实习报告

**题目**：编制一个多关键字排序程序，用三种不同算法进行排序

**组员**：吕星宇、胡康、赵昱杰 **成稿日期**：2022.06.25

一、需求分析

1. 问题描述

在进行高考分数处理时，除了需对总分进行排序外，不同的专业对单科分数的要求不同，因此尚需在总分不同的情况下，按用户提出的单科分数的次序要求排出考生录取的次序。编制一个程序，按照多个关键字的优先级，给出含多关键字的记录的排序结果。

2. 功能描述

用户首先选择关键字优先级，接着选择排序方式（LSD稳定的内部排序，LSD“收集”和“分配策略”和MSD三者之一），系统输出原始的数据顺序和排序后的数据顺序。

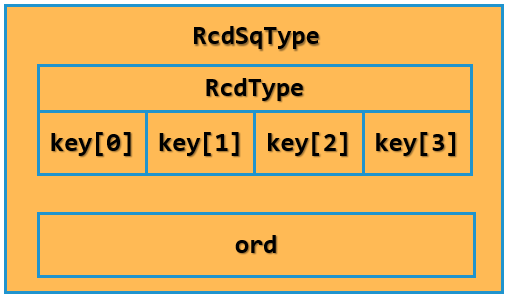
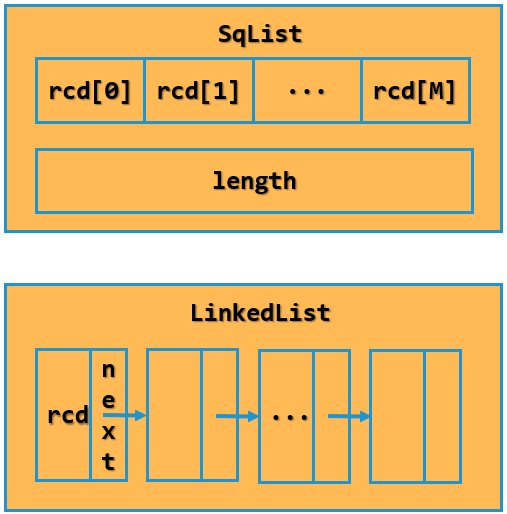
3. 测试数据

由随机数生成函数产生。

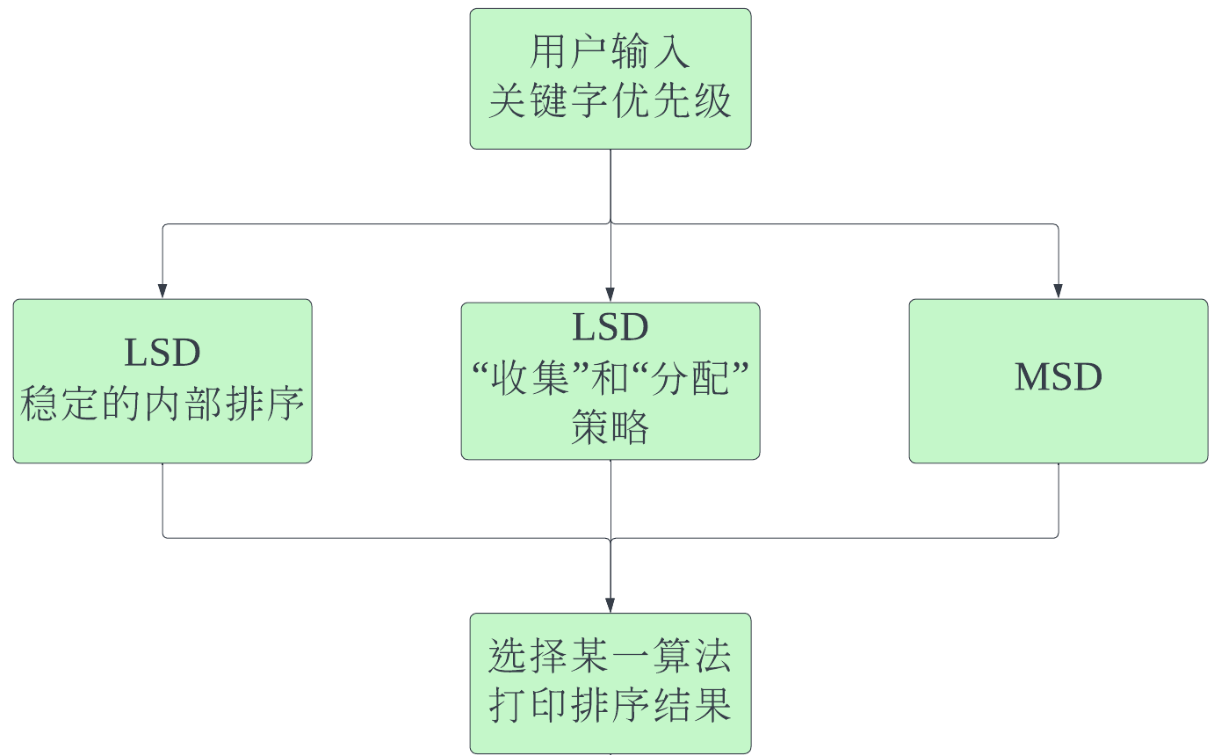
二、设计框架

1. 数据结构

使用顺序表和链表。其中为了达到“稳定的”快速排序，顺序表记录类型比链表记录类型多一个ord域，用于保存记录的位置。示意图如下图所示，其中左侧为记录类型，右侧为链表和顺序表：

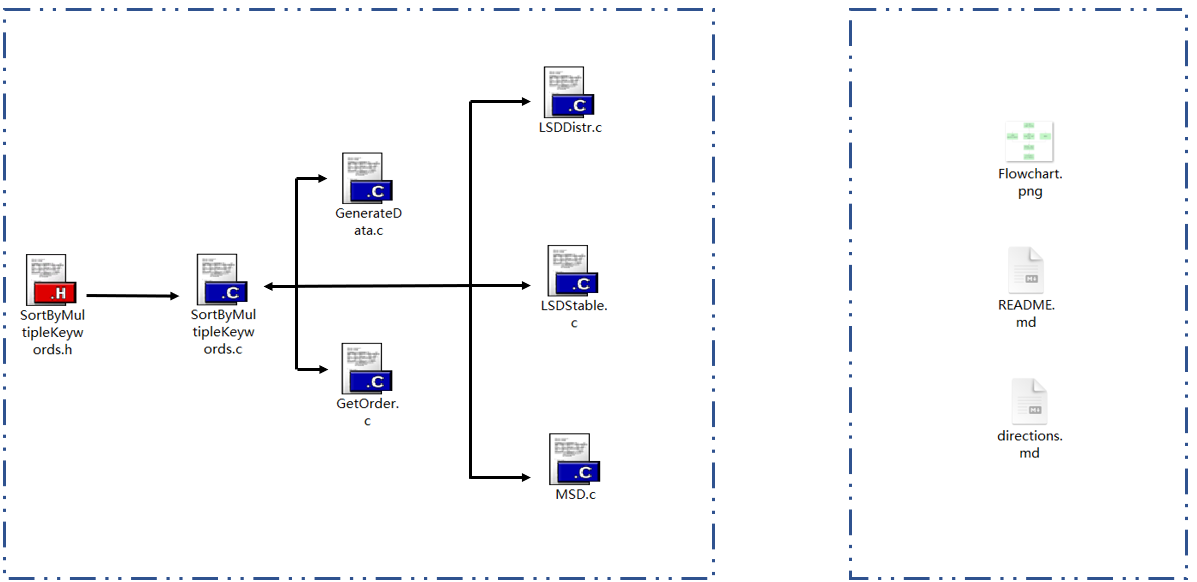
 

2. 执行流程



3. 相关文件

文件结构如下图所示：

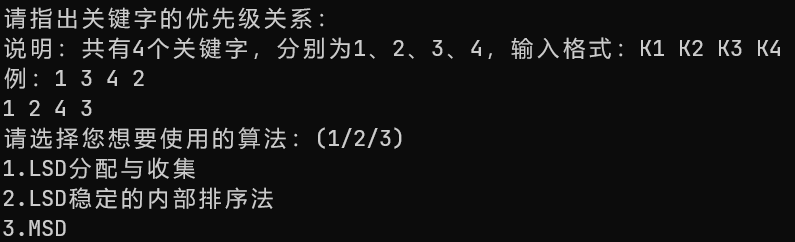


其中，头文件SortByMultipleKeywords.h保存了数据结构和主要函数的定义，可以查看该文件大致了解本程序的执行流程。SortByMultipleKeywords.h中含有主函数，根据用户输入，选择需要调用的函数。GenerateData.c将随机数产生的数字组织成顺序表和链表，GetOrder.c从用户处得到关键字的优先级，随后用户选择某一算法，对应地进入到LSDDistr.c，LSDStable.c或MSD.c中计算出排序结果，并打印。

其它文件包括：Flowchart.png，展示程序的执行流程；README.md，对程序整体架构和代码规范做说明；directions.md，用户手册。

三、测试运行结果

用户选择界面如下图所示：



这里可以选择三种算法之一，以算法3：MSD方法为例，程序通过随机数函数生成4\*100个测试数据，范围为0~100，运行结果为：

原始数据：

92 17 54 74

5 63 63 44

25 72 10 22

80 53 3 60

66 81 62 77

10 66 98 82

44 100 0 7

65 56 47 16

70 68 99 45

50 72 100 3

61 82 28 18

29 22 73 56

62 93 56 82

67 6 68 80

63 75 93 58

34 9 89 64

57 38 51 86

86 52 30 33

91 94 39 38

39 79 86 17

24 29 56 10

74 32 39 90

94 89 48 9

64 11 45 69

78 87 77 16

62 82 71 95

1 98 77 68

52 18 69 19

96 83 99 97

65 55 96 64

19 87 41 61

19 23 75 97

82 9 97 5

40 85 77 43

43 8 56 23

99 94 85 26

100 9 80 87

87 68 93 93

30 52 93 5

78 35 2 93

8 97 69 11

98 0 57 81

19 63 48 28

62 3 20 43

95 17 86 65

89 10 67 11

82 46 26 83

76 37 65 62

92 3 65 100

68 82 32 0

51 4 27 83

42 76 47 97

87 24 62 4

89 38 90 95

77 11 69 23

75 66 52 88

11 36 59 66

10 71 58 99

38 32 30 61

95 25 45 17

63 13 13 86

33 94 4 83

24 85 7 74

96 73 63 70

16 25 18 57

87 83 18 52

71 48 24 0

47 83 32 71

57 60 23 32

11 4 19 88

45 68 3 92

72 82 79 30

59 23 10 21

30 89 62 7

23 26 94 15

52 95 10 95

84 64 44 62

41 32 84 80

16 14 94 64

50 48 61 5

39 90 76 1

40 19 22 23

13 28 12 39

52 6 47 24

61 62 35 83

53 20 68 20

0 99 17 97

92 53 86 60

69 95 19 71

22 96 11 99

97 0 27 23

71 71 35 94

31 85 86 50

34 2 14 75

46 22 2 1

17 27 64 80

76 2 88 23

48 20 67 76

22 10 62 83

92 5 5 45

排序结果3：

100 9 80 87

99 94 85 26

98 0 57 81

97 0 27 23

96 83 99 97

96 73 63 70

95 25 45 17

95 17 86 65

94 89 48 9

92 53 86 60

92 17 54 74

92 5 5 45

92 3 65 100

91 94 39 38

89 38 90 95

89 10 67 11

87 83 18 52

87 68 93 93

87 24 62 4

86 52 30 33

84 64 44 62

82 46 26 83

82 9 97 5

80 53 3 60

78 87 77 16

78 35 2 93

77 11 69 23

76 37 65 62

76 2 88 23

75 66 52 88

74 32 39 90

72 82 79 30

71 71 35 94

71 48 24 0

70 68 99 45

69 95 19 71

68 82 32 0

67 6 68 80

66 81 62 77

65 56 47 16

65 55 96 64

64 11 45 69

63 75 93 58

63 13 13 86

62 93 56 82

62 82 71 95

62 3 20 43

61 82 28 18

61 62 35 83

59 23 10 21

57 60 23 32

57 38 51 86

53 20 68 20

52 95 10 95

52 18 69 19

52 6 47 24

51 4 27 83

50 72 100 3

50 48 61 5

48 20 67 76

47 83 32 71

46 22 2 1

45 68 3 92

44 100 0 7

43 8 56 23

42 76 47 97

41 32 84 80

40 85 77 43

40 19 22 23

39 90 76 1

39 79 86 17

38 32 30 61

34 9 89 64

34 2 14 75

33 94 4 83

31 85 86 50

30 89 62 7

30 52 93 5

29 22 73 56

25 72 10 22

24 85 7 74

24 29 56 10

23 26 94 15

22 96 11 99

22 10 62 83

19 87 41 61

19 63 48 28

19 23 75 97

17 27 64 80

16 25 18 57

16 14 94 64

13 28 12 39

11 36 59 66

11 4 19 88

10 71 58 99

10 66 98 82

8 97 69 11

5 63 63 44

1 98 77 68

0 99 17 97

四、调试分析

1. 问题与解决方式

**兼顾速度与稳定**

多关键字排序在使用内部排序时必须要求排序算法稳定。快速排序算法在随机数分布均匀时，时间复杂度为，但是不稳定。为了兼顾速度与稳定性，增加了ord域（相当于增加了一个关键字）虽然增加了存储空间，但是将快速排序算法转化为一个稳定的排序方式。

2. 算法的性能分析

① **LSD基于“分配”与“收集”方法策略**

共需执行次“分配”与“收集”，其中是关键字数目。每次“分配”与“收集”时，需要对个记录进行“分配”，对个链表进行“收集”，因此时间复杂度为；由于另设subLL存储分配后的新链表，空间复杂度为。

② **MSD**和**LSD基于稳定的内部排序方法策略**

时间开销为，空间开销为。

五、用户使用说明

本程序实现运用LSD分配与收集法、LSD稳定的内部排序法、MSD法三种方法进行多关键词排序，可用于高考分数排序等多个场景。

本程序设置了100个待排序记录，每个记录有4个关键字。可通过微调来改变待排记录数和关键字数。

程序开始运行时，用户需要对4个关键字的优先级进行排序，例如输入1 3 4 2代表。然后用户需要选择三种排序方法中的一种：

1. LSD分配与收集法

2. LSD稳定的内部排序法

3. MSD法

之后程序会输出100个原始记录与排序后的记录。其中原始记录由随机数产生，排序后的记录按照从高到低的顺序输出。