Saddleback Search（鞍背搜索）是什么?

鞍背搜索（Saddleback Search）是一种针对**二维有序矩阵**的高效搜索算法，用于快速定位目标值的位置。它得名于矩阵中“鞍背”（类似马鞍形状）的搜索路径，结合了**分治策略**和**双指针技巧**，在特定条件下时间复杂度可优化至 **O(m + n)**（m、n为矩阵的行列数）。

核心思想

矩阵特性：

矩阵需满足**行列双重有序**，即：

每行从左到右**非递减**。

每列从上到下**非递减**。

示例矩阵：

复制 下载

```
[ [1,  4,  7, 11],
  [2,  5,  8, 12],
  [3,  6,  9, 16],
  [10, 13, 14, 17] ]
```

搜索起点：

从矩阵的**右上角**（或左下角）开始，利用行列有序性逐步缩小搜索范围：

若当前元素 **> 目标值**：目标不可能在当前列，**向左移动一列**。

若当前元素 **< 目标值**：目标不可能在当前行，**向下移动一行**。

若相等，则返回位置；若越界则说明目标不存在。

算法步骤（以右上角为例）

初始化指针 `(i, j)` 为 `(0, n-1)`（n为列数）。

循环比较 `matrix[i][j]` 与目标值 `target`：

- `matrix[i][j] == target`：返回 `(i, j)`。
- `matrix[i][j] > target`：`j--`（左移）。
- `matrix[i][j] < target`：`i++`（下移）。

若 `i ≥ m` 或 `j < 0`，终止并返回未找到。

代码示例（Python）

python 复制 下载

```python
def saddleback_search(matrix, target):
    if not matrix:
        return (-1, -1)
    m, n = len(matrix), len(matrix[0])
    i, j = 0, n - 1  # 从右上角开始

    while i < m and j >= 0:
        if matrix[i][j] == target:
            return (i, j)
        elif matrix[i][j] > target:
            j -= 1  # 左移
        else:
            i += 1  # 下移
    return (-1, -1)  # 未找到
```

复杂度分析

时间复杂度：**O(m + n)**

最坏情况下需遍历矩阵的一行和一列（如从右上角移动到左下角）。

空间复杂度：**O(1)**

仅需常数空间存储指针。

适用场景

行列有序的二维矩阵（如Young tableau）。

替代暴力搜索（O(mn)）或二分搜索的嵌套（O(m log n)），在行列数接近时更优。

局限性

若矩阵无序，需先排序（排序成本可能较高）。

对于极稀疏矩阵，可能存在更优方法。

通过利用数据的局部有序性，鞍背搜索在二维搜索中实现了高效的剪枝，是经典的空间换时间策略的体现。