

算法设计与分析_第2次作业

各种排序算法的实现与分析

请用任意语言（C/C++/Java/Python等），实现选择排序、插入排序、快速排序、归并排序。

选择排序：

```
1  #include "stdio.h"
2  #include "time.h"
3
4  void sort (int nums[], int n){
5      if(nums == NULL || n <= 0){
6          return ;
7      }
8      int i, j, t;
9      clock_t st = clock();
10     for(i = 0; i < n; i++){
11         int min = nums[i];
12         for(j = i; j < n; j++){
13             if(min > nums[j]){
14                 t = nums[j];
15                 nums[j] = min;
16                 min = t;
17             }
18         }
19         nums[i] = min;
20     }
21     clock_t ft = clock();
22     double dt = (ft - st) * 1.0 / CLOCKS_PER_SEC;
23
24     for(i = 0; i < n ; i++){
25         printf("%4d ", nums[i]);
26     }
27
28     printf("\n duration: %f s\n", dt);
29 }
30
31 int main(void) {
32     int n, nums[1000005];
33     scanf("%d", &n);
34     int i;
35     for(i = 0; i < n; i++){
36         scanf("%d", &nums[i]);
37     }
38     sort(nums, n);
39
40
41     return 0;
42 }
```

插入排序：

```

1  #include "stdio.h"
2  #include <time.h>
3
4  void sort(int nums[], int n){
5      int i, j, cur;
6      clock_t st = clock();
7      for(i = 1; i < n; i++){
8          cur = nums[i];
9          for(j = i - 1; j >= 0 && nums[j] > cur; j--){
10             nums[j + 1] = nums[j];
11         }
12         nums[j + 1] = cur;
13     }
14     clock_t ft = clock();
15     double dt = (ft - st) * 1.0 / CLOCKS_PER_SEC;
16     for(i = 0; i < n; i++){
17         printf("%4d ", nums[i]);
18     }
19     printf("\n duration: %f s\n", dt);
20 }
21
22 int main(void){
23     int n, nums[1000005];
24     scanf("%d", &n);
25     int i;
26     for(i = 0; i < n; i++){
27         scanf("%d", &nums[i]);
28     }
29     sort(nums, n);
30     return 0;
31 }

```

快速排序:

```

1  #include "stdio.h"
2  #include "time.h"
3
4  int n, nums[1000005];
5
6  int keySort(int nums[], int L, int H){
7      int key, count = 1, temp;
8      int M = L + (H - L) / 2;
9      if (nums[L] > nums[H]){
10         temp = nums[L];
11         nums[L] = nums[H];
12         nums[H] = temp;
13     }
14     if (nums[M] > nums[H]){
15         temp = nums[H];
16         nums[H] = nums[M];
17         nums[M] = temp;
18     }
19     if (nums[L] < nums[M]){
20         temp = nums[L];
21         nums[L] = nums[M];
22         nums[M] = temp;
23     }

```

```

24     nums[0] = nums[L];
25     key = nums[L];
26     while (L < H){
27         while (L < H && nums[H] >= key)
28             --H;
29         nums[L] = nums[H];
30         while (L < H && nums[L] <= key)
31             ++L;
32         nums[H] = nums[L];
33         count++;
34     }
35     nums[L] = nums[0];
36     return L;
37 }
38
39 void Sort(int nums[], int L, int H){
40     int k;
41     if (L < H){
42         k = keySort(nums, L, H);
43         Sort(nums, L, k - 1);
44         Sort(nums, k + 1, H);
45     }
46 }
47
48 int main(void){
49     int i;
50     scanf("%d", &n);
51     for (i = 1; i <= n; i++){
52         scanf("%d", &nums[i]);
53     }
54     clock_t st = clock();
55     Sort(nums, 1, n);
56     clock_t ft = clock();
57     double dt = (ft - st) * 1.0 / CLOCKS_PER_SEC;
58     printf("\n duration: %f s\n", dt);
59     return 0;
60 }

```

归并排序:

```

1  #include "stdio.h"
2  #include "time.h"
3
4  int res[1000005], nums[1000005];
5
6  void merge(int L, int M, int H){
7      int i = L, j = M + 1, k = L, n;
8      for(n = L; n <= H; n++){
9          res[n] = nums[n];
10     }
11     while(i <= M && j <= H){
12         if(res[i] > res[j]){
13             nums[k] = res[j];
14             j++;
15         }else{
16             nums[k] = res[i];
17             i++;

```

```

18     }
19     k++;
20 }
21 if(i <= M){
22     for(n = i; n <= M; n++){
23         nums[k++] = res[n];
24     }
25 }
26 }
27
28 void mergeSort(int L, int H){
29     if(L + 1 >= H){
30         if(nums[L] > nums[H]){
31             int t = nums[L];
32             nums[L] = nums[H];
33             nums[H] = t;
34         }
35     }else{
36         int M = (L + H) / 2;
37         mergeSort(L, M);
38         mergeSort(M + 1, H);
39         merge(L, M, H);
40     }
41 }
42
43 int main(void){
44     int n;
45     scanf("%d", &n);
46     int i;
47     for(i = 0; i < n; i++){
48         scanf("%d", &nums[i]);
49     }
50     clock_t st = clock();
51     mergeSort(0, n - 1);
52     clock_t ft = clock();
53     double dt = (ft - st) * 1.0 / CLOCKS_PER_SEC;
54     for(i = 0; i < n; i++){
55         printf("%4d ", nums[i]);
56     }
57     printf("\n duration: %f s\n", dt);
58
59     return 0;
60 }

```

问题：

1. 生成不同规模、不同分布的测试数据，对比测试上述4种代码的运行表现。

(要求数据规模至少包括1万、10万、100万，分布要求必须包含一定范围内的随机数据。下表仅写排序运行时间，不要包括读入数据的时间。)

答：(至少包含以下内容，推荐对是否有重复元素附加测试)

	选择排序	插入排序	快速排序	归并排序
1万正序	0.151000 s	0.000000 s	0.000000 s	0.001000 s
1万逆序	0.195000 s	0.163000 s	0.001000 s	0.001000 s
1万随机	0.342000 s	0.081000 s	0.002000 s	0.002000 s
10万正序	14.904000 s	0.001000 s	0.005000 s	0.006000 s
10万逆序	20.961000 s	19.095000 s	0.007000 s	0.008000 s
10万随机	28.099000 s	9.745000 s	0.015000 s	0.019000 s
100万正序		0.000000 s	0.056000 s	0.068000 s
100万逆序	2920.444000 s	1204.200000 s	0.076000 s	0.091000 s
100万随机	2677.662000 s	634.461000 s	0.187000 s	0.172000 s

附：各规模随机数的说明

1万随机：（应至少包括范围，是否有重复元素，分布：均匀分布即可）

10万随机：（应至少包括范围，是否有重复元素，分布：均匀分布即可）

100万随机：（应至少包括范围，是否有重复元素，分布：均匀分布即可）

生成随机数据的方法：（以10万规模随机数为例）

```

1  import random
2
3  n = int(input())
4  print(n)
5  LR = min(10000, n)
6  for _ in range(n):
7      print(random.randint(-LR, LR), end=" ")

```

2.实际的排序问题输入，元素可能有重复，如果输入中有较多重复元素，如何改进快速排序，使划分效率更高？请写出改进后的核心代码或算法

答：优化找数，优化重复

```

1  #include "stdio.h"
2  #include "time.h"
3
4  int n, nums[1000005];
5
6  int keySort(int nums[], int L, int H){
7      int key, count = 1, temp;
8      int M = L + (H - L) / 2;
9      if (nums[L] > nums[H]){
10         temp = nums[L];
11         nums[L] = nums[H];
12         nums[H] = temp;
13     }
14     if (nums[M] > nums[H]){
15         temp = nums[H];
16         nums[H] = nums[M];

```

```

17     nums[M] = temp;
18 }
19 if (nums[L] < nums[M]){
20     temp = nums[L];
21     nums[L] = nums[M];
22     nums[M] = temp;
23 }
24 nums[0] = nums[L];
25 key = nums[L];
26 while (L < H){
27     while (L < H && nums[H] >= key)
28         --H;
29     nums[L] = nums[H];
30     while (L < H && nums[L] <= key)
31         ++L;
32     nums[H] = nums[L];
33     count++;
34 }
35 nums[L] = nums[0];
36 return L;
37 }
38
39 void Sort(int nums[], int L, int H){
40     int k;
41     if (L < H){
42         k = keySort(nums, L, H);
43         Sort(nums, L, k - 1);
44         Sort(nums, k + 1, H);
45     }
46 }
47
48 int main(void){
49     int i;
50     scanf("%d", &n);
51     for (i = 1; i <= n; i++){
52         scanf("%d", &nums[i]);
53     }
54     clock_t st = clock();
55     Sort(nums, 1, n);
56     clock_t ft = clock();
57     double dt = (ft - st) * 1.0 / CLOCKS_PER_SEC;
58     printf("\n duration: %f s\n", dt);
59     return 0;
60 }

```