Asymptotic Complexity

CS10001: Programming & Data Structures

Pallab Dasgupta
Professor, Dept. of Computer Sc. & Engg.,
Indian Institute of Technology Kharagpur

Transitive Closure

```
    Transclosure ( int adjmat[ ][max], int path[ ][max] )
{
    for (i = 0; i < max; i++)
        for (j = 0; j < max; j++)
            path[i][j] = adjmat[i][j];

    for (k = 0; k < max; k++)
      for (i = 0; i < max; i++)
         for (j = 0; j < max; j++)
        if ((path[i][k] == 1)&&(path[k][j] == 1)) path[i][j] = 1;
    }
```

How many operations are performed?

Merge-Sort

```
void mergesort ( int a[ ], int lo, int hi )          → T(n)
{
    int m;
    if (lo<hi) {
        m=(lo+hi)/2;
        mergesort(a, lo, m);                         → T(n/2)
        mergesort(a, m+1, hi);                       → ?
        merge(a, lo, m, hi);                         → T(n/2)
    }
}
```

Function Merge

```
void merge ( int a[ ], int lo, int m, int hi )
{
int i, j, k, b[MAX];

// copy both halves to auxiliary array b
for (i=lo; i<=hi; i++) b[i]=a[i];

i=lo; j=m+1; k=lo;
// copy back next-greatest element at each time
while (i<=m && j<=hi)
if (b[i]<=b[j]) a[k++]=b[i++];
else a[k++]=b[j++];

// copy back remaining elements of first half (if any)
while (i<=m) a[k++]=b[i++];
}
```

Complexity of mergesort

T(0) = 1

T(n) = T(n/2) + n + T(n/2)
= 2T(n/2) + n

Rewrite n as 2x:

T(2x) = 2T(2x-1) + 2x
= 2T(2T(2x-2) + 2x-1) + 2x
= 22T(2x-2) + 2x + 2x
= x2x

Therefore: T(n) n log2 n

O-notation

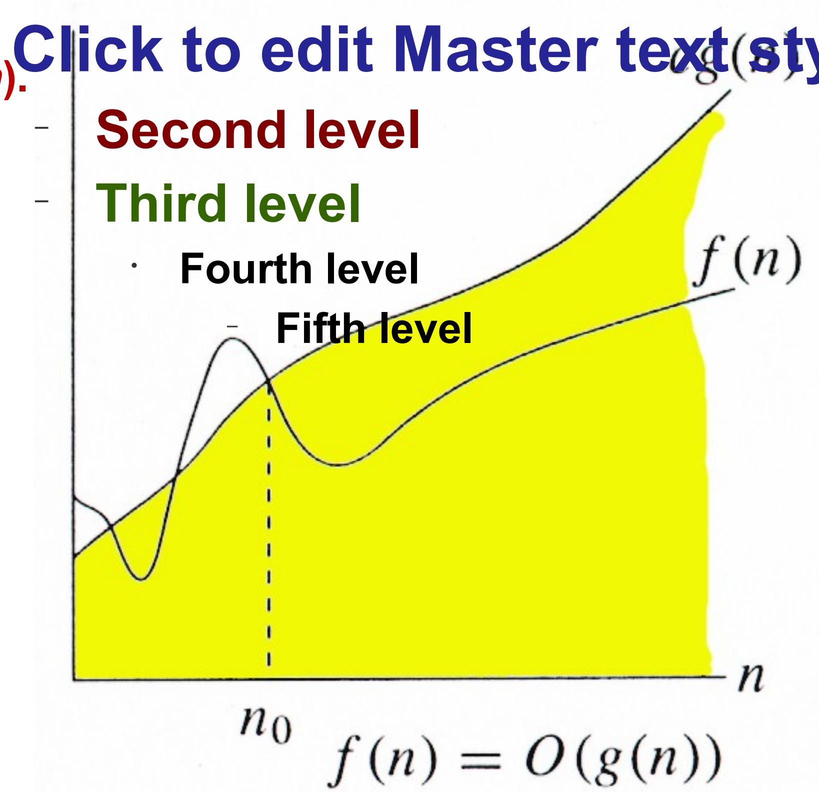
For function $g(n)$, we define $O(g(n))$, big-O of n, as the set:

$O(g(n)) = \{ f(n) : \exists$ positive constants c and n_0, such that $\forall n \geq n_0$, we have $0 \leq f(n) \leq cg(n) \}$

Intuition: Set of all functions whose *rate of growth* is the same as or lower than that of $g(n)$.

$g(n)$ is an *asymptotic upper bound* for $f(n)$.

Click to edit Master text styles
- Second level
- Third level
  - Fourth level
    - Fifth level



Examples

$O(g(n)) = \{ f(n) :$ ∃ positive constants c and n_0, such that ∀ $n \geq n_0$, we have $0 \leq f(n) \leq cg(n) \}$

- Any linear *function* $an + b$ is in $O(n^2)$. **How?**
- Show that $3n^3 = O(n^4)$ for appropriate c and n_0.

Some common notations

[redacted]

O(1)	Constant
O(log n)	Logarithmic
O(n log n) = O(log n!)	Loglinear
O(n2)	Quadratic
O(nc)	Polynomial
O(cn)	Exponential
O(n!)	Factorial

Recursive Permutation Generator

```
void perm (char list[ ], int i, int n)
{
int j, tmp;
if (i == n) {
for (j=0; j<=n; j++) printf("%c", list[ j ]);
printf("\n");
}
else {
for (j=i; j <= n; j++) {
SWAP(list[ i ], list[ j ], tmp);
perm(list, i+1, n);
SWAP(list[ i ], list[ j ], tmp);
}
}
}
```

```
#define SWAP(x, y, t) ((t) = (x), (x) = (y), (y) = (t))
```