题目描述

商人经营一家店铺，有number种商品，
由于仓库限制每件商品的最大持有数量是item[index]
每种商品的价格是item-price[item_index][day]
通过对商品的买进和卖出获取利润
请给出商人在days天内能获取的最大的利润
注：同一件商品可以反复买进和卖出

输入描述

第一行输入商品的数量number，比如3
第二行输入商品售货天数 days，比如3
第三行输入仓库限制每件商品的最大持有数量是item[index]，比如4 5 6

后面继续输入number行days列，含义如下：
第一件商品每天的价格，比如1 2 3
第二件商品每天的价格，比如4 3 2
第三件商品每天的价格，比如1 5 3

输出描述

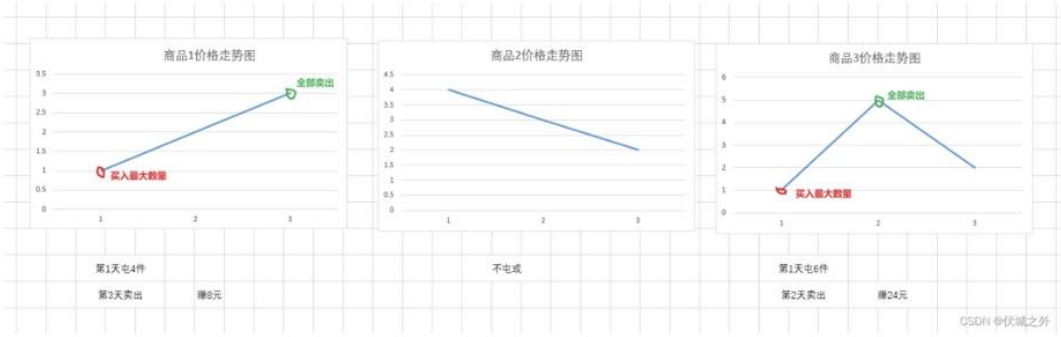
输出商人在这段时间内的最大利润。

用例

输入	3 3 4 5 6 1 2 3 4 3 2 1 5 2
输出	32
说明	无

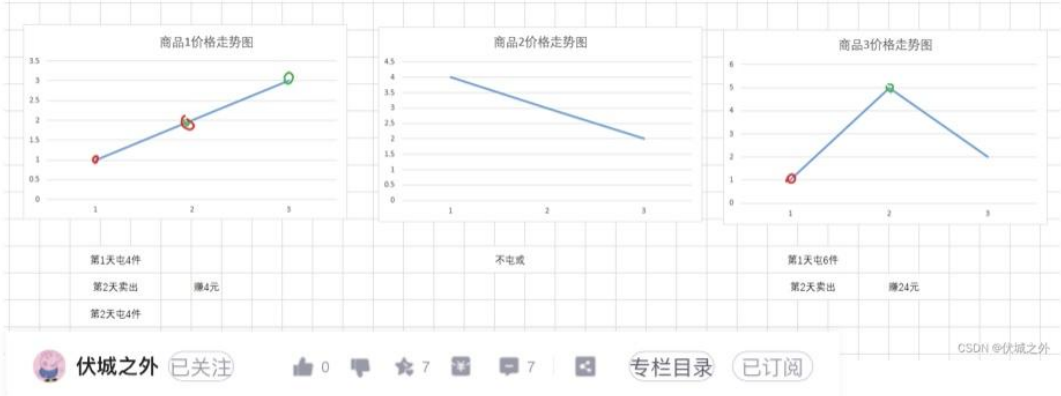
题目解析

用例含义如下所示



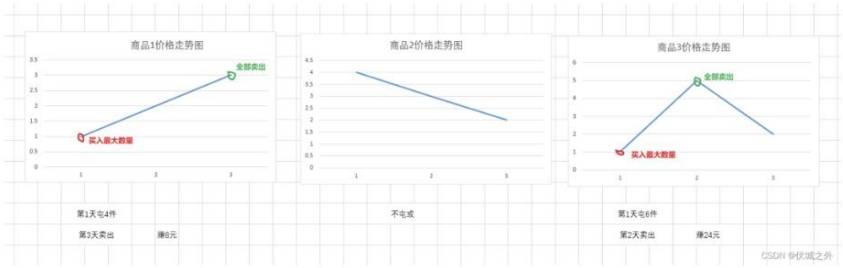
因此最多可以赚8 + 24 = 32元。

另外，本题描述说：同一件商品可以反复买进和卖出，因此上面图示买卖操作还可以变为这样：



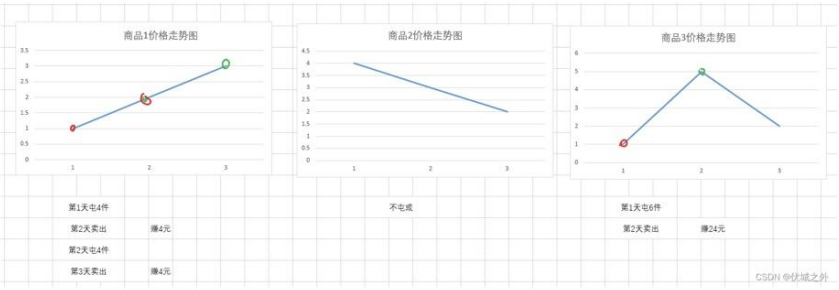
题目解析

用例含义如下所示



因此最多可以赚8 + 24 = 32元。

另外，本题描述说：同一件商品可以反复买进和卖出，因此上面图示买卖操作还可以变为这样：



本题算是华为机试 - 最大股票收益_伏城之外的博客-CSDN博客 的变种题。

我们只要找到商品价格走势的上升区段，然后低价all in买入，高价all out卖出即可求得最大利润。

那么如何找到上升区段呢？

我们假设商品1的第 i 天的价格为 price1[i]，那么只要price1[i] < price1[i+1]，则说明当前处于上升区段的低价位，因此可以all in，然后到i+1天的时候all out。

JavaScript算法源码

> 根据网友20221207的优化补充，下面代码中24~26行会将代码正确率从40%提高到70%

```
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

const lines = [];
let number, days, maxCount;
rl.on("line", (line) => {
  lines.push(line);

  if (lines.length === 3) {
    number = lines[0] - 0;
    days = lines[1] - 0;
    maxCount = lines[2].split(" ").map(Number);
  }

  if (number && lines.length === number + 3) {
    const prices = lines.slice(3).map((line) => line.split(" ").map(Number));
    console.log(getResult(number, days, maxCount, prices));
    lines.length = 0;
  } else {
    console.log("0"); // 感谢网友20221207的提供的优化思路，根据机试结果，加上此段else输出0的逻辑，可以将正确率从40%提高到
  }
});

/**
 *
 * @param {*} number 几种商品
 * @param {*} days 几天
 * @param {*} maxCount 每种商品的最大囤货数量
 * @param {*} prices 每种商品的在days天内的价格变动情况
 */
function getResult(number, days, maxCount, prices) {
  let ans = 0;
  for (let i = 0; i < number; i++) {
    const price = prices[i];
    for (let j = 0; j < days - 1; j++) {
      if (price[j] < price[j + 1]) {
```

JavaScript算法源码

> 根据网友20221207的优化补充，下面代码中24~26行会将代码正确率从40%提高到70%

```javascript
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

const lines = [];
let number, days, maxCount;
rl.on("line", (line) => {
  lines.push(line);

  if (lines.length === 3) {
    number = lines[0] - 0;
    days = lines[1] - 0;
    maxCount = lines[2].split(" ").map(Number);
  }

  if (number && lines.length === number + 3) {
    const prices = lines.slice(3).map((line) => line.split(" ").map(Number));
    console.log(getResult(number, days, maxCount, prices));
    lines.length = 0;
  } else {
    console.log("0"); // 感谢网友20221207的提供的优化思路，根据机试结果，加上此段else输出0的逻辑，可以将正确率从40%提高到
  }
});

/**
 *
 * @param {*} number 几种商品
 * @param {*} days 几天
 * @param {*} maxCount 每种商品的最大囤货数量
 * @param {*} prices 每种商品的在days天内的价格变动情况
 */
function getResult(number, days, maxCount, prices) {
  let ans = 0;
  for (let i = 0; i < number; i++) {
    const price = prices[i];
    for (let j = 0; j < days - 1; j++) {
      if (price[j] < price[j + 1]) {
        ans += (price[j + 1] - price[j]) * maxCount[i];
      }
    }
  }
  return ans;
}
```

Java算法源码

```java
import java.util.Scanner;

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    try { // 此段try...catch不影响正常业务逻辑，只是有网友说可能存在如输入行数校验要返回0，因此加了
      int number = sc.nextInt();
      int days = sc.nextInt();

      int[] maxCount = new int[number];
      for (int i = 0; i < number; i++) {
        maxCount[i] = sc.nextInt();
      }

      int[][] prices = new int[number][days];
      for (int i = 0; i < number; i++) {
        for (int j = 0; j < days; j++) {
          prices[i][j] = sc.nextInt();
        }
      }

      System.out.println(getResult(number, days, maxCount, prices));
    } catch (Exception e) {
      System.out.println(0);
    }
  }

  /**
   * @param number 几种商品
   * @param days 几天
   * @param maxCount 每种商品的最大囤货数量
   * @param prices 每种商品的在days天内的价格变动情况
   * @return 最大利润
   */
  public static int getResult(int number, int days, int[] maxCount, int[][] prices) {
    int ans = 0;
```

Java算法源码

```java
import java.util.Scanner;

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    try { // 此段try...catch不影响正常业务逻辑，只是有网友说可能存在如输入行数校验要返回0，因此加了
      int number = sc.nextInt();
      int days = sc.nextInt();

      int[] maxCount = new int[number];
      for (int i = 0; i < number; i++) {
        maxCount[i] = sc.nextInt();
      }

      int[][] prices = new int[number][days];
      for (int i = 0; i < number; i++) {
        for (int j = 0; j < days; j++) {
          prices[i][j] = sc.nextInt();
        }
      }

      System.out.println(getResult(number, days, maxCount, prices));
    } catch (Exception e) {
      System.out.println(0);
    }
  }

  /**
   * @param number 几种商品
   * @param days 几天
   * @param maxCount 每种商品的最大囤货数量
   * @param prices 每种商品的在days天内的价格变动情况
   * @return 最大利润
   */
  public static int getResult(int number, int days, int[] maxCount, int[][] prices) {
    int ans = 0;
    for (int i = 0; i < number; i++) {
      int[] price = prices[i];
      for (int j = 0; j < days - 1; j++) {
        if (price[j] < price[j + 1]) {
          ans += (price[j + 1] - price[j]) * maxCount[i];
        }
      }
    }
    return ans;
  }
}
```

Python算法源码

```python
# 输入获取
number = int(input())
days = int(input())
item = list(map(int, input().split()))
prices = [list(map(int, input().split())) for i in range(number)]


# 算法入口
def getResult(number, days, item, prices):
    """
    :param number: 几种商品
    :param days: 几天
    :param item: 每种商品的最大囤货数量
    :param prices: 每种商品的在days天内的价格变动情况
    :return: 最大利润
    """
    ans = 0
    for i in range(number):
        price = prices[i]
        for j in range(days - 1):
            if price[j] < price[j + 1]:
                ans += (price[j + 1] - price[j]) * item[i]
    return ans


# 算法调用
print(getResult(number, days, item, prices))
```