6种商品的重量分别为(1, 3, 7, 8, 10,13), 价值为 （1, 4, 8, 10, 11, 15)。

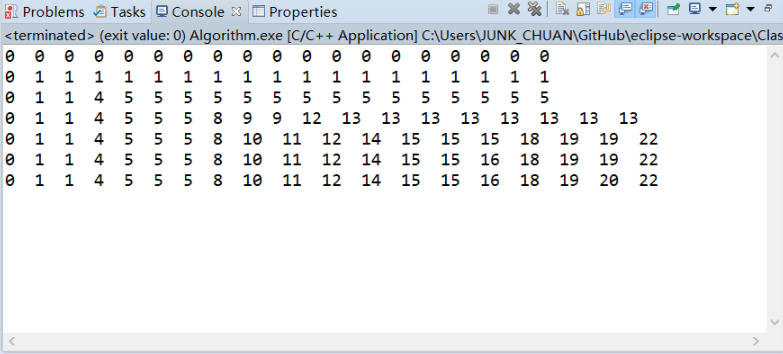
（1）0-1背包问题：设背包重量限制为18，用表格计算最优装法

（2) 无限制背包问题：设背包重量限制为20，用表格计算最优解

设0-1背包问题计算出的表格为M[0..n, 0..W], 编写程序输出n件商品，背包容量为W的最优装法

|  |  |  |
| --- | --- | --- |
| num | weight | value |
| 1 | 1 | 1 |
| 2 | 3 | 4 |
| 3 | 7 | 8 |
| 4 | 8 | 10 |
| 5 | 10 | 11 |
| 6 | 13 | 15 |

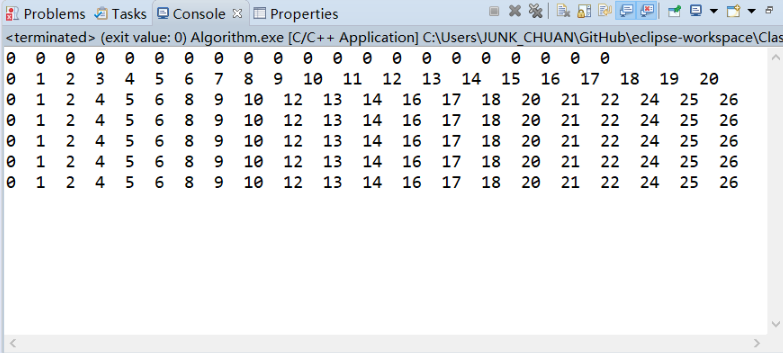
1)



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| {} | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| {1} | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| {1,3} | 0 | 1 | 1 | 4 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| {1,3,7} | 0 | 1 | 1 | 4 | 5 | 5 | 5 | 8 | 9 | 9 | 12 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 |
| {1,3,7,8} | 0 | 1 | 1 | 4 | 5 | 5 | 5 | 8 | 10 | 11 | 12 | 14 | 15 | 15 | 15 | 18 | 19 | 19 | 22 |
| {1,3,7,8,10} | 0 | 1 | 1 | 4 | 5 | 5 | 5 | 8 | 10 | 11 | 12 | 14 | 15 | 15 | 16 | 18 | 19 | 20 | 22 |
| {1,3,7,8,10,13} | 0 | 1 | 1 | 4 | 5 | 5 | 5 | 8 | 10 | 11 | 12 | 14 | 15 | 15 | 16 | 18 | 19 | 20 | 22 |

源代码与第三题类似，故省略。

2)



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| {} | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| {1} | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| {1,3} | 0 | 1 | 2 | 4 | 5 | 6 | 8 | 9 | 10 | 12 | 13 | 14 | 16 | 17 | 18 | 20 | 21 | 22 | 24 | 25 | 26 |
| {1,3,7} | 0 | 1 | 2 | 4 | 5 | 6 | 8 | 9 | 10 | 12 | 13 | 14 | 16 | 17 | 18 | 20 | 21 | 22 | 24 | 25 | 26 |
| {1,3,7,8} | 0 | 1 | 2 | 4 | 5 | 6 | 8 | 9 | 10 | 12 | 13 | 14 | 16 | 17 | 18 | 20 | 21 | 22 | 24 | 25 | 26 |
| {1,3,7,8,10} | 0 | 1 | 2 | 4 | 5 | 6 | 8 | 9 | 10 | 12 | 13 | 14 | 16 | 17 | 18 | 20 | 21 | 22 | 24 | 25 | 26 |
| {1,3,7,8,10,13} | 0 | 1 | 2 | 4 | 5 | 6 | 8 | 9 | 10 | 12 | 13 | 14 | 16 | 17 | 18 | 20 | 21 | 22 | 24 | 25 | 26 |

**源代码：**

**#include** <iostream>

**using** **namespace** std;

**struct** Things{

**int** weight[7] = {0,1,3,7,8,10,13};

**int** value[7] = {0,1,4,8,10,11,15};

**bool** used[7] = {0,0,0,0,0,0,0};

};

**int** **main**(){

Things th;

**int** num = 6;

**int** weight = 20;

**int** Map[num+1][weight+1];

**int** i,j;

**for**(i=0;i<=num;i++) {

Map[i][0] = 0;

}

**for**(j=0;j<=weight;j++) {

Map[0][j] = 0;

}

**for**(i=1;i<=num;i++) {

**for**(j=1;j<=weight;j++) {

**if**(j<th.weight[i]) {

// If beyond weight limits

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

} **else** {

// If within weight limits

**if**(Map[i-1][j]>Map[i][j-th.weight[i]]+th.value[i]) {

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

}

**else** {

Map[i][j]=Map[i][j-th.weight[i]]+th.value[i];

}

}

}

}

// Print

**int** n = 0;

**int** w = 0;

**for**(;n<=num;n++) {

**for**(w = 0;w<=weight;w++) {

cout << Map[n][w] << " ";

}

cout << **endl**;

}

**return** 0;

}

3）

**#include** <iostream>

**using** **namespace** std;

**struct** Things{

**int** weight[7] = {0,1,3,7,8,10,13};

**int** value[7] = {0,1,4,8,10,11,15};

**bool** used[7] = {0,0,0,0,0,0,0};

};

**int** **main**() {

Things th;

**int** num = 6;

**int** weight\_limit = 18;

cin >> num;

cin >> weight\_limit;

**int** Map[num+1][weight\_limit+1];

**int** i,j;

**for**(i=0;i<=num;i++) {

Map[i][0] = 0;

}

**for**(j=0;j<=weight\_limit;j++) {

Map[0][j] = 0;

}

**for**(i=1;i<=num;i++) {

**for**(j=1;j<=weight\_limit;j++) {

**if**(j<th.weight[i]) {

// If beyond weight limits

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

}

**else** {

// If within weight limits

**if**(Map[i-1][j]>Map[i-1][j-th.weight[i]]+th.value[i]) {

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

}

**else** {

Map[i][j]=Map[i-1][j-th.weight[i]]+th.value[i];

}

}

}

}

// Find Path

**for** (i = num; i>=1; i--) {

**if** (Map[i][weight\_limit] == Map[i-1][weight\_limit]) {

th.used[i] = 0;

} **else** **if** (weight\_limit-th.weight[i]>=0 &&

Map[i][weight\_limit] == (Map[i-1][weight\_limit-th.weight[i]]+th.value[i])) {

th.used[i] = 1;

weight\_limit-=th.weight[i];

}

}

// Print Path

**for**(**int** n = 0;n<=num;n++) {

**if** (th.used[n] == 1) {

cout << "ID:" << n << " " << "weight:" << th.weight[n] << " " << "Value:" << th.value[n] << **endl**;

}

}

**return** 0;

}

