34、书籍叠放，考点 or 实现——耐心排序+二分查找

题目描述

书籍的长、宽都是整数对应 (l,w)。如果书A的长宽度都比B长宽大时，则允许将B排列放在A上面。现在有一组规格的书籍，书籍叠放时要求书籍不能做旋转，请计算最多能有多少个规格书籍能叠放在一起。

输入描述

输入：books = [[20,16],[15,11],[10,10],[9,10]]

说明：总共4本书籍，第一本长度为20宽度为16；第二本书长度为15宽度为11，依次类推，最后一本书长度为9宽度为10.

输出描述

输出：3

说明: 最多3个规格的书籍可以叠放到一起, 从下到上依次为: [20,16],[15,11],[10,10]

用例

输入	[[20,16],[15,11],[10,10],[9,10]]
输出	3

题目解析

可以采用耐心排序+二分查找，实现O(nlgn)时间复杂度的算法。

Java算法源码

```java
import java.util.*;

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

    String input = sc.nextLine();

    // (?<=]),(?=\[) 正则表达式含义是：找这样一个逗号，前面跟着]，后面跟着[
    // 其中(?<=) 表示前面跟着
    // 其中(?=) 表示后面跟着
    Integer[][] books =
        Arrays.stream(input.substring(1, input.length() - 1).split("(?<=]),(?=\\[)"))
            .map(
                s ->
                    Arrays.stream(s.substring(1, s.length() - 1).split(","))
                        .map(Integer::parseInt)
                        .toArray(Integer[]::new))
            .toArray(Integer[][]::new);

    System.out.println(getResult(books));
  }

  public static int getResult(Integer[][] books) {
    // 长度升序，若长度相同，则宽度降序
    Arrays.sort(books, (a, b) -> Objects.equals(a[0], b[0]) ? b[1] - a[1] : a[0] - b[0]);
    Integer[] widths = Arrays.stream(books).map(book -> book[1]).toArray(Integer[]::new);
```

```java
    return getMaxLIS(widths);
  }

  // 最长递增子序列
  public static int getMaxLIS(Integer[] nums) {
    //  dp数组元素dp[i]含义是：长度为i+1的最优子序列的尾数
    ArrayList<Integer> dp = new ArrayList<>();
    dp.add(nums[0]);

    for (int i = 1; i < nums.length; i++) {
      if (nums[i] > dp.get(dp.size() - 1)) {
        dp.add(nums[i]);
        continue;
      }

      if (nums[i] < dp.get(0)) {
        dp.set(0, nums[i]);
        continue;
      }

      int idx = Collections.binarySearch(dp, nums[i]);
      if (idx < 0) dp.set(-idx - 1, nums[i]);
    }

    return dp.size();
  }
}
```

JavaScript算法源码

```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,
});

rl.on("line", (line) => {
  const books = JSON.parse(line);
  console.log(getMaxStackCount(books));
});

function getMaxStackCount(books) {
  // 长度升序，若长度相同，则宽度降序
  const widths = books
    .sort((a, b) => (a[0] === b[0] ? b[1] - a[1] : a[0] - b[0]))
    .map((book) => book[1]);

  return getMaxLIS(widths);
}

// 最长递增子序列
function getMaxLIS(nums) {
  // dp数组元素dp[i]含义是：长度为i+1的最优子序列的尾数
  const dp = [nums[0]];

  for (let i = 1; i < nums.length; i++) {
    if (nums[i] > dp[dp.length - 1]) {
      dp.push(nums[i]);
      continue;
    }

    if (nums[i] < dp[0]) {
      dp[0] = nums[i];
      continue;
    }

    const idx = binarySearch(dp, nums[i]);
    if (idx < 0) dp[-idx - 1] = nums[i];
  }

```

```
  return dp.length;
}

// 二分查找
function binarySearch(arr, key) {
  let low = 0;
  let high = arr.length - 1;

  while (low <= high) {
    let mid = (low + high) >>> 1;
    let midVal = arr[mid];

    if (key > midVal) {
      low = mid + 1;
    } else if (key < midVal) {
      high = mid - 1;
    } else {
      return mid;
    }
  }
  return -(low + 1);
}
```

Python算法源码

```python
# 输入获取
books = eval(input())


# 二分查找
def binarySearch(arr, key):
    low = 0
    high = len(arr) - 1

    while low <= high:
        mid = (low + high) >> 1
        midVal = arr[mid]

        if key > midVal:
            low = mid + 1
        elif key < midVal:
            high = mid - 1
        else:
            return mid

    return -(low + 1)


# 最长递增子序列
def getMaxLIS(nums):
    # dp数组元素dp[i]含义是：长度为i+1的最优子序列的尾数
    dp = [nums[0]]

    for i in range(1, len(nums)):
        if nums[i] > dp[-1]:
            dp.append(nums[i])
            continue

        if nums[i] < dp[0]:
            dp[0] = nums[i]
            continue

        idx = binarySearch(dp, nums[i])
        if idx < 0:
            dp[-idx - 1] = nums[i]

    return len(dp)
```

```python


# 算法入口
def getResult(books):
    # 长度升序，若长度相同，则宽度降序
    books.sort(key=lambda x: (x[0], -x[1]))
    widths = list(map(lambda x: x[1], books))
    return getMaxLIS(widths)


# 算法调用
print(getResult(books))
```