

【华为OD机考 统一考试机试C卷】高效货运（C++ Java JavaScript Python C语言）

华为OD机考:统一考试 C卷 + D卷 + B卷 +A卷

目前在考C卷，经过两个月的收集整理，**C卷真题已基本整理完毕**

抽到原题的概率为2/3到3/3，**也就是最少抽到两道原题。请注意：大家刷完C卷真题，最好要把B卷的真题刷一下，因为C卷的部分真题来自B卷。**

另外订阅专栏还可以联系笔者开通在线 **OJ** 进行刷题，提高刷题效率。

真题目录：华为OD机考机试 真题目录（C卷 + D卷 + B卷 + A卷） + 考点说明

专栏：2023华为OD机试(B卷+C卷+D卷) (C++JavaJSPy)

华为OD面试真题精选：华为OD面试真题精选

在线OJ：点击立即刷题，模拟真实机考环境

题目描述

老李是货运公司承运人，老李的货车额定载货重量为 w_t 。

现有两种货物：

- 货物 A 单件重量为 w_a ，单件运费利润为 p_a
- 货物 B 单件重量为 w_b ，单件运费利润为 p_b

老李每次发车时载货总重量刚好为货车额定的载货重量 w_t ，车上必须同时有货物 A 和货物 B，货物A、B不可切割。

老李单次满载运输可获得的最高利润是多少？

输入描述

- 第一列输入为货物 A 的单件重量 w_a ， $0 < w_a < 10000$
- 第二列输入为货物 B 的单件重量 w_b ， $0 < w_b < 10000$
- 第三列输入为货车的额定载重 w_t ， $0 < w_t < 100000$
- 第四列输入为货物 A 的单件运费利润 p_a ， $0 < p_a < 1000$

- 第五列输入为货物 B BB 的单件运费利润 $pb, 0 < pb < 1000$

输出描述

单次满载运输的最高利润

用例1

输入

1 | 10 8 36 15 7

输出

1 | 44

用例2

输入

1 | 1 1 2 1 1

输出

1 | 2

解题思路

暴力模拟：遍历所有可能的货物A的装载数量，对于每种情况计算剩余重量能否完全由货物B填满，如果可以则计算当前组合的利润，并与已知的最大利润进行比较，更新最大利润。

重量计算：对于货物A和B，计算总重量的公式是：

$$\text{总重量} = (\text{货物A的数量} \times \text{货物A的单件重量}) + (\text{货物B的数量} \times \text{货物B的单件重量})$$

利润计算：计算总利润的公式是：

总利润 = (货物A的数量 × 货物A的单件利润) + (货物B的数量 × 货物B的单件利润)

C++

```
1  #include <iostream>
2  #include <sstream>
3  #include <vector>
4
5  // 定义calculateMaxProfit函数, 计算最高利润
6  int calculateMaxProfit(int wa, int wb, int wt, int pa, int pb) {
7      int maxProfit = 0; // 初始化最高利润为0
8
9      // 遍历可能的货物A数量, 确保货物A和B的总重量不超过额定载重
10     for (int countA = 1; countA * wa <= wt - wb; countA++) {
11         // 计算除去货物A后剩余的重量
12         int remainingWeight = wt - countA * wa;
13         // 如果剩余重量可以被货物B的单件重量整除, 说明可以装满货物B
14         if (remainingWeight % wb == 0) {
15             // 计算货物B的数量
16             int countB = remainingWeight / wb;
17             // 计算当前组合的利润
18             int profit = countA * pa + countB * pb;
19             // 如果当前组合的利润大于之前的最高利润, 则更新最高利润
20             if (profit > maxProfit) {
21                 maxProfit = profit;
22             }
23         }
24     }
25     // 返回最高利润
26     return maxProfit;
27 }
28
29 int main() {
30     std::string input;
31     // 从标准输入读取一行
32     std::getline(std::cin, input);
33     std::istringstream iss(input);
34     std::vector<int> values;
35     int value;
```

```
36
37 // 使用istringstream将输入的字符串按空格分割, 并转换为整数数组
38 while (iss >> value) {
39     values.push_back(value);
40 }
41
42 // 从数组中提取各个变量的值
43 int wa = values[0]; // 货物A的单件重量
44 int wb = values[1]; // 货物B的单件重量
45 int wt = values[2]; // 货车的额定载重
46 int pa = values[3]; // 货物A的单件运费利润
47 int pb = values[4]; // 货物B的单件运费利润
48
49 // 调用calculateMaxProfit函数计算最高利润
50 int maxProfit = calculateMaxProfit(wa, wb, wt, pa, pb);
51 // 输出最高利润
52 std::cout << maxProfit << std::endl;
53
54 return 0;
55 }
```

Java

```
1 import java.util.Scanner;
2 import java.util.stream.Stream;
3
4 public class Main {
5     public static void main(String[] args) {
6         Scanner scanner = new Scanner(System.in);
7         String input = scanner.nextLine();
8         // 使用Stream API将输入的字符串按空格分割, 并转换为整数数组
9         int[] values = Stream.of(input.split(" ")) // 将输入的字符串分割成字符串数组
10                                .mapToInt(Integer::parseInt) // 将字符串数组的每个元素转换为整数
11                                .toArray(); // 将流转换为数组
12
13         // 从数组中提取各个变量的值
14         int wa = values[0]; // 货物A的单件重量
15         int wb = values[1]; // 货物B的单件重量
16         int wt = values[2]; // 货车的额定载重
17         int pa = values[3]; // 货物A的单件运费利润
18     }
19 }
```

```
18     int pb = values[4]; // 货物B的单件运费利润
19
20     // 调用calculateMaxProfit方法计算最高利润
21     int maxProfit = calculateMaxProfit(wa, wb, wt, pa, pb);
22     // 输出最高利润
23     System.out.println(maxProfit);
24 }
25
26 // 定义calculateMaxProfit方法, 计算最高利润
27 public static int calculateMaxProfit(int wa, int wb, int wt, int pa, int pb) {
28     int maxProfit = 0; // 初始化最高利润为0
29
30     // 遍历可能的货物A数量, 确保货物A和B的总重量不超过额定载重
31     for (int countA = 1; countA * wa < wt; countA++) {
32         // 计算除去货物A后剩余的重量
33         int remainingWeight = wt - countA * wa;
34         // 如果剩余重量可以被货物B的单件重量整除, 说明可以装满货物B
35         if (remainingWeight % wb == 0) {
36             // 计算货物B的数量
37             int countB = remainingWeight / wb;
38             // 计算当前组合的利润
39             int profit = countA * pa + countB * pb;
40             // 如果当前组合的利润大于之前的最高利润, 则更新最高利润
41             if (profit > maxProfit) {
42                 maxProfit = profit;
43             }
44         }
45     }
46     // 返回最高利润
47     return maxProfit;
48 }
49 }
```

JavaScript

```
1 const readline = require('readline');
2
3 // 创建readline接口实例
4 const rl = readline.createInterface({
5     input: process.stdin,
```

```
6   output: process.stdout
7   });
8
9   // 定义calculateMaxProfit函数, 计算最高利润
10  function calculateMaxProfit(wa, wb, wt, pa, pb) {
11    let maxProfit = 0; // 初始化最高利润为0
12
13    // 遍历可能的货物A数量, 确保货物A和B的总重量不超过额定载重
14    for (let countA = 1; countA * wa < wt; countA++) {
15      // 计算除去货物A后剩余的重量
16      let remainingWeight = wt - countA * wa;
17      // 如果剩余重量可以被货物B的单件重量整除, 说明可以装满货物B
18      if (remainingWeight % wb === 0) {
19        // 计算货物B的数量
20        let countB = remainingWeight / wb;
21        // 计算当前组合的利润
22        let profit = countA * pa + countB * pb;
23        // 如果当前组合的利润大于之前的最高利润, 则更新最高利润
24        maxProfit = Math.max(maxProfit, profit);
25      }
26    }
27    // 返回最高利润
28    return maxProfit;
29  }
30
31  // 主函数
32  rl.on('line', (input) => {
33    // 使用split方法将输入的字符串按空格分割, 并转换为整数数组
34    const values = input.split(' ').map(Number);
35
36    // 从数组中提取各个变量的值
37    const [wa, wb, wt, pa, pb] = values;
38
39    // 调用calculateMaxProfit函数计算最高利润
40    const maxProfit = calculateMaxProfit(wa, wb, wt, pa, pb);
41
42    // 输出最高利润
43    console.log(maxProfit);
44
45    // 关闭readLine接口实例
46    --
```

```
47 | r1.close();  
    | });
```

Python

```
1 | import sys  
2 |  
3 | def calculate_max_profit(wa, wb, wt, pa, pb):  
4 |     max_profit = 0  
5 |  
6 |     # 遍历可能的货物A数量  
7 |     for count_a in range(1, (wt - wb) // wa + 1):  
8 |         remaining_weight = wt - count_a * wa  
9 |  
10 |        # 计算在剩余重量下, 最多可以装载多少货物B  
11 |        count_b = remaining_weight // wb  
12 |  
13 |        # 计算当前组合的利润  
14 |        profit = count_a * pa + count_b * pb  
15 |  
16 |        # 更新最高利润  
17 |        max_profit = max(max_profit, profit)  
18 |  
19 |    return max_profit  
20 |  
21 | def main():  
22 |     input_str = sys.stdin.readline().strip()  
23 |     values = list(map(int, input_str.split()))  
24 |  
25 |     wa, wb, wt, pa, pb = values  
26 |     max_profit = calculate_max_profit(wa, wb, wt, pa, pb)  
27 |  
28 |     print(max_profit)  
29 |  
30 | if __name__ == "__main__":  
31 |     main()
```

C语言


```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5 // 定义calculateMaxProfit函数, 计算最高利润
6 int calculateMaxProfit(int wa, int wb, int wt, int pa, int pb) {
7     int maxProfit = 0; // 初始化最高利润为0
8
9     // 遍历可能的货物A数量, 确保货物A和B的总重量不超过额定载重
10    for (int countA = 1; countA * wa <= wt - wb; countA++) {
11        int remainingWeight = wt - countA * wa; // 计算除去货物A后剩余的重量
12        if (remainingWeight % wb == 0) { // 如果剩余重量可以被货物B的单件重量整除
13            int countB = remainingWeight / wb; // 计算货物B的数量
14            int profit = countA * pa + countB * pb; // 计算当前组合的利润
15            if (profit > maxProfit) { // 更新最高利润
16                maxProfit = profit;
17            }
18        }
19    }
20    return maxProfit; // 返回最高利润
21 }
22
23 int main() {
24     int wa, wb, wt, pa, pb; // 定义相关变量
25     // 从标准输入读取变量值
26     scanf("%d %d %d %d %d", &wa, &wb, &wt, &pa, &pb);
27
28     // 调用calculateMaxProfit函数计算最高利润
29     int maxProfit = calculateMaxProfit(wa, wb, wt, pa, pb);
30     printf("%d\n", maxProfit); // 输出最高利润
31
32     return 0;
33 }
```

文章目录

华为OD机考:统一考试 C卷 + D卷 + B卷 +A卷

题目描述

输入描述

输出描述

用例1

用例2

解题思路

C++

Java

JavaScript

Python

C语言

机考真题 华为OD



CSDN @算法大师