插入排序

41 62 13 84 35 96 57 39 79 61 15 83

13 41 62 84 35 96 57 39 79 61 15 83

13 41 62 84 35 96 57 39 79 61 15 83

13 35 41 62 84 96 57 39 79 61 15 83

13 35 41 62 84 96 57 39 79 61 15 83

13 35 41 57 62 84 96 39 79 61 15 83

13 35 39 41 57 62 84 96 79 61 15 83

13 35 39 41 57 62 79 84 96 61 15 83

13 35 39 41 57 61 62 79 84 96 15 83

13 15 35 39 41 57 61 62 79 84 96 83

13 15 35 39 41 57 61 62 79 83 84 96

简单排序

13 62 41 84 35 96 57 39 79 61 15 83

13 15 41 84 35 96 57 39 79 61 62 83

13 15 35 84 41 96 57 39 79 61 62 83

13 15 35 39 41 96 57 84 79 61 62 83

13 15 35 39 41 96 57 84 79 61 62 83

13 15 35 39 41 57 96 84 79 61 62 83

13 15 35 39 41 57 61 84 79 96 62 83

13 15 35 39 41 57 61 62 79 96 84 83

13 15 35 39 41 57 61 62 79 96 84 83

13 15 35 39 41 57 61 62 79 83 84 96

13 15 35 39 41 57 61 62 79 83 84 96

13 15 35 39 41 57 61 62 79 83 84 96

堆排序

96 84 83 57 79 62 41 39 35 61 15 13

84 83 79 57 61 62 41 39 35 13 15 96

83 79 62 57 61 15 41 39 35 13 84 96

79 62 61 57 35 15 41 39 13 83 84 96

62 61 35 57 13 15 41 39 79 83 84 96

61 57 35 41 13 15 39 62 79 83 84 96

57 41 35 39 13 15 61 62 79 83 84 96

41 39 35 15 13 57 61 62 79 83 84 96

39 35 13 15 41 57 61 62 79 83 84 96

35 15 13 39 41 57 61 62 79 83 84 96

15 13 35 39 41 57 61 62 79 83 84 96

13 15 35 39 41 57 61 62 79 83 84 96

起泡排序

41 13 62 35 84 57 39 79 61 15 83 96

13 41 35 62 57 39 79 61 15 83 84 96

13 35 41 57 39 62 61 15 79 83 84 96

13 35 41 39 57 61 15 62 79 83 84 96

13 35 39 41 57 15 61 62 79 83 84 96

13 35 39 41 15 57 61 62 79 83 84 96

13 35 39 15 41 57 61 62 79 83 84 96

13 35 15 39 41 57 61 62 79 83 84 96

13 15 35 39 41 57 61 62 79 83 84 96

13 15 35 39 41 57 61 62 79 83 84 96

快排

13 15 39 35 41 96 57 84 79 61 62 83

13 15 35 39 41 96 57 84 79 61 62 83

13 15 35 39 41 96 57 84 79 61 62 83

13 15 35 39 41 96 57 84 79 61 62 83

13 15 35 39 41 96 57 84 79 61 62 83

13 15 35 39 41 57 61 62 79 83 84 96

13 15 35 39 41 57 61 62 79 83 84 96

13 15 35 39 41 57 61 62 79 83 84 96

13 15 35 39 41 57 61 62 79 83 84 96

13 15 35 39 41 57 61 62 79 83 84 96

13 15 35 39 41 57 61 62 79 83 84 96

13 15 35 39 41 57 61 62 79 83 84 96

13 15 35 39 41 57 61 62 79 83 84 96

13 15 35 39 41 57 61 62 79 83 84 96

13 15 35 39 41 57 61 62 79 83 84 96

13 15 35 39 41 57 61 62 79 83 84 96

\*/2.41

#include<iostream>

#include<vector>

#include<string>

using namespace std;

#define dj (randlist[use]+d1)%13

string hash1[13];

bool hash2[13];

int randlist[5] = { 3,7,1,12,10 };

int d1;

void combat(string x)

{

int use = 0;

while (1)

{

if (use > 4)

{

use = 0;

}

int mid = dj;

if (hash2[mid] != 1)

{

hash1[mid] = x;

hash2[mid] = 1;

return ;

}

use += 1;

}

}

int main()

{

for (int n1 = 0; n1 < 13; n1++)

{

hash2[n1] = 0;

}

int n;

cin >> n;

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

{

string mid;

cin >> mid;

int midn = mid[0] - 65;

if (hash2[midn] == 0)

{

hash2[midn] = 1;

hash1[midn] = mid;

}

else

{

d1 = midn;

combat(mid);

}

}

for (int n1=0;n1<13;n1++)

{

if (hash2[n1])

{

cout << hash1[n1] << " ";

}

}

system("pause");

return 0;

}

/\*

输入

10

ICKES BARBER ELYOT KERN FRENCE LOWER BENSDN FONK ERVIN KNOX

输出

KNOX BARBER BENSDN ELYOT FRENCE ERVIN ICKES KERN LOWER FONK

\*/2.40

#include<iostream>

#include<vector>

#include<algorithm>

#include<queue>

using namespace std;

struct node

{

node\* left, \*right;

int val;

};

vector<int> num;

void build(node \* & x,int n)

{

if (n >= num.size())

{

x = NULL;

return;

}

x = new(node);

build(x->left, 2 \* n);

x->val = num[n];

build(x->right, 2 \* n + 1);

}

void levelorder(node \* x)

{

queue<node \*> qu;

qu.push(x);

int nq = qu.size();

while (!qu.empty())

{

node \* mid = qu.front();

qu.pop();

cout << mid->val << " ";

if (mid->left != NULL)

{

qu.push(mid->left);

}

if (mid->right != NULL)

{

qu.push(mid->right);

}

nq--;

if (nq == 0)

{

nq = qu.size();

cout << "\n";

}

}

return;

}

int main()

{

int n;

cin >> n;

num.push\_back(0);

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

{

int mid;

cin >> mid;

num.push\_back(mid);

}

sort(num.begin()+1, num.end());

node \* root;

build(root,1);

levelorder(root);

system("pause");

return 0;

}

/\*

输入

20

1 5 8 9 4 7 6 2 23 78 54 89 96 21 45 16 57 81 39 48

输出

1

2 4

5 6 7 8

9 16 21 23 39 45 48 54

57 78 81 89 96

\*/2.39

#include<iostream>

#include<vector>

#include<algorithm>

using namespace std;

struct block

{

int boock\_val;

vector<int> stu\_id;

};

const int block\_num = 4;

int main()

{

int n;

cin >> n;

vector<block \*> sum;

int start = 97200;

for (int n1 = 0; n1 < block\_num; n1++)

{

block \* mid = new (block);

mid->boock\_val = start;

sum.push\_back(mid);

start += 100;

}

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

{

int mid;

cin >> mid;

if (mid > 97100 && mid <= 97200)

{

sum[0]->stu\_id.push\_back(mid);

continue;

}

else if (mid > 97200 && mid <= 97300)

{

sum[1]->stu\_id.push\_back(mid);

continue;

}

else if (mid > 97300 && mid <= 97400)

{

sum[2]->stu\_id.push\_back(mid);

continue;

}

else if (mid > 97500 && mid <= 97600)

{

sum[3]->stu\_id.push\_back(mid);

continue;

}

}

for (int n1 = 0; n1 < block\_num; n1++)

{

sort(sum[n1]->stu\_id.begin(), sum[n1]->stu\_id.end());

cout << sum[n1]->boock\_val << '\n';

for (int n2 = 0; n2 < sum[n1]->stu\_id.size(); n2++)

{

cout << sum[n1]->stu\_id[n2] << " ";

}

cout << "\n";

}

system("pause");

return 0;

}

/\*

输入

17

97438 97102 97528 97136 97338 97250 97407 97239 97227 97517 97321 97421 97451 97241 97118 97543 97309

输出

97200

97102 97118 97136

97300

97227 97239 97241 97250

97400

97309 97321 97338

97500

97517 97528 97543

\*/