选择排序 堆排序

2019年6月29日 20:27

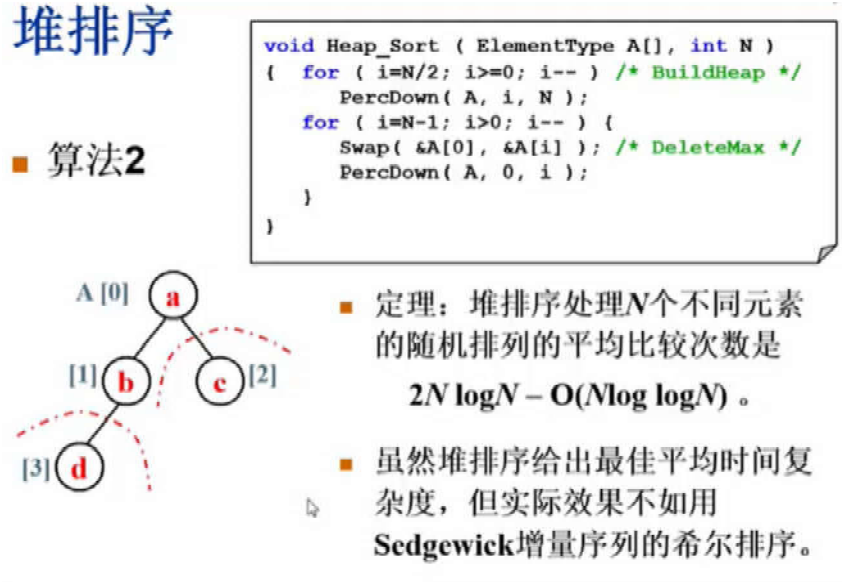
选择排序

```
void Selection_Sort(ElementType A[], int N)
{
	for (i = 0; i < N; i++) {
		MinPosition = ScanForMin(A, i, N–1);
		/* 从A[i]到A[N–1]中找最小元，并将其位置赋给MinPosition */
		Swap(A[i], A[MinPosition]);
		/* 将未排序部分的最小元换到有序部分的最后位置 */
	}
}
```

无论如何： T = O(N^2)

如何快速找到最小元？用最小堆

堆排序（选择排序的改进算法）



https://blog.csdn.net/u013384984/article/details/79496052

总结算法思路：（建堆（调整）->交换->调整）

1. 初始化完全二叉树，存入一个数组
2. 建最大堆，调整思路：从最后一个非叶子节点开始 从下向上从右到左的调整
3. 排序，思路：从上到下从左到右，每次选择根来和最后一个元素交换，交换之后进行除了排好序的元素以外的堆调整（交换后调整堆）

堆排序不稳定

调整都是比较父节点，左子节点，右子节点里找最大值替换父节点

代码：

```
#include<iostream>
using namespace std;
```

```
void swap(int* a, int x, int y)
{
	int temp = a[y];
	a[y] = a[x];
	a[x] = temp;
}
void adjustHeap(int* a, int i, int length)
{
	//把当前元素（根）取出,因为当前元素可能一直移动
	int temp = a[i];
	//k的值是从上到下的
	for (int k = 2 * i + 1; k < length; k = 2 * k + 1)
	{
		//检测k的左右子节点,让k指向子节点中的最大
		//(k左子节点，k+1右子节点,两节点作比较)k+1<length判断是否有右节点
		if (k + 1 < length && a[k] < a[k + 1])
		{
			k++;
		}
		//如果子节点更大，就交换
		if (a[k] > temp)
		{
			swap(a, i, k);
			// 如果子节点更换了，那么，以子节点为根的子树会不会受到影响呢?
			// 循环对子节点所在的树继续进行判断
			i = k;
		}
		//不用交换，则已经是最大堆，不用循环了
		else break;
	}

}

void sort(int* a, int length)
{
	//因为堆是完全二叉树，最后一个非叶子节点就是length/2，但是因为树是数组存储的，下标从0开始，所以是length/2-1
	//是从最下面开始调整成最大堆，所以在数组中是从右到左的
	//建堆，从下到上从右到左调整
	for (int i = length / 2 - 1; i >= 0; i--)
	{
		adjustHeap(a, i, length);
	}
	//开始排序，排序是从上到下从左到右的调整
	for (int j = length - 1; j > 0; j--)
	{
		//把最大堆的根节点放到最后，就是把最大值放到最后面
		swap(a, 0, j);
		//交换后，可能堆的顺序乱了，要调整成最大堆
		//最后一个去掉的元素就不用考虑排序了
		//从上到下，从左到右的调整
		adjustHeap(a, 0, j);
	}
}
```

```
int main()
{
	int length;
	cin >> length;
	int* a = new int[length];
	for (int i = 0; i < length; i++)
	{
		cin >> a[i];
	}
	sort(a, length);
	for (int i = 0; i < length; i++)
	{
		cout << a[i] << " ";
	}
	delete[] a;

}
```