Chapter 1

排序算法

1.1 Quick Sort

1.1.1 性能

时间复杂度

Average	Worst
$O(n \cdot \log n)$	$O(n^2)$

Worst case: In the most unbalanced case, a single Quicksort call involves $O(n)$ work plus two recursive calls on lists of size 0 and $n-1$, so the recurrence relation is:

$$\begin{aligned} T(n) &= O(n) + T(0) + T(n-1) \\ &= O(n) + T(n-1) \\ &= O(n^2) \end{aligned} \tag{1.1}$$

Average: In the most balanced case, a single quicksort call involves $O(n)$ work plus two recursive calls on lists of size $n/2$, so the recurrence relation is:

$$\begin{aligned} T(n) &= O(n) + 2T(\frac{n}{2}) \\ &= O(n \log n) \end{aligned} \tag{1.2}$$

空间复杂度

@TODO

1.1.2 实现

```
1  int partition(vector<int>& data, int low, int high)
2  {
3    int pivot = data[high];
4    int small = low - 1;
5    for (int i = low; i < high; ++i)
6    {
7      if (data[i] < data[high])
8      {
9        small++;
10       if (small != i)
11         swap(data[i], data[small]);
12     }
13   }
14   ++small;
15   swap(data[high], data[small]);
16   return small;
17 }
18
19 void quicksort(vector<int>& data, int low, int high)
20 {
21   if (low < high)
22   {
23     int k = partition(data, low, high);
24     quicksort(data, low, k-1);
25     quicksort(data, k+1, high);
26   }
27 }
```

Listing 1.1: 算法导论中的实现

1.2 Merge Sort

1.2.1 实现

1.3 Heap Sort

Chapter 2

搜索算法

2.1 Binary Search

2.1.1 实现

```
1  int bsearch(const vector<int>& data, int key)
2  {
3    int low = 0;
4    int high = data.size() - 1;
5    while (low <= high)
6    {
7      int mid = low + ((high - low) >> 1);
8      if (key < data[mid])
9        high = mid - 1;
10     else if (key > data[mid])
11       low = mid + 1;
12     else
13       return mid;
14   }
15   return -1;
16 }
```

Listing 2.1: Iterative Implementation

```
1  int bsearch(vector<int>& data, int key, int low, int high)
2  {
```

```
3   if (low > high)
4     return -1;
5
6   int mid = low + ((high - low) >> 1);
7   if (key < data[mid])
8     binary_search(data, key, low, mid-1);
9   else if (key > data[mid])
10    binary_search(data, key, mid+1, high);
11  else
12    return mid;
13 }
```

Listing 2.2: Recursive Implementation

Chapter 3

数组相关问题

Chapter 4

链表相关问题

Chapter 5

二叉树相关问题