上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

VE 281 Homework 1 Gengchen Yang 518370910088

Exercise 1.

Since we have $\log n < \sqrt{n} < n < n\log n < n^2 < 2^n < n! < 2^{n^2}$

we directly obtain: $(\log n)! < (n+1)!$;

$n^3 < ne^n$; $ne^n < (n+1)!$;

Then, setting $n = 10^a$, we obtain the following functions:

$\sqrt{2}^a$, $(10^a+1)!$, $10^a \cdot e^{10^a}$, $a!$, 10^{3a}, 10.

We immediately find 10 is the smallest, and:

$10 < \sqrt{2}^a < 10^{3a} < a!$;

Then combining the two, we have

$$\boxed{n^{\frac{1}{\log n}} < (\sqrt{2})^{\log n} < n^3 < (\log n)! < ne^n < (n+1)!}$$

Exercise 2.

$$\log(n!) = \log 1 + \log 2 + \cdots + \log n$$

$$< \underbrace{\log \tfrac{n}{2} + \log\left(\tfrac{n}{2}+1\right) + \cdots + \log n}_{\frac{n}{2}\text{ terms}}$$

(Suppose n is even, if not, simply take $\lfloor \frac{n}{2} \rfloor$)

$$< \underbrace{\log \tfrac{n}{2} + \log \tfrac{n}{2} + \cdots + \log \tfrac{n}{2}}_{\frac{n}{2}\text{ terms}}$$

$$= \frac{n}{2}\log\frac{n}{2} = O(n\log n)$$

For the proof of $\frac{n}{2}\log\frac{n}{2} = \Omega(n\log n)$,

We find that $\frac{n}{2}\log_{10}\frac{n}{2} \geq \frac{1}{4} n\log n = (\frac{1}{4}n)\log\frac{n}{2} + \frac{1}{4}n\cdot\log 2$

When n is large enough, there obviously exist n_0 such that

for all $n > n_0$, $\frac{1}{2}n\log\frac{n}{2} - \frac{1}{4}n\log\frac{n}{2} = \frac{1}{4}n\log\frac{n}{2} > \frac{1}{4}n\log 2$.

Exercise 3.

We only need to use Dselect to choose pivot.

Instead of pivot = rand(); or pivot = first, we use:

```
pivot = Dselect(A, n, i);
Dselect (int A[], int n, int i){
    if (n==1) return A[1];
    C = n/5 medians;
    p = Dselect (C, n/5, n/10);
    if (j==i) return p;
    if (j>i) return Dselect (1st part of A, j-1, i);
    else return Dselect (2nd part of A, n-j, i-j); }
```

Exercise 4.

```
bool Func(int A[], int n) {
    int value = 0;
    int count = 0;    // suppose 0 is not in A, otherwise change
                      the initialization.

    for (int i=0; i<n; i++) {
        if (A[i] == value) count ++;
        else {
            if (count > 1) count --;
            else {
                value = A[i];
                count = 1;  }}}
    count = 0;
    for (int i=0; i<n; i++) {
        if (A[i] == value) count++; }
    if (count > n/2) return 1;
    else return 0;
```

Exercise 6

1. Use two loops:

```
for (int i=0; i<n; i++) {
    for (int j=0; j<n; j++) {
        if (A[i] + A[j] == S)  return 1; }}
return 0;
```

2. Use Binary Search inside the first loop

```
for (int i=0; i<n; i++) {
    valueNeed = S - A[i];
    BinarySearch(valueNeed);  // if found return 1,
    ... }                        if not found, do nothing
return 0;
```

Exercise 7.

1. Insertion Sort. Because the size is small, n^2 is not large compared to $kn\log n + mn + 0$, n^2 will be quite small.
2. Insertion Sort. Because best case is $O(n)$ and happens when the array is already sorted.

3. Counting Sort. Because we know the range is limited, and counting sort is wrote to be good in this case, also, we need stability.

4. Merge Sort. Because it's the fastest ~~s~~ stable ~~s~~ sorting when n is large. Complexity $O(n\log n)$.

Exercise 9.

1. Wrong. When $n \to \infty$, we can find no constant c that $cn \geq n\log n$ since $\log n \to \infty$.

2. ~~Wrong~~ Right $\log_2 2^n = n$, $\log_2 n! = O(n\log_2 n)$ since $n = O(n\log n)$ we have $2^n = O(n!)$

3. Right.

We have $f_1(n) \geq c_1 g_1(n)$, for $n > n_1$; $f_2(n) \geq c_2 g_2(n)$ for $n > n_2$, then $f_1(n) \cdot f_2(n) \geq c_1 c_2 g_1(n) \cdot g_2(n)$ for $n > \max\{n_1, n_2\}$

4. Right.

We have $f_1(n) \geq c_1 g_1(n)$, $f_1(n) \leq d_1 g_1(n)$

$f_2(n) \geq c_2 g_2(n)$, $f_2(n) \leq d_2 g_2(n)$

Then $f_1(n) + f_2(n) \geq c_1 g_1(n) + c_2 g_2(n) > 1 \cdot \max\{c_1 g_1(n), c_2 g_2(n)\}$

~~And, similar for $\leq$, Here proof complete.~~

and $f_1 + f_2 < 2 \cdot \max\{c_1 g_1(n), c_2 g_2(n)\}$.

~~then~~

Hence $f_1(n)+f_2(n)=\Omega(\max\{g_1(n), g_2(n)\})$

Similar for $O(\max\{g_1(n), g_2(n)\})$

~~the~~ proof complete.

Quiz problem 1.

~~2nd call: quickSort({3,2,1,5,4,6,7,9,10,8},0,5)~~

~~3rd call: quickSort({2,1,3,5,4,6,7,9,10,8},~~

2nd call: quickSort({2,5,6,4,1,3,7,9,10,8},0,5) 左

3rd call: quickSort({1,2,3,4,5,6,7,9,10,8},0,0) 左

4th call: quickSort({1,2,3,4,5,6,7,9,10,8},2,5) 左

{1,2,3,6,5,4,7,9,10,8}, 2,1) 左

same , 3,5) 左

{1,2,3,5,4,6,7,9,10,8},3,4) 左

{1,2,3,4,5,6,7,9,10,8},3,3) 左

same ,4,4) 左

same ,6,5) 左

same ,7,9) 左

{1,2,3,4,5,6,7,8,9,10},7,7)

same 9,9)

Quiz Problem 2.

1. swap to beginning of array.
2. start counters i=1 and j=N-1.
3. increment i until we find A[i] >= pivot,
4. decrement j until we find A[j] <= pivot
5. if i<j, swap A[i] with A[j]. go to step 3.
6. else, swap pivot with A[j].

Quiz problem 3.

1. No it's not
2. (1,a), (2,a), (2,b), (3,a)
3. (2,a), (2,b), (3,a), (1,a)
4. Yes, it is. Though they both swap and are $O(n^2)$, ~~Quiz Problem 4.~~ bubble sort always swap in one direction, hence the situation of swapping the ~~smal~~ second smallest element to the back and then swap it back is avoided.

Quiz Problem 4.

	Worst	Average	In-place	Stable
Insertion	$O(N^2)$	$O(N^2)$	Yes	Yes
Selection	$O(N^2)$	$O(N^2)$	Yes	No
Bubble	$O(N^2)$	$O(N^2)$	Yes	Yes
Merge	$O(N\log N)$	$O(N\log N)$	No	Yes
Quick	$O(N^2)$	$O(N\log N)$	Weakly	No

Quiz Problem 5.

Change step 3 into: $D[0]=0$ for $i=1$ to k: $D[i]=D[i-1]+C[i-1]$,

Change the decrease in step four into increase.

Finished