实验三 算法设计与编码

1. 分治法

① 选择排序（减而治之）

1. 从一个数组当中找到最大值，和最后一个位置进行交换。
2. 变化这个寻找最大值的范围，逐步实现排序（从I到III的过程，不需代码）
3. 用循环或递归来控制这个重复的过程，实现选择排序

|  |
| --- |
| **# @Time : 2021/11/25 11:34**  **# @Author ：kklab**  **# @File : 快速排序.py**  **# @Software : PyCharm**  **def sele\_min\_Sort(arr):**  **for i in range(len(arr) - 1):**  **# 记录最小数的索引**  **minIndex = i**  **for j in range(i + 1, len(arr)):**  **if arr[j] < arr[minIndex]:**  **minIndex = j**  **# i 不是最小数时，将 i 和最小数进行交换**  **if i != minIndex:**  **arr[i], arr[minIndex] = arr[minIndex], arr[i]**  **return arr**  **def sele\_max\_Sort(arr):**  **for i in range(len(arr) - 1, 0):**  **# 记录最大数的索引**  **maxIndex = i**  **for j in range(0, len(arr)-i):**  **if arr[j] > arr[maxIndex]:**  **maxIndex = j**  **# i 不是最大数时，将 i 和最大数进行交换**  **if i != maxIndex:**  **arr[i], arr[maxIndex] = arr[maxIndex], arr[i]**  **return arr**  **def dp\_sele\_max\_Sort(arr, n):**  **if n>1:**  **for i in range(0 ,n):**  **if arr[i] > arr[i+1]:**  **arr[i], arr[i+1] = arr[i+1], arr[i]**  **return dp\_sele\_max\_Sort(arr,n-1)**  **print(arr)**  **if n==1:**  **return arr**  **arr=[1,8,9,10,2,6]**  **print("选择排序前:", arr)**  **print("----------------循环实现----------------------")**  **return\_arr = sele\_min\_Sort(arr)**  **print("选择最小排序后:", return\_arr)**  **return\_arr = sele\_max\_Sort(arr)**  **print("选择最大值交换排序后:", return\_arr)**  **print("----------------递归实现----------------------")**  **return\_arr = dp\_sele\_max\_Sort(arr, len(arr)-1)**  **print("递归选择最大值交换排序后:", return\_arr)**  **print("---------------------------------------------")** |

② 冒泡排序（减而治之）

1. 从一个数组第一个位置到最后一个进行冒泡
2. 变化冒泡的范围，逐步实现排序 （从I到III的过程，不需代码）
3. 用循环或递归来控制这个重复的过程，实现冒泡排序。

|  |
| --- |
| **# @Time : 2021/11/25 11:34**  **# @Author ：kklab**  **# @File : 冒泡排序.py**  **# @Software : PyCharm**  **def bubbleSort(arr):**  **for i in range(1, len(arr)):**  **for j in range(0, len(arr)-i):**  **if arr[j] > arr[j+1]:**  **arr[j], arr[j + 1] = arr[j + 1], arr[j]**  **return arr**  **def dp\_bubbleSort(arr, n):**  **if n>1:**  **for j in range(0, n):**  **if arr[j] > arr[j + 1]:**  **arr[j], arr[j + 1] = arr[j + 1], arr[j]**  **return dp\_bubbleSort(arr , n-1)**  **if n==1:**  **return arr**  **arr=[1, 8, 9, 10, 2, 6]**  **print("冒泡排序前:", arr)**  **print("----------------循环实现----------------------")**  **return\_arr = bubbleSort(arr)**  **print("冒泡交换排序后:", return\_arr)**  **print("----------------递归实现----------------------")**  **# return\_arr = dp\_bubbleSort(arr, len(arr)-1)**  **print("递归冒泡交换排序后:", return\_arr)**  **print("---------------------------------------------")** |

③ 数组求和（减而治之与二分分治）

|  |
| --- |
| **# @Time : 2021/11/28 15:47**  **# @Author ：kklab**  **# @File : 数组求和.py**  **# @Software : PyCharm**  **from math import ceil**  **def arr\_sum(arr):**  **result = 0**  **for i in arr:**  **result += i**  **return result**  **def arr\_sum\_jian(arr, n):**  **if n>=0:**  **global result**  **result = arr[n]**  **return result+arr\_sum\_jian(arr, n-1)**  **if n <0:**  **return 0**  **def arr\_sum\_fen(arr, low, high):**  **if high==low:**  **return arr[low]**  **mid = (low+high)//2**  **print(f"low:{low} ,mid:{mid},high:{high}")**  **return arr\_sum\_fen(arr, low, mid) + arr\_sum\_fen(arr, mid+1, high)**  **result = 0**  **arr=[1,8,9,10,2,6]**  **print("数组:", arr)**  **return\_result = arr\_sum(arr)**  **print("普通求法结果:", return\_result)**  **print("---------------------------------------------")**  **return\_result = arr\_sum\_jian(arr, len(arr)-1)**  **print("减而治之结果:", return\_result)**  **print("---------------------------------------------")**  **return\_result = arr\_sum\_fen(arr, 0, len(arr)-1)**  **print("二分分治结果:", return\_result)**  **print("---------------------------------------------")** |

③ 归并排序（二分分治）

|  |
| --- |
| **# @Time : 2021/11/28 15:43**  **# @Author ：kklab**  **# @File : 归并排序.py**  **# @Software : PyCharm**  **def mergeSort(arr):**  **import math**  **if(len(arr)<2):**  **return arr**  **middle = math.floor(len(arr)/2)**  **left, right = arr[0:middle], arr[middle:]**  **return merge(mergeSort(left), mergeSort(right))**  **def merge(left,right):**  **result = []**  **while left and right:**  **if left[0] <= right[0]:**  **result.append(left.pop(0))**  **else:**  **result.append(right.pop(0));**  **while left:**  **result.append(left.pop(0))**  **while right:**  **result.append(right.pop(0));**  **return result**  **arr=[1,8,9,10,2,6]**  **print("排序前:", arr)**  **return\_arr = mergeSort(arr)**  **print("排序后:", return\_arr)**  **print("---------------------------------------------")** |

1. 递归与备忘录

递归解法

1. 用递归计算斐波那契数列的值
2. 用备忘录改进运算速度（思路，在递归的过程当中，如果已经计算了（措施是看值是不是-1），直接返回该位置的值，如果没有计算，则递归计算并记录到备忘录里）。

|  |
| --- |
| **# @Time : 2021/11/28 16:25**  **# @Author ：kklab**  **# @File : 斐波那契数列.py**  **# @Software : PyCharm**  **def re\_fibo(n):**  **global time**  **time += 1**  **if n <= 1:**  **return n**  **else:**  **return(re\_fibo(n-1) + re\_fibo(n-2))**  **def fibo(n):**  **global time**  **time += 1**  **if n <= 1:**  **res[n]=n**  **return n**  **if res[n]!= -1:**  **return res[n]**  **res[n] = fibo(n-1) + fibo(n-2)**  **return res[n]**  **def fx(n):**  **for i in range(0, n+1):**  **res.append(-1)**  **return fibo(n);**  **res = []**  **time = 0**  **sum = re\_fibo(10)**  **print(f"正常计算{sum},时间：{time}")**  **time = 0**  **sum = fx(10)**  **print(f"用备忘录改进运算速度计算{sum},时间：{time}")** |