24暑期基础培训第二讲

林凯

2024 年 7 月 10 日

冒泡排序

冒泡排序

- **算法描述**：冒泡排序是一种简单的排序算法。它重复地走访过要排序的数列，一次比较两个元素，如果它们的顺序错误就把它们交换过来。走访数列的工作是重复进行的，直到没有再需要交换，也就是说该数列已经排序完成。
- **时间复杂度**：$O(n^2)$，因为有两个嵌套的循环。
- **空间复杂度**：$O(1)$，因为只需要一个额外的交换空间。
- **稳定性**：稳定，冒泡排序不会改变相同元素的相对位置。

直接插入排序

直接插入排序

- **算法描述**：插入排序的工作原理是通过构建有序序列，对于未排序数据，在已排序序列中从后向前扫描，找到相应位置并插入。插入排序在实现上，通常采用in-place排序（即只需用到$O(1)$的额外空间的排序），因而在从后向前扫描过程中，需要反复把已排序元素逐步向后挪位，为最新元素提供插入空间。
- **时间复杂度**：$O(n^2)$，最坏情况下需要两层循环。
- **空间复杂度**：$O(1)$，只需要常数级别的额外空间。
- **稳定性**：稳定，插入排序不会改变相同元素的相对位置。

希尔排序

希尔排序

- **算法描述**：希尔排序，也称递减增量排序算法，是插入排序的一种更高效的改进版本。希尔排序是非稳定排序算法。希尔排序通过将原序列按下标的一定增量分组，对每组使用直接插入排序算法排序；随着增量逐渐减少，每组包含的关键词越来越多，当增量减至1时，整个序列恰好被分成一组，算法便终止。
- **时间复杂度**：最坏情况 $O(n^2)$，最优情况 $O(n \log^2 n)$。
- **空间复杂度**：$O(1)$，只需常数空间。
- **稳定性**：不稳定，排序过程中可能改变相同元素的相对位置。

选择排序

选择排序

- **算法描述**：选择排序是一种简单直观的排序算法。它的工作原理如下。首先在未排序序列中找到最小（大）元素，存放到排序序列的起始位置，然后，再从剩余未排序元素中继续寻找最小（大）元素，然后放到已排序序列的末尾。以此类推，直到所有元素均排序完毕。
- **时间复杂度**：$O(n^2)$，因为有两个嵌套的循环。
- **空间复杂度**：$O(1)$，只需常数空间。
- **稳定性**：不稳定，选择排序可能改变相同元素的相对位置。

快速排序

快速排序

- **算法描述**：快速排序的基本思想是：通过一趟排序将待排记录分隔成独立的两部分，其中一部分的所有记录均比另一部分的所有记录小，然后分别对这两部分继续进行排序，直到整个序列有序。
- **时间复杂度**：最坏情况 $O(n^2)$，平均 $O(n \log n)$。
- **空间复杂度**：$O(\log n)$，由于递归栈的深度。
- **稳定性**：不稳定，排序过程中可能改变相同元素的相对位置。

归并排序

归并排序

- **算法描述**：归并排序是采用分治法的一个非常典型的应用。归并排序的思想就是先递归地将序列划分为两部分，再将排序好的子序列合并在一起。
- **时间复杂度**：$O(n \log n)$，因为每次划分后合并所需时间为线性级别。
- **空间复杂度**：$O(n)$，需要与原数组同样大小的辅助空间。
- **稳定性**：稳定，归并排序不会改变相同元素的相对位置。

堆的基本操作

- **插入一个数**：将新元素插入堆尾，然后向上调整堆以保持堆的性质。
- **求集合当中的最小值或最大值**：最小堆的堆顶元素即为最小值，最大堆的堆顶元素即为最大值。
- **删除最小值或最大值**：将堆顶元素与堆尾元素交换，删除堆尾元素，然后向下调整堆以保持堆的性质。
- **删除任意一个元素**：将要删除的元素与堆尾元素交换，删除堆尾元素，然后根据情况向上或向下调整堆。
- **修改任意一个元素**：将要修改的元素进行修改，然后根据情况向上或向下调整堆。

堆的实现

- **最小堆**：每个节点都小于等于其子节点。插入操作时，将新元素加到堆的末尾，然后向上调整。删除操作时，将堆顶元素与堆的末尾元素交换，然后删除堆的末尾元素，再向下调整堆顶元素。
- **最大堆**：每个节点都大于等于其子节点。插入操作与最小堆相同。删除操作也与最小堆相同，只是调整方向相反。

堆的应用

- **优先队列**：堆是实现优先队列的一种常见数据结构。在优先队列中，每次操作都是针对优先级最高的元素，堆可以高效地实现这种操作。

基本操作

- **top**：访问队头元素，即优先级最高的元素。
- **empty**：判断队列是否为空。
- **size**：返回队列中元素的个数。
- **push**：插入新元素到队列中，并根据优先级进行排序。
- **emplace**：原地构造一个元素并插入队列。
- **pop**：移除队头元素。
- **swap**：交换两个优先队列的内容。

优先队列的实现

- **基于堆的实现**：优先队列通常使用二叉堆来实现。二叉堆可以在对数时间内完成插入、删除和访问优先级最高元素的操作。

手写二分查找

- **算法描述**：二分查找也称折半查找，其基本思想是在一个有序数组中，通过不断地将查找范围减半来定位目标值。每次将查找范围缩小为之前的一半，直到找到目标值或者范围为空。
- **时间复杂度**：$O(\log n)$，每次查找范围减半。
- **空间复杂度**：$O(1)$，只需要常数空间。

二分答案

- **应用场景**：二分答案通常用于解决在一个连续的数值区间内查找满足某个条件的最优值的问题。例如，寻找某个最小值或最大值，使得某个函数在这个值上达到特定条件。

STL中的二分查找

- **lower_bound()**：返回大于或等于目标值的第一个位置。
- **upper_bound()**：返回大于目标值的第一个位置。
- **binary_search()**：检查目标值是否存在于序列中。