"Runtime" Comparisons

Which of these grows the fastest

$\log_2(N^5)$ $\quad N^1 \quad$ $\log(N^N)$

log trick: $5\log_2(N)$ slowest growth $\quad 1 \cdot N \quad N\log(N)$ fastest growth

Note: $\log(N)$ is faster growing than 1

Exponential vs Polynomial

Exponential $\quad$ Polynomial

C^N, $C > 1$ $\quad$ N^k, $k > 0$

Exponential always grows faster than polynomial

- Derivative of C^N is $a C^N$ for some positive a. We don't care about constant factors and Derivative of N^k is $b N^{k-1}$ for integer k we will eventually reach N^0

Calculus Def. of $O(f(n))$

$f(n) = O(g(n))$ = for some $c \in \mathbb{R}^+$

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} \leq c$$

Common Error Best Case

Don't say

"If we let n be 1, then the best case is $O(1)$." ← Bad

def. say n tends towards ∞

n should be large!

```
int func7(int a[], int sizea, int b[], int sizeb) {
   int i, j;
   for (i = 0; i < sizea; i++)
      for (j = 0; j < sizeb; j++)
         if (a[i] == b[j])
            return 1;
   return 0;
}
```

Best Worst
$O(1)$ $O(|A| \cdot |B|)$
$O(sizea \cdot sizeb)$

```
int func8(int a[], int sizea, int b[], int sizeb) {
   int i, j;
   for (i = 0; i < sizea; i++)
      if (binSearch(b, sizeb, a[i])
         return 1;
   return 0;
}
```

Best Worst
$O(1)$ $O(|A| \log(|B|))$

$4n^3 + 4n^2$
$n^3 + n^2$

```
int func9(int n) {
   int x = 0, i, j;
   for (i = 1; i <= n*(8*n+8); i++) {
      for (j = n / 2; j <= n; j++) {
         x = x + (n - j);
      }
   }
   return x;
}
```

$(8n^2 + 8n)\,n/2$

$O(n^3)$

$i = 1, 2, \ldots, (8n^2+8n)\,n/2$ ($(8n^2+8n)\,n/2$ total)

$j = n/2, n/2+1, n/2+2, \ldots, n$

$n - n/2$

```
for(i=0; i<|A|; i++)
   sum++;
if (binSearch(b, sizeb, a[0])
   return sum;
return 0;
```

$O(|A| + \log(|B|))$
$\neq O(|A|)$

```
int func4(int ** array, int n) {
   int i = 0, j = 0;
   while (i < n) {
      while (j < n && array[i][j] == 1)
         j++;
      i++;
   }
   return j;
}
```

N = 30 1 seconds

N = 60 2 seconds

↳ O(N)

```
int func5(int ** array, int n) {
   int i = 0, j;
   while (i < n) {
      j = 0;
      while (j < n && array[i][j] == 1)
         j++;
      i++;
   }
   return j;
}
```

for (int i = 1; i < N; i++)

N/1 [for (int j = i; j < N; j += i)

sum += j

$N \cdot \left(\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{N}\right)$

$\log(N)$

$O(N \cdot \log(N))$

Harmonic Series

```
int func6(int array[], int n) {
   int i, j, sum = 0;
   for (i = 0; i < n; i++)
      for (j=i+1; j < n; j++)
         if (array[i] > array[j])
            sum++;
   return sum;
}
```

i=0 j = 1, 2, ..., n-1 (n-1)

i=1 j = 2, 3, ..., n-1 (n-2)

i=2 j = 3, ..., n-1 (n-3)

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

n n n

$1+2+3+\cdots+n-3+n-2+n-1$

$O(n^2)$