分支界限法求解背包问题：

#include<stdio.h>

#include<stdlib.h>

#define MAXNUM 100

struct node{

int step;

double price;

double weight;

double max, min;

unsigned long po;

};

typedef struct node DataType;

struct SeqQueue { /\* 顺序队列类型定义 \*/

int f, r;

DataType q[MAXNUM];

};

typedef struct SeqQueue \*PSeqQueue;

PSeqQueue createEmptyQueue\_seq( void ) {

PSeqQueue paqu;

paqu = (PSeqQueue)malloc(sizeof(struct SeqQueue));

if (paqu == NULL)

printf("Out of space!! /n");

else

paqu->f = paqu->r = 0;

return paqu;

}

int isEmptyQueue\_seq( PSeqQueue paqu ) {

return paqu->f == paqu->r;

}

/\* 在队列中插入一元素x \*/

void enQueue\_seq( PSeqQueue paqu, DataType x ) {

if( (paqu->r + 1) % MAXNUM == paqu->f )

printf( "Full queue./n" );

else {

paqu->q[paqu->r] = x;

paqu->r = (paqu->r + 1) % MAXNUM;

}

}

/\* 删除队列头元素 \*/

void deQueue\_seq( PSeqQueue paqu ) {

if( paqu->f == paqu->r )

printf( "Empty Queue./n" );

else

paqu->f = (paqu->f + 1) % MAXNUM;

}

/\* 对非空队列,求队列头部元素 \*/

DataType frontQueue\_seq( PSeqQueue paqu ) {

return (paqu->q[paqu->f]);

}

/\* 物品按性价比从新排序\*/

void sort(int n, double p[], double w[]){

int i, j;

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

for (j = i; j < n-1; j++) {

double a = p[j]/w[j];

double b = p[j+1]/w[j+1];

if (a < b) {

double temp = p[j];

p[j] = p[j+1];

p[j+1] = temp;

temp = w[j];

w[j] = w[j+1];

w[j+1] = temp;

}

}

}

/\* 求最大可能值\*/

double up(int k, double m, int n, double p[], double w[]){

int i = k;

double s = 0;

while (i < n && w[i] < m) {

m -= w[i];

s += p[i];

i++;

}

if (i < n && m > 0) {

s += p[i] \* m / w[i];

i++;

}

return s;

}

/\* 求最小可能值\*/

double down(int k, double m, int n, double p[], double w[]){

int i = k;

double s = 0;

while (i < n && w[i] <= m) {

m -= w[i];

s += p[i];

i++;

}

return s;

}

/\* 用队列实现分支定界算法\*/

double solve(double m, int n, double p[], double w[], unsigned long\* po){

double min;

PSeqQueue q = createEmptyQueue\_seq();

DataType x = {0,0,0,0,0,0};

sort(n, p, w);

x.max = up(0, m, n, p, w);

x.min = min = down(0, m, n, p, w);

if (min == 0) return -1;

enQueue\_seq(q, x);

while (!isEmptyQueue\_seq(q)){

int step;

DataType y;

x = frontQueue\_seq(q);

deQueue\_seq(q);

if (x.max < min) continue;

step = x.step + 1;

if (step == n+1) continue;

y.max = x.price + up(step, m - x.weight, n, p, w);

if (y.max >= min) {

y.min = x.price + down(step, m-x.weight, n, p, w);

y.price = x.price;

y.weight = x.weight;

y.step = step;

y.po = x.po << 1;

if (y.min >= min) {

min = y.min;

if (step == n) \*po = y.po;

}

enQueue\_seq(q, y);

}

if (x.weight + w[step-1] <= m) {

y.max = x.price + p[step-1] +

up(step, m-x.weight-w[step-1], n, p, w);

if (y.max >= min) {

y.min = x.price + p[step-1] +

down(step, m-x.weight-w[step-1], n, p, w);

y.price = x.price + p[step-1];

y.weight = x.weight + w[step-1];

y.step = step;

y.po = (x.po << 1) + 1;

if (y.min >= min) {

min = y.min;

if (step == n) \*po = y.po;

}

enQueue\_seq(q, y);

}

}

}

return min;

}

#define n 4

double m = 15;

double p[n] = {10, 10, 12, 18};//价值

double w[n] = {2, 4, 6, 9}; //重量

int main() {

int i;

double d;

unsigned long po;

d = solve(m, n, p, w, &po);

printf("价值：");

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

printf("%f ",p[i]);

printf("\n");

printf("重量：");

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

printf("%f ",w[i]);

printf("\n");

if (d == -1)

printf("No solution!/n");

else {

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

printf("x%d is %d\n", i + 1, ((po & (1<<(n-i-1))) != 0));//Xn=1即代表将第n个物品放入背包，Xn=0即不放入。

printf("The max weight is %f\n", d);

}

return 0;

}