Theoretical Analysis

Lose information
- Consider the worse case only
- Keep the largest term only

e.g.
- Selection Sort
- Wild-Guess Sort → All $O(n^2)$!

Pseudocode → Programming Syntax / Natural Language

Standard Keyword

- branch: if / then / else
- loop: while, for repeat / until
- value: return

Notation:

- variable ← value
- Array[index]
- function(arguments)

Indentation ☆ (Also can use { } for clarity)

Remarks:

①

i++	i = i + 1
x*y, x%y	x·y, x mod y
sqrt(x), power(a,b)	$\sqrt{x}$, a^b

② Define Data Structure first. (if new)

③ Use Standard / learned Algorithm as black boxes.

④ functions → decompose complex algorithm

⑤ Use plain natural language

Prove: $\log(n!) = \Theta(n\log n)$ Stirling's Formula

first. prove $\log(n!) = O(n\log n)$

$\log(n!) = \log(n) + \log(n-1) \cdots + \log(1) \le n\log n$

$= O(n\log n)$

Then, prove $\log(n!) = \Omega(n\log n)$

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

$\ge \frac{n}{2} \cdot \log(\frac{n}{2}) = \frac{n}{2}(\log n - \log 2)$

$= \Omega(n\log n)$

Hence $\log(n!) = \Theta(n\log n)$

Divide and Conquer

Binary Search

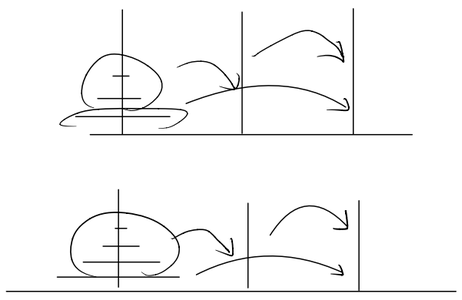
```
BinarySearch(A, p, r, x):
if p > r then return nil
q ← ⌊(p+r)/2⌋
if x = A[q] return q
if x < A[q] then BinarySearch(A, p, q-1, x)
else BinarySearch(A, q+1, r, x)
```

Analysis

$T(n)$ → number of comparison.

$T(n) = T(n/2) + 2, \quad T(1) = 1$

Tower of Hanoi



```
MoveTower(n, peg1, peg2, peg3)
if n = 1 then
    move the only disc from peg1 to peg3.
else
    MoveTower(n-1, peg1, peg3, peg2)
    move the only disc from peg1 to peg3.
    MoveTower(n-1, peg2, peg1, peg3)
```

$$\begin{cases} T(n) = 2T(n-1) + 1 \\ T(1) = 1 \end{cases}$$

$\Rightarrow T(n) = 2T(n-1) + 1$

$= 2(2T(n-2) + 1) + 1$

$= 2^{n-1}T(1) + 1 + 2 + 4 + \cdots 2^{n-1}$

$= 2^{n-1} + \frac{2^{n-1}-1}{2-1} = \underline{2^n - 1} = \Theta(2^n)$

Merge Sort

```
MergeSort(A, p, r):
if p = r then return
q ← ⌊(p+r)/2⌋
MergeSort(A, p, q)
MergeSort(A, q+1, r)
Merge(A, p, q, r)
```

```
Merge(A, p, q, r):
create two array L, R.
L ← A[p, q], R ← A[q+1, r]
append ∞ at the end of L and R.
i ← 1, j ← 1
for k ← p to r
    if L[i] ≤ R[j] then
        A[k] ← L[i]
        i = i + 1
    else
        A[k] ← R[j]
        j = j + 1
```

Analysis

$T(n) \le T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + O(n)$

$T(1) = O(1)$

☆ Simplification: We replace "≤" with "="

$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + O(n)$

☆ Simplification: replace $O(n)$ with n, $O(1)$ with 1.

Also. We assume n is power of 2 → ignore $\lceil\ \rceil$ $\lfloor\ \rfloor$

assume $n = 2^k$

$T(n) = 2T(\frac{n}{2}) + n$

$= 2(2T(\frac{n}{4}) + \frac{n}{2}) + n$

$= 2^i T(\frac{n}{2^i}) + in$

$= 2^{\log_2 n} T(1) + n\log_2 n$

$= n\log_2 n + n = \Theta(n\log n)$