【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 wt。

现有两种货物：

- 货物 A 单件重量为 wa，单件运费利润为 pa
- 货物 B 单件重量为 wb，单件运费利润为 pb

老李每次发车时载货总重量刚好为货车额定的载货重量 wt，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 AA 的单件重量 wa ， 0 < wa < 10000
- 第二列输入为货物 B BB 的单件重量wb,0 < wb < 10000
- 第三列输入为货车的额定载重wt,0 < wt < 100000
- 第四列输入为货物 A AA 的单件运费利润pa,0 < pa < 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

```
10 8 36 15 7
```

输出

```
44
```

用例2

输入

```
1 1 2 1 1
```

输出

```
2
```

解题思路

暴击模拟：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：

$$总利润 = (货物A的数量 \times 货物A的单件利润) + (货物B的数量 \times 货物B的单件利润)$$

C++

```cpp
#include <iostream>
#include <sstream>
#include <vector>

// 定义calculateMaxProfit函数，计算最高利润
int calculateMaxProfit(int wa, int wb, int wt, int pa, int pb) {
    int maxProfit = 0; // 初始化最高利润为0

    // 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
    for (int countA = 1; countA * wa <= wt - wb; countA++) {
        // 计算除去货物A后剩余的重量
        int remainingWeight = wt - countA * wa;
        // 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
        if (remainingWeight % wb == 0) {
            // 计算货物B的数量
            int countB = remainingWeight / wb;
            // 计算当前组合的利润
            int profit = countA * pa + countB * pb;
            // 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
            if (profit > maxProfit) {
                maxProfit = profit;
            }
        }
    }
    // 返回最高利润
    return maxProfit;
}

int main() {
    std::string input;
    // 从标准输入读取一行
    std::getline(std::cin, input);
    std::istringstream iss(input);
    std::vector<int> values;
    int value;
```

```
    // 使用istringstream将输入的字符串按空格分割，并转换为整数数组
    while (iss >> value) {
        values.push_back(value);
    }

    // 从数组中提取各个变量的值
    int wa = values[0]; // 货物A的单件重量
    int wb = values[1]; // 货物B的单件重量
    int wt = values[2]; // 货车的额定载重
    int pa = values[3]; // 货物A的单件运费利润
    int pb = values[4]; // 货物B的单件运费利润

    // 调用calculateMaxProfit函数计算最高利润
    int maxProfit = calculateMaxProfit(wa, wb, wt, pa, pb);
    // 输出最高利润
    std::cout << maxProfit << std::endl;

    return 0;
}
```

Java

```
import java.util.Scanner;
import java.util.stream.Stream;

public class Main {
    public static void main(String[] args) {
         Scanner scanner = new Scanner(System.in);
         String input = scanner.nextLine();
        // 使用Stream API将输入的字符串按空格分割，并转换为整数数组
        int[] values = Stream.of(input.split(" ")) // 将输入的字符串分割成字符串数组
                             .mapToInt(Integer::parseInt) // 将字符串数组的每个元素转换为整数
                             .toArray(); // 将流转换为数组

        // 从数组中提取各个变量的值
        int wa = values[0]; // 货物A的单件重量
        int wb = values[1]; // 货物B的单件重量
        int wt = values[2]; // 货车的额定载重
        int pa = values[3]; // 货物A的单件运费利润
```

```
        int pb = values[4]; // 货物B的单件运费利润

        // 调用calculateMaxProfit方法计算最高利润
        int maxProfit = calculateMaxProfit(wa, wb, wt, pa, pb);
        // 输出最高利润
        System.out.println(maxProfit);
    }

    // 定义calculateMaxProfit方法，计算最高利润
    public static int calculateMaxProfit(int wa, int wb, int wt, int pa, int pb) {
        int maxProfit = 0; // 初始化最高利润为0

        // 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
        for (int countA = 1; countA * wa < wt; countA++) {
            // 计算除去货物A后剩余的重量
            int remainingWeight = wt - countA * wa;
            // 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
            if (remainingWeight % wb == 0) {
                // 计算货物B的数量
                int countB = remainingWeight / wb;
                // 计算当前组合的利润
                int profit = countA * pa + countB * pb;
                // 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
                if (profit > maxProfit) {
                    maxProfit = profit;
                }
            }
        }
        // 返回最高利润
        return maxProfit;
    }
}
```

javaScript

```
const readline = require('readline');

// 创建readline接口实例
const rl = readline.createInterface({
  input: process.stdin,
```

```
  output: process.stdout
});

// 定义calculateMaxProfit函数，计算最高利润
function calculateMaxProfit(wa, wb, wt, pa, pb) {
  let maxProfit = 0; // 初始化最高利润为0

  // 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
  for (let countA = 1; countA * wa < wt; countA++) {
    // 计算除去货物A后剩余的重量
    let remainingWeight = wt - countA * wa;
    // 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
    if (remainingWeight % wb === 0) {
      // 计算货物B的数量
      let countB = remainingWeight / wb;
      // 计算当前组合的利润
      let profit = countA * pa + countB * pb;
      // 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
      maxProfit = Math.max(maxProfit, profit);
    }
  }
  // 返回最高利润
  return maxProfit;
}

// 主函数
rl.on('line', (input) => {
  // 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
  const values = input.split(' ').map(Number);

  // 从数组中提取各个变量的值
  const [wa, wb, wt, pa, pb] = values;

  // 调用calculateMaxProfit函数计算最高利润
  const maxProfit = calculateMaxProfit(wa, wb, wt, pa, pb);

  // 输出最高利润
  console.log(maxProfit);

  // 关闭readline接口实例
```

```
  rl.close();
});
```

Python

```python
import sys

def calculate_max_profit(wa, wb, wt, pa, pb):
    max_profit = 0

    # 遍历可能的货物A数量
    for count_a in range(1,(wt - wb) // wa + 1):
        remaining_weight = wt - count_a * wa

        # 计算在剩余重量下，最多可以装载多少货物B
        count_b = remaining_weight // wb

        # 计算当前组合的利润
        profit = count_a * pa + count_b * pb

        # 更新最高利润
        max_profit = max(max_profit, profit)

    return max_profit

def main():
    input_str = sys.stdin.readline().strip()
    values = list(map(int, input_str.split()))

    wa, wb, wt, pa, pb = values
    max_profit = calculate_max_profit(wa, wb, wt, pa, pb)

    print(max_profit)

if __name__ == "__main__":
    main()
```

C语言

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// 定义calculateMaxProfit函数，计算最高利润
int calculateMaxProfit(int wa, int wb, int wt, int pa, int pb) {
    int maxProfit = 0; // 初始化最高利润为0

    // 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
    for (int countA = 1; countA * wa <= wt - wb; countA++) {
        int remainingWeight = wt - countA * wa; // 计算除去货物A后剩余的重量
        if (remainingWeight % wb == 0) { // 如果剩余重量可以被货物B的单件重量整除
            int countB = remainingWeight / wb; // 计算货物B的数量
            int profit = countA * pa + countB * pb; // 计算当前组合的利润
            if (profit > maxProfit) { // 更新最高利润
                maxProfit = profit;
            }
        }
    }
    return maxProfit; // 返回最高利润
}

int main() {
    int wa, wb, wt, pa, pb; // 定义相关变量
    // 从标准输入读取变量值
    scanf("%d %d %d %d %d", &wa, &wb, &wt, &pa, &pb);

    // 调用calculateMaxProfit函数计算最高利润
    int maxProfit = calculateMaxProfit(wa, wb, wt, pa, pb);
    printf("%d\n", maxProfit); // 输出最高利润

    return 0;
}
```

文章目录