Data Structure & Algorithms

Sunbeam Infotech

Nilesh Ghule

1 2 3 4 5 6 7 8 9 10

```
void print(int start, int end) {
    printf(start);
    if(start == end)
        return;
    print(start+1, end);
}
```

9

get first (last) bit

9 % 2 = 1

9 / 2 = 4 ← apply same for n/2

base cond → n == 0

```
void dec2bin(int n) {
    if(n==0)
        return 0;
    print(n%2);
    dec2bin(n/2);
}
```

24 → 2 * 12 ← apply same logic for n/prime

get first prime factor

2 * 6

2 * 3

3 * 1

base cond

```
void prime_fact(int n, int f) {
    if(n==1)
        return;
    if(n%f == 0) {
        print(f);
        prime_fact(n/f, f);
    }
    else
        prime_fact(n, f+1);
}
```

```
int fact(int n) {            // 5
    if (n == 0)
        return 1;
    r = n * fact(n-1);       // 5 * 24
    return r;                // 120
}
```

main

```
int fact(int n) {            // 4
    if (n == 0)
        return 1;
    r = n * fact(n-1);       // 4 * 6
    return r;                // 24
}
```

```
int fact(int n) {            // 3
    if (n == 0)
        return 1;
    r = n * fact(n-1);       // 3 * 2
    return r;                // 6
}
```

```
int fact(int n) {            // 2
    if (n == 0)
        return 1;
    r = n * fact(n-1);       // 2
    return r;                // 2
}
```

```
int fact(int n) {            // 1
    if (n == 0)
        return 1;
    r = n * fact(n-1);       // 1
    return r;                // 1
}
```

```
int fact(int n) {            // 0
    if (n == 0)
        return 1;
    r = n * fact(n-1);
    return r;
}
```

120 = 5! = 5 * 4! (24)

24 = 4! = 4 * 3! (6)

6 = 3! = 3 * 2! (2)

2 = 2! = 2 * 1! (1)

1 = 1! = 1 * 0! (1)

0! = 1

stack

f(0)
f(1)
f(2)
f(3)
f(4) n=4 r=
f(5) n=5 r=
main()

space ↑ ✗

time ↑ ✗

readable ↑ ✓

function activation record / stack frame

↓

Created on stack for each fn call & destroyed when fn returns.

↓

1. local vars
2. arguments
3. return address

Binary Search

0	1	2	3	4	5	6	7	8
11	22	33	44	55	66	77	88	99
L				m				R

```
search (L, R, key) {
    if invalid part (L > R), ele not found (-1);

    find mid of partition (L to R)
    if mid ele is same as key, return mid;
    if key < mid ele, search in left part (L, m-1)
    else, search in right part (m+1, R)

}
```

Selection Sort

→ unstable sort

4 2 6 3 5 1

$$T \propto \frac{n(n-1)}{2}$$

$$T \propto n^2 - n$$

× $n >> 1$

$n^2 >>>>> n$

↓

can ignore all lower order terms

$$T \propto n^2$$

$$\boxed{O(n^2)}$$

		cmps
i=0 pass 1	1	→ 5
i=1	2	→ 4
i=2	3	→ 3
i=3	4	→ 2
i=4	5	→ 1
		15

n elements

↓

itrs ↓

n−1
+ n−2
+ n−3
+ ...
+ 1

$$\frac{n(n-1)}{2}$$

i=0	i=1	i=2	i=3	i=4	
0	1	2	3	4	5
1	2	3	4	5	6

```
for ( i=0 ; i< n-1 ; i++) {
    for(j= i+1 ; j< n ; j++) {
        if (a[i]  > a[j])
            swap(a[i], a[j]);
    }
}
```

Bubble Sort (basic)

4 2 6 3 5 1

each pass iters = $n-1$

num of passes = $n-1$

total iterations = $(n-1)^2$

$$T \propto (n-1)^2$$

$$T \propto n^2 \underbrace{-2n + 1}_{\times}$$

$$T \propto n^2$$

$$\boxed{O(n^2)}$$

0	1	2	3	4	5
1	2	3	4	5	6

j j+1

n-1 passes →

```
for(i=1; i<n; i++) {
    for(j=0; j<n-1; j++) {
        if(a[j] > a[j+1])
            swap(a[j], a[j+1]);
```

3

3

Improved Bubble Sort

4 2 6 3 5 1

0	1	2	3	4	5
2	1	3	4	5	6

Pass 1 = 5
2 = 4
3 = 3
4 = 2
5 = 1

15

$\boxed{O(n^2)}$

like selection sort calculation

```
for(i=1; i<n; i++) {
    for(j=0; j<n-i; j++) {
        if(a[j] > a[j+1])
            swap(a[j], a[j+1]);
```

3

3

Improved Bubble Sort

4 2 6 3 5 1

pass 1 = 5
2 = 4
3 = 3
4 = 2
5 = 1
_____
15

$\boxed{O(n^2)}$

like selection sort calculation

0	1	2	3	4	5
1	2	3	4	5	6

```
for(i=1; i<n; i++) {
    flag = false;
    for(j=0; j<n-i; j++) {
        if(a[j] > a[j+1]) {
            swap(a[j], a[j+1]);
            flag=true;
        }
    }
    if (flag==false)
        break;
}
```

Best case: (already sorted)
→ $O(n)$

avg/worst case: (random)
→ $O(n^2)$

Insertion Sort

4 2 6 3 5 1

1 2 3 4 5 6

$\boxed{temp < a[j]}$

0	1	2	3	4	5
1	2	3	4	5	6

$\boxed{j >= 0}$

j -1 0 1 2 3 4 5

× 1 2 3 4 5 6 1

```
temp = a[i];
for( j=i-1; j >= 0 && temp < a[j]; j-- )
    a[j+1] = a[j];
a[j+1] = temp;
```

pass 1 = 1
2 = 2
3 = 3
4 = 4
5 = 5

15

$1 + 2 + \dots + (n-1)$

$T \propto \frac{n(n-1)}{2}$

$O(n^2)$

	i=1	i=2	i=3	i=4	i=5
0	1	2	3	4	5
1	2	3	4	5	6

1 2 3 4 5 6

Best case (already sorted)
$O(n)$ → only one itr of inner loop for each pass.

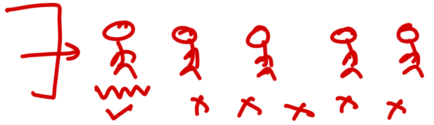
worst case (reverse)
$O(n^2)$

Stack and Queue

- Stack & Queue are utility data structures.
- Can be implemented using array or linked lists.
- Usually time complexity of stack & queue operations is O(1).
- Stack is Last-In-First-Out structure.
- Stack operations
  - push()
  - pop()
  - peek()
  - isEmpty()
  - isFull()*
- Simple queue is First-In-First-Out structure.
- Queue operations
  - push() enque()
  - pop() deque()
  - peek()
  - isEmpty()
  - isFull()*
- Queue types
  - Linear queue
  - Circular queue
  - Deque
  - Priority queue

ADT

1. Array
   - get ele at pos
   - set ele at pos
   - sort, search
   - slice, ...
2. Queue



Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>