lecture 5 – September 14

EXAMPLE 7. $f(N) = N\log(N)$, $g(N) = N^{1.5} = N \cdot N^{0.5}$

Which function grows more quickly? $f(N)$ or $g(N)$?

$N\log(N)$ or $N\ N^{0.5}$? $\Big) :N$

$\log(N)$ or $N^{0.5}$? $\Big)$ square

$\log^2(N)$ or $\underline{\underline{N}}$?

$\rightsquigarrow$ $g(N)$ grows more quickly than $f(N)$.

1.3 RUNNING TIME CALCULATIONS

We will focus on Big-O analysis on running time.

$\longrightarrow$ important :
- do not underestimate running time
- ignore constant factors, low-order terms, ...

EXAMPLE 8.

```
int mySum (int n)
{
    int partialSum;            ignore / constant
    partialSum = 0;            O(1)   (1 unit)
                  1 unit     1 unit   1 unit
    for (int i=1; i<=n; i++)          O(1) }
        partialSum += i*i*i;  4 units O(1) }  O(n)   enter loop n times
    return partialSum;         ignore / constant
}
```

$\rightsquigarrow$ running time is $O(n)$

General rules

RULE 1 Consecutive statements / fragments

- add the running times of the statements / fragments
- Big-O : Big-O of the max of the two (recall Theorem 2)

$$\left.\begin{array}{ll} a += 3; & O(1) \\ b = a; & O(1) \end{array}\right\} O(1)$$

$$\{$$
$$\left.\begin{array}{ll} \text{fragment 1} & O(N) \\ \text{fragment 2} & O(\log(N)) \end{array}\right\} O(N)$$
$$\}$$

RULE 2 If / Else

```
if (condition)
    S1
else
    S2
```

- add the running time of testing "condition" to the larger of the running times of S1 and S2.

$$\left.\begin{array}{ll} \text{if } (a[i] < a[j]) & \longrightarrow O(1) \\ \quad S1 & \longrightarrow O(N) \\ \text{else} & \\ \quad S2 & \longrightarrow O(N^2) \end{array}\right| \; O(\cancel{1}+N^2) = O(N^2)$$

RULE 3. Loops

- multiply the running time of the fragment inside the loop, including tests, by the number of iterations

→ see Example 8

RULE 4. Nested loops

- analyze inside-out, repeatedly applying rule 3.

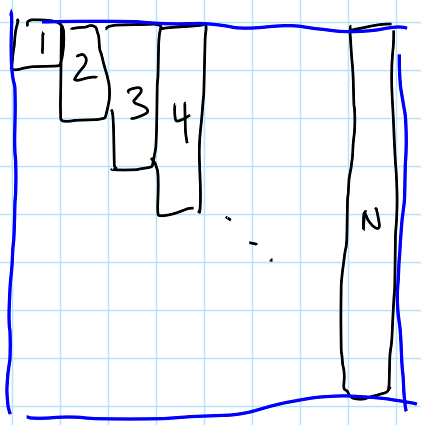
— — $O(1)$ —

— — —

`a[i] += a[j] + i+j;` $\rightarrow O(1)$ } $O(N)$ } $O(N^2)$

```
for (i=0; i<N; i++)
    for (j=0; j<=i, j++)
        S1
```
($\rightarrow O(1)$) } $O(i)$??

$$T(N) = 1 + 2 + 3 + 4 + \ldots + N = \sum_{k=1}^{N} k = \frac{N(N+1)}{2} = O(N^2)$$



blue area : N^2

black strips:

$\sum_{k=1}^{N} k$ $\qquad O(N^2)$

Rule 5. Recursion

- determine an appropriate recurrence relation and solve it

```
void someRecFct (int list[], int left, int right)
{
    if (left == right)            //base case
        fragment 1                                  //e.g. O(1)
    else
    {  int center = (left+right)/2;                 // O(1)
       someRecFct (list, left, center);             // O(T(N/2))
       someRecFct (list, center+1, right);          // O(T(N/2))
       fragment2                                    // e.g. O(N)
}
```

q. Mergesort

$$T(1) = 1$$

$$T(N) = 2 \cdot T\left(\frac{N}{2}\right) + N \qquad \text{if } N = 2^k \text{ for some } k \in \mathbb{N}$$

fragment 2

$T(1) = 1$

$T(2) = T(2^1) = 2 \cdot T(1) + 2 = 2 \cdot 1 + 2$

$T(4) = T(2^2) = 2 \cdot T(2) + 4 = 2 \cdot 2 \cdot 1 + 2 \cdot 2 + 4$

$T(8) = T(2^3) = 2 \cdot T(4) + 8 = 2 \cdot 2 \cdot 2 \cdot 1 + 2 \cdot 2 \cdot 2 + 2 \cdot 4 + 8$

$$T(2^k) = \underline{2^k \cdot 1} + \underline{2^{k-1} \cdot 2} + \ldots + \underline{2 \cdot 2^{k-1}} + \underline{2^k}$$

$$= (k+1) \cdot 2^k = 2^k + k \cdot 2^k$$

$N = 2^k$ $\qquad T(N) = N + \log(N) \cdot N = O(N \log(N))$

$k = \log(N)$