作业 1：算法分析与复杂度

宋雨桐

2024 年 3 月 24 日

1 排序算法

1.1 循环

1.1.1 冒泡排序

时间复杂度

- $\mathcal{O}(n^2)$
- $\Omega(n)$

空间复杂度

- $\Theta(1)$

```c
void bubble_loop(int *arr, int len)
{
	for (int i = 0; i < len; i++)
	{
		for (int j = 0; j < len - i - 1; j++)
		{
			if (arr[j] > arr[j + 1])
			{
				arr[j] = arr[j] ^ arr[j + 1];
				arr[j + 1] = arr[j] ^ arr[j + 1];
				arr[j] = arr[j] ^ arr[j + 1];
			}
		}
	}

	return;
}
```

输出结果：

```
Before bubble_loop:
41 59 51 4 45 68 87 35 67 10 78 22 29 44 2 27 67 20 8 46 5 62 99 28 60 73 8
45 79 78 10 72 89 13 76 86 81 15 22 49 77 0 71 7 97 25 86 16 46 95
After bubble_loop:
0 2 4 5 7 8 8 10 10 13 15 16 20 22 22 25 27 28 29 35 41 44 45 45 46 46 49 51
59 60 62 67 67 68 71 72 73 76 77 78 78 79 81 86 86 87 89 95 97 99
```

1.1.2 选择排序

时间复杂度

- $\Theta(n^2)$

空间复杂度

- $\Theta(1)$

```c
void selection_loop(int *arr, int len)
{
        for (int i = 0; i < len; i++)
        {
                int min = arr[i];
                int index = i;

                for (int j = i; j < len; j++)
                {
                        if (arr[j] < min)
                        {
                                min = arr[j];
                                index = j;
                        }
                }

                if (index != i)
                {
                        arr[i] = arr[i] ^ arr[index];
                        arr[index] = arr[i] ^ arr[index];
                        arr[i] = arr[i] ^ arr[index];
                }
        }

        return;
}
```

输出结果：

```
Before selection_loop:
82 87 76 26 34 8 79 34 83 79 83 57 83 81 60 19 60 61 97 27 61 53 0 89 26 26
33 48 72 87 54 7 74 83 85 60 91 64 46 26 43 29 84 79 62 44 98 23 57 96
After selection_loop:
0 7 8 19 23 26 26 26 26 27 29 33 34 34 43 44 46 48 53 54 57 57 60 60 60 61
61 62 64 72 74 76 79 79 79 81 82 83 83 83 83 84 85 87 87 89 91 96 97 98
```

1.1.3 插入排序

时间复杂度

- $\mathcal{O}(n^2)$
- $\Omega(n)$

空间复杂度

- $\Theta(1)$

```
void insertion_loop(int *arr, int len)
{
        for (int i = 1; i < len; i++)
        {
                for (int j = i - 1, k = i; j >= 0; j--, k--)
                {
                        if (arr[k] < arr[j])
                        {
                                arr[k] = arr[k] ^ arr[j];
                                arr[j] = arr[k] ^ arr[j];
                                arr[k] = arr[k] ^ arr[j];
                                continue;
                        }
                        break;

                }
        }
        return;
}
```

输出结果：

```
Before insertion_loop:
58 21 3 38 22 84 72 33 73 56 89 21 32 50 5 43 23 74 74 29 0 15 86 84 10 41
82 62 94 67 96 4 88 99 42 62 83 66 96 8 22 37 30 6 88 35 1 63 62 75
After insertion_loop:
0 1 3 4 5 6 8 10 15 21 21 22 22 23 29 30 32 33 35 37 38 41 42 43 50 56 58 62
62 62 63 66 67 72 73 74 74 75 82 83 84 84 86 88 88 89 94 96 96 99
```

1.2 递归

1.2.1 冒泡排序

时间复杂度

- $\mathcal{O}(n^2)$
- $\Omega(n)$

空间复杂度

- $\Theta(1)$ 由于使用了尾递归，所以空间复杂度不随 n 的大小改变。

```c
void bubble_recursion(int *arr, int len)
{
        if (len == 1)
                return;

        for (int i = 0; i < len - 1; i++)
        {
                if(arr[i] > arr[i + 1])
                {
                        arr[i] = arr[i] ^ arr[i + 1];
                        arr[i + 1] = arr[i] ^ arr[i + 1];
                        arr[i] = arr[i] ^ arr[i + 1];
                }
        }

        return bubble_recursion(arr, len - 1);
}
```

输出结果：

```
Before bubble_recursion:
42 45 54 1 96 27 86 43 35 46 41 21 91 50 98 40 17 76 6 69 43 66 73 96 61 53
```

```
33 31 82 85 74 77 82 81 78 78 8 64 73 43 62 67 65 6 17 15 98 34 91 57
After bubble_recursion:
1 6 6 8 15 17 17 21 27 31 33 34 35 40 41 42 43 43 43 45 46 50 53 54 57 61 62
64 65 66 67 69 73 73 74 76 77 78 78 81 82 82 85 86 91 91 96 96 98 98
```

1.2.2 选择排序

时间复杂度

- $\Theta(n^2)$

空间复杂度

- $\Theta(1)$

```c
void selection_recursion(int *arr, int len)
{
        if (len == 1)
        {
                return;
        }

        int index = len - 1;
        int max = arr[len - 1];

        for (int j = 0; j < len; j++)
        {
                if (arr[j] > max)
                {
                        max = arr[j];
                        index = j;
                }
        }

        if (index != len - 1)
        {
                arr[index] = arr[index] ^ arr[len - 1];
                arr[len - 1] = arr[index] ^ arr[len - 1];
                arr[index] = arr[index] ^ arr[len - 1];
        }

        return selection_recursion(arr, len - 1);
}
```

输出结果：

```
Before selection_recursion:
1 24 15 74 12 77 28 53 19 28 12 75 42 6 49 61 51 38 59 71 92 38 50 5 45 53 5
73 60 0 9 61 76 24 35 89 53 16 94 25 96 6 0 38 64 1 51 15 39 11
After selection_recursion:
0 0 1 1 5 5 6 6 9 11 12 12 15 15 16 19 24 24 25 28 28 35 38 38 38 39 42 45
49 50 51 51 53 53 53 59 60 61 61 64 71 73 74 75 76 77 89 92 94 96
```

1.2.3 插入排序

时间复杂度

- $\mathcal{O}(n^2)$
- $\Omega(n)$

空间复杂度

- $\Theta(1)$

```c
void insertion_recursion(int *arr, int len)
{
        if (len == 1)
        {
                return;
        }

        int index = len - 1;
        int count = len - 2;

        while(arr[count] > arr[index] && index < SIZE)
        {
                arr[count] = arr[count] ^ arr[index];
                arr[index] = arr[count] ^ arr[index];
                arr[count] = arr[count] ^ arr[index];
                index++;
                count++;
        }

        return insertion_recursion(arr, len - 1);
}
```

输出结果：

```
Before insertion_recursion:
55 79 48 10 41 65 82 15 85 74 23 80 42 84 61 78 21 71 87 69 44 58 10 7 62 89
30 8 60 43 14 15 22 14 77 15 79 11 30 16 86 54 96 28 90 10 58 12 81 97
After insertion_recursion:
7 8 10 10 10 11 12 14 14 15 15 15 16 21 22 23 28 30 30 41 42 43 44 48 54 55
58 58 60 61 62 65 69 71 74 77 78 79 79 80 81 82 84 85 86 87 89 90 96 97
```

2 汉诺塔问题

2.1 循环

时间复杂度

- $\Theta(n)$

空间复杂度

- $\Theta(1)$

```c
#include <gmp.h>
#include <stdio.h>
#include <stdlib.h>
#define SIZE 64

int main(void)
{
        mpz_t result;
        mpz_init(result);
        mpz_set_ui(result, 7);

        mpz_t tmp;
        mpz_init(tmp);

        for (int i = 4; i <= SIZE; i++)
        {
                mpz_mul_ui(tmp, result, 2);
                mpz_set(result, tmp);
                mpz_add_ui(tmp, result, 1);
                mpz_set(result, tmp);
        }
        gmp_printf("hanoi(%d) = %Zu\n",SIZE, result);

        mpz_clear(result);
        mpz_clear(tmp);

        return EXIT_SUCCESS;
}
```

输出结果：

```
hanoi(64) = 18446744073709551615
```

2.2 递归

时间复杂度

- $\Theta(n)$

空间复杂度

- $\Theta(n)$

```
#include <inttypes.h>
#include <stdio.h>
#include <stdlib.h>
#define SIZE 64

u_int64_t hanoi(int size);

int main(void)
{
        printf("moves = %" PRIu64 "\n", hanoi(SIZE));
        return EXIT_SUCCESS;
}

u_int64_t hanoi(int size)
{
        if (size == 3)
        {
                return 7;
        }
        return (u_int64_t) 2 * hanoi(size - 1) + (u_int64_t) 1;
}
```

输出结果：

```
hanoi(64) = 18446744073709551615
```

3 角谷猜想

```c
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
        int max_count = 0;
        int index = 1;
        for (int i = 1; i <= 100; i++)
        {
                int result = i;
                int count = 0;

                while(result != 1)
                {
                        if (result % 2 == 0)
                                result /= 2;
                        else
                                result = 3 * result + 1;
                        count++;
                }

                if (max_count < count)
                {
                        max_count = count;
                        index = i;
                }
        }

        printf(”All␣numbers␣are␣satisfied␣with␣collatz␣conjecture!\n”);

        printf(”%d␣has␣the␣longest␣sequence!\n”, index);
        printf(”The␣sequence␣is:␣\n”);
        printf(”%d␣”, index);
        for (int i = 0; i < max_count; i++)
        {
                if (index % 2 == 0)
                        index = index / 2;
                else
                        index = index * 3 + 1;
                printf("%d␣", index);
        }
        printf("\n");
        return EXIT_SUCCESS;
}
```

输出结果：

```
All numbers are satisfied with collatz conjecture!
97 has the longest sequence!
The sequence is:
97 292 146 73 220 110 55 166 83 250 125 376 188 94 47 142 71 214 107 322 161
484 242 121 364 182 91 274 137 412 206 103 310 155 466 233 700 350 175 526
263 790 395 1186 593 1780 890 445 1336 668 334 167 502 251 754 377 1132 566
283 850 425 1276 638 319 958 479 1438 719 2158 1079 3238 1619 4858 2429
7288 3644 1822 911 2734 1367 4102 2051 6154 3077 9232 4616 2308 1154 577
1732 866 433 1300 650 325 976 488 244 122 61 184 92 46 23 70 35 106 53 160
80 40 20 10 5 16 8 4 2 1
```