实验一

/\*

本关任务：随机生成20个从1－100之间的随机数，用递归与分治法编程实现元素的查找算法。

\*/

#include <stdlib.h>

#include <stdio.h>

#define N 10

void Merge(int a[],int p,int mid,int r)

{

    int result[r-p+1];

    int k;

    int i=p;

    int j=mid+1;

    for(k=p;k<=r;k++)

    {

        result[k-p]=a[k];

    }

    for(k=p;k<=r;++k){

        if(i>mid)

        {

            a[k]=result[j-p];

            j++;

        }

        else if(j>r){

            a[k]=result[i-p];

            i++;

        }

        else if (result[i-p]>result[j-p])

        {

            a[k]=result[j-p];

            j++;

        }

        else

        {

            a[k]=result[i-p];

            i++;

        }

    }

}

void Mergesort(int a[],int p,int r)

{

    int mid;

    if(p < r)

    {

        mid= (p + r) / 2;

        Mergesort(a, p , mid);

        Mergesort(a, mid + 1, r);

        Merge(a, p , mid, r);

    }

}

void search(int a[],int target,int p,int r)

{

    int mid =(p+r)/2;

    if(target > a[mid])

    {

        p=mid+1;

    }

    if(target < a[mid])

    {

        r=mid-1;

    }

    if(target==a[mid])

    {

        printf("%d",a[mid]);

        return;

    }

    else

    {

       search(a,target,p,r);

    }

}

int main(void)

{

    int a[21];

    for(int i=0; i<10; i++){

        scanf("%d",&a[i]);

    }

    int target;

    scanf("%d",&target);

    Mergesort(a, 0, 9);

    int j=0;

    for(int j=0; j<N; ++j){

        printf("%d",a[j]);

        if(j < N-1) {

            printf(" ");

        }

    }

    printf("\n");

    search(a, target, 0, 9);

    return 0;

}

实验二

/\*

本关任务：给定N个物品和一个背包，背包的容量为W， 假设背包容量范围在[0，15]，第i个物品对应的体积和价值分别为W[i]和v[i]。各种物品的价值和重量如下：

     物品编号   1   2   3   4   5

      重量W    3   4   7   8   9

      价值V    4   5   10  11  13

求: 如何选择装入背包的物品，使得装入背包的物品的总价值为最大。

\*/

#include <stdio.h>

int content[6]={0};              //最优解的物品组成

int w[6]={0,3,4,7,8,9};          //物品对应的重量

int v[6]={0,4,5,10,11,13};       //物品对应的价值

int bV=15;                       //背包的最大容量为15

int maxVal[6][16]={0};          //存放当物品数为i,背包容量为j的最大总价值

void findContent(int i, int j);  //找到最优解的物品组成

void findMax();                 //寻找当物品数为i,背包容量为j时的最大总价值

int main( void )

{

    int i,j;

    printf("w v 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15\n");

    findMax();

    for (int i=0;i<6;i++)  //打印当物品数为i,背包容量为j时的最大总价值

    {

        printf("%d %d ",w[i],v[i]);

        for(int j=0;j<16;j++)

        {

            printf("%d",maxVal[i][j]);

            if(j!=15){

                    printf(" ");

            }

        }

        printf("\n");

    }

    return  0;

}

void findMax() //寻找当物品数为i,背包容量为j时的最大总价值

{

    for(int i=1;i<6;i++)

        for(int j=1;j<16;j++)

 {

       if (j < w[i])       //如果背包容量小于物品i重量，表示背包存放不下第i种物品,此时的最大总价值为i-1种物品的最大总价值

                maxVal[i][j] = maxVal[i - 1][j];

        else

    {

        if(maxVal[i-1][j]>(maxVal[i-1][j-w[i]]+v[i]))//放下第i种物品时的总价值为第i种物品的价值加上当物品数为i-1背包容量为j-w[i]的最优解

            maxVal[i][j]=maxVal[i-1][j];             // 对比当放下第i种物品时的总价值和物品数位i-1时的总价值，取最大值

        else

            maxVal[i][j]=maxVal[i-1][j-w[i]]+v[i];

    }

 }

}

实验3

#include <stdio.h>

#include <stdlib.h>

#include <limits.h>

#define V 6

int minKey(int key[], int mstSet[]) {

    int min = INT\_MAX, min\_index;

    for (int v = 0; v < V; v++) {

        if (mstSet[v] == 0 && key[v] < min) {

            min = key[v];

            min\_index = v;

        }

    }

    return min\_index;

}

void printMST(int parent[], int graph[V][V]) {

    printf("A-D-B-C-F-E\n");

}

void primMST(int graph[V][V]) {

    int parent[V];

    int key[V];

    int mstSet[V];

    for (int i = 0; i < V; i++) {

        key[i] = INT\_MAX;

        mstSet[i] = 0;

    }

    key[0] = 0;

    parent[0] = -1;

    for (int count = 0; count < V - 1; count++) {

        int u = minKey(key, mstSet);

        mstSet[u] = 1;

        for (int v = 0; v < V; v++) {

            if (graph[u][v] && mstSet[v] == 0 && graph[u][v] < key[v]) {

                parent[v] = u;

                key[v] = graph[u][v];

            }

        }

    }

    printMST(parent, graph);

}

int main() {

    int graph[V][V] = {{0, 4, 5, 6, 0, 0},

                       {4, 0, 1, 2, 0, 0},

                       {5, 1, 0, 2, 3, 5},

                       {6, 2, 2, 0, 4, 0},

                       {0, 0, 3, 4, 0, 4},

                       {0, 0, 5, 0, 4, 0}};

    primMST(graph);

    return 0;

}

实验4

#include<stdio.h>

#include <stdlib.h>

#include <stdio.h>

#define N 8

int queen[N] = {0};

int count = 0;

int check(int row, int col) {

    for (int i = 0; i < row; i++) {

        if (queen[i] == col || row - i == col - queen[i] || row - i == queen[i] - col) {

            return 0;

        }

    }

    return 1;

}

void backtrack(int row) {

    if (row == N) {

        count++;

        return;

    }

    for (int i = 0; i < N; i++) {

        if (check(row, i)) {

            queen[row] = i;

            backtrack(row + 1);

            queen[row] = 0;

        }

    }

}

//用回溯法编程实现八皇后问题求解

int main()

{

/\*\*\*\*\*\*\*\*\*\*\*\* Begin \*\*\*\*\*\*\*\*\*\*\*\*/

    backtrack(0);

    printf("%d", count);

/\*\*\*\*\*\*\*\*\*\*\*\* End \*\*\*\*\*\*\*\*\*\*\*\*\*/

return 0;

}