Tilde Approximation

EMBRY-RIDDLE Aeronautical University

Definition. We write $\sim f(N)$ to represent any function that, when divided by $f(N)$, approaches 1 as N grows, and we write $g(N) \sim f(N)$ to indicate that $g(N)/f(N)$ approaches 1 as N grows.

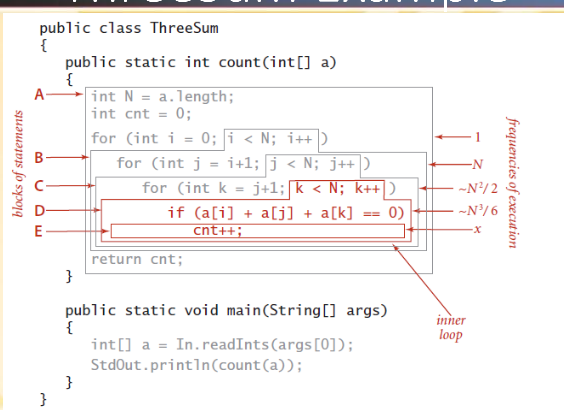
We say our algorithm's performance is approximately if $\sim f(n)$

$$\lim_{n\to\infty} \frac{g(N)}{f(N)} = 1$$

1

ThreeSum Example

EMBRY-RIDDLE Aeronautical University



Anatomy of a program's statement execution frequencies

3-sum cost model. When studying algorithms to solve the 3-sum problem, we count *array accesses* (the number of times an array entry is accessed, for read or write).

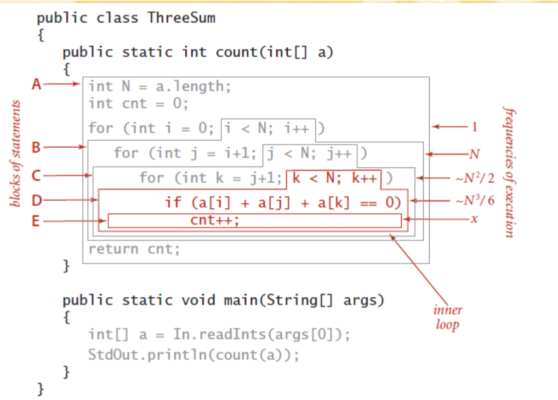
statement block	time in seconds	frequency	total time
E	t_0	x (*depends on input*)	$t_0 x$
D	t_1	$N^3/6 - N^2/2 + N/3$	$t_1 (N^3/6 - N^2/2 + N/3)$
C	t_2	$N^2/2 - N/2$	$t_2 (N^2/2 - N/2)$
B	t_3	N	$t_3 N$
A	t_4	1	t_4
		grand total	$(t_1/6) N^3 + (t_2/2 - t_1/2) N^2 + (t_1/3 - t_2/2 + t_3) N + t_4 + t_0 x$
		tilde approximation	$\sim (t_1 / 6) N^3$ (*assuming x is small*)
		order of growth	N^3

Analyzing the running time of a program (example)

How many three number combinations sum to zero from our array?

2

ThreeSum Example



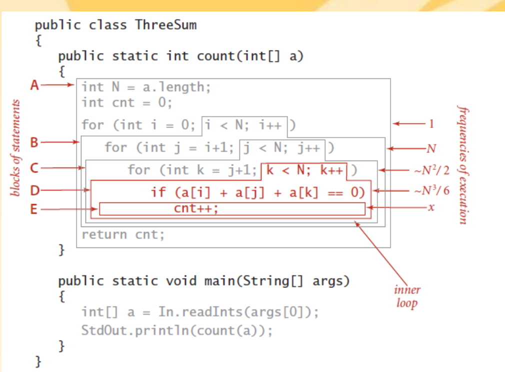
Anatomy of a program's statement execution frequencies

- How do we determine these frequencies of execution?
- One option, we can count the number of operations in each loop and how many times each loop is called
- Working from the inside to the outside
- We can sum up operations that are on the same level
- More on this approach with some examples soon.
- How else can we make these estimations?

CS 315 College of Engineering, Daytona Beach, FL 3

3

ThreeSum Example



Anatomy of a program's statement execution frequencies

statement block	time in seconds	frequency	total time
E	t_0	x (*depends on input*)	$t_0 x$
D	t_1	$N^3/6 - N^2/2 + N/3$	$t_1 (N^3/6 - N^2/2 + N/3)$
C	t_2	$N^2/2 - N/2$	$t_2 (N^2/2 - N/2)$
B	t_3	N	$t_3 N$
A	t_4	1	t_4
		grand total	$(t_1/6) N^3 + (t_2/2 - t_1/2) N^2 + (t_1/3 - t_2/2 + t_3) N + t_4 + t_0 x$
		tilde approximation	$\sim (t_1 / 6) N^3$ (*assuming x is small*)
		order of growth	N^3

Analyzing the running time of a program (example)

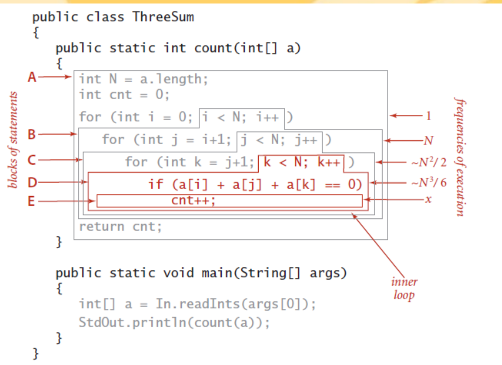
E. Is the operation of cnt++, which is listed as approximately t0 time and is performed x times. We do not know x precisely because it is determined by our input.

CS 315 College of Engineering, Daytona Beach, FL 4

4

ThreeSum Example

EMBRY-RIDDLE Aeronautical University



Anatomy of a program's statement execution frequencies

statement block	time in seconds	frequency	total time
E	t_0	x (depends on input)	$t_0 x$
D	t_1	$N^3/6 - N^2/2 + N/3$	$t_1 (N^3/6 - N^2/2 + N/3)$
C	t_2	$N^2/2 - N/2$	$t_2 (N^2/2 - N/2)$
B	t_3	N	$t_3 N$
A	t_4	1	t_4
		grand total	$(t_1/6)N^3 + (t_2/2 - t_1/2)N^2 + (t_1/3 - t_2/2 + t_3)N + t_4 + t_0 x$
		tilde approximation	$\sim (t_1/6)N^3$ (assuming x is small)
		order of growth	N^3

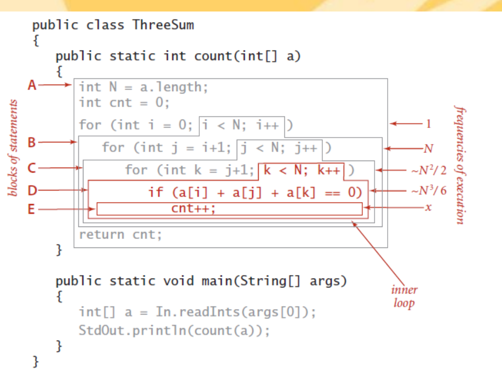
Analyzing the running time of a program (example)

The code for block D is given an approximation of ~N^3/6

D is performing E block about $\sim \frac{n^3}{6}$ times

ThreeSum Example

EMBRY-RIDDLE Aeronautical University



Anatomy of a program's statement execution frequencies

statement block	time in seconds	frequency	total time
E	t_0	x (depends on input)	$t_0 x$
D	t_1	$N^3/6 - N^2/2 + N/3$	$t_1 (N^3/6 - N^2/2 + N/3)$
C	t_2	$N^2/2 - N/2$	$t_2 (N^2/2 - N/2)$
B	t_3	N	$t_3 N$
A	t_4	1	t_4
		grand total	$(t_1/6)N^3 + (t_2/2 - t_1/2)N^2 + (t_1/3 - t_2/2 + t_3)N + t_4 + t_0 x$
		tilde approximation	$\sim (t_1/6)N^3$ (assuming x is small)
		order of growth	N^3

Analyzing the running time of a program (example)

The code for block C executes the content of block D approximately $\sim \frac{N^2}{2}$ times, which is typical of the 2nd nested loop of a multi-nested loop.

ThreeSum Example

EMBRY-RIDDLE Aeronautical University

```
public class ThreeSum
{
   public static int count(int[] a)
   {
      int N = a.length;
      int cnt = 0;
      for (int i = 0; i < N; i++)
         for (int j = i+1; j < N; j++)
            for (int k = j+1; k < N; k++)
               if (a[i] + a[j] + a[k] == 0)
                  cnt++;
      return cnt;
   }

   public static void main(String[] args)
   {
      int[] a = In.readInts(args[0]);
      StdOut.println(count(a));
   }
}
```

blocks of statements: A, B, C, D, E — frequencies of execution: 1, N, $\sim N^2/2$, $\sim N^3/6$, x — inner loop

Anatomy of a program's statement execution frequencies

statement block	time in seconds	frequency	total time
E	t_0	x (depends on input)	$t_0 x$
D	t_1	$N^3/6 - N^2/2 + N/3$	$t_1(N^3/6 - N^2/2 + N/3)$
C	t_2	$N^2/2 - N/2$	$t_2(N^2/2 - N/2)$
B	t_3	N	$t_3 N$
A	t_4	1	t_4
		grand total	$(t_1/6)N^3 + (t_2/2 - t_1/2)N^2 + (t_1/3 - t_2/2 + t_3)N + t_4 + t_0 x$
		tilde approximation	$\sim (t_1/6)N^3$ (assuming x is small)
		order of growth	N^3

Analyzing the running time of a program (example)

Block B executes N time calling C and its subblocks with each call.
Lastly, A times a fixed set of time to perform the configuration of the array once.

7

ThreeSum Example

EMBRY-RIDDLE Aeronautical University

```
public class ThreeSum
{
   public static int count(int[] a)
   {
      int N = a.length;
      int cnt = 0;
      for (int i = 0; i < N; i++)
         for (int j = i+1; j < N; j++)
            for (int k = j+1; k < N; k++)
               if (a[i] + a[j] + a[k] == 0)
                  cnt++;
      return cnt;
   }

   public static void main(String[] args)
   {
      int[] a = In.readInts(args[0]);
      StdOut.println(count(a));
   }
}
```

blocks of statements: A, B, C, D, E — frequencies of execution: 1, N, $\sim N^2/2$, $\sim N^3/6$, x — inner loop

Anatomy of a program's statement execution frequencies

statement block	time in seconds	frequency	total time
E	t_0	x (depends on input)	$t_0 x$
D	t_1	$N^3/6 - N^2/2 + N/3$	$t_1(N^3/6 - N^2/2 + N/3)$
C	t_2	$N^2/2 - N/2$	$t_2(N^2/2 - N/2)$
B	t_3	N	$t_3 N$
A	t_4	1	t_4
		grand total	$(t_1/6)N^3 + (t_2/2 - t_1/2)N^2 + (t_1/3 - t_2/2 + t_3)N + t_4 + t_0 x$
		tilde approximation	$\sim (t_1/6)N^3$ (assuming x is small)
		order of growth	N^3

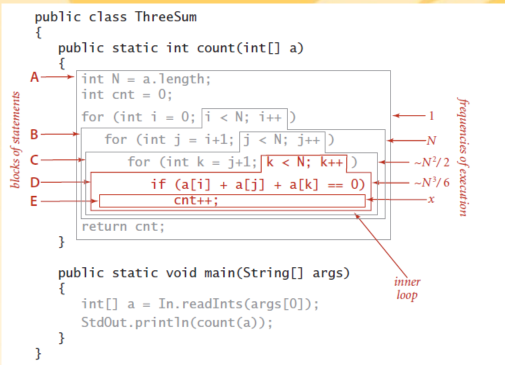
Analyzing the running time of a program (example)

In essence, we really need to know how many times the content of block D occur given the input.
We can treat its operations as close to constant time, but we need to know how often.

8

ThreeSum Example

EMBRY-RIDDLE Aeronautical University



Anatomy of a program's statement execution frequencies

ThreeSum essentially visits each combination of three elements from the array only once.

Recall, we can use the binomial coefficient as a means of identifying how many combinations given N values and choosing k of those values.

binomial coefficients $\binom{N}{k} \sim N^k/k!$ when k is a small constant

By the vary nature of the algorithm and its purpose, we can use this as the approximation for how many times we call block D, which is the number of times this block is executed.

So, we can see that this is a problem that is approximately $\sim \frac{N^3}{3!} = \sim \frac{N^3}{6}$

Helpful Tables

EMBRY-RIDDLE Aeronautical University

description	notation	definition
floor	$\lfloor x \rfloor$	largest integer not greater than x
ceiling	$\lceil x \rceil$	smallest integer not smaller than x
natural logarithm	$\ln N$	$\log_e N$ (x such that $e^x = N$)
binary logarithm	$\lg N$	$\log_2 N$ (x such that $2^x = N$)
integer binary logarithm	$\lfloor \lg N \rfloor$	largest integer not greater than $\lg N$ (# bits in binary representation of N) – 1
harmonic numbers	H_N	$1 + 1/2 + 1/3 + 1/4 + \ldots + 1/N$
factorial	$N!$	$1 \times 2 \times 3 \times 4 \times \ldots \times N$

Commonly encountered functions in the analysis of algorithms

description	approximation
harmonic sum	$H_N = 1 + 1/2 + 1/3 + 1/4 + \ldots + 1/N \sim \ln N$
triangular sum	$1 + 2 + 3 + 4 + \ldots + N \sim N^2/2$
geometric sum	$1 + 2 + 4 + 8 + \ldots + N = 2N - 1 \sim 2N$ when $N = 2^n$
Stirling's approximation	$\lg N! = \lg 1 + \lg 2 + \lg 3 + \lg 4 + \ldots + \lg N \sim N \lg N$
binomial coefficients	$\binom{N}{k} \sim N^k/k!$ when k is a small constant
exponential	$(1 - 1/x)^x \sim 1/e$

Useful approximations for the analysis of algorithms

Helpful Tables (2)

description	order of growth	typical code framework	description	example
constant	1	`a = b + c;`	*statement*	*add two numbers*
logarithmic	$\log N$	[*see page 47*]	*divide in half*	*binary search*

linear	N	`double max = a[0];`
		`for (int i = 1; i < N; i++)`
		`if (a[i] > max) max = a[i];`
			loop	*find the maximum*

linearithmic	$N \log N$	[*see* ALGORITHM 2.4]	*divide and conquer*	*mergesort*

quadratic	N^2	`for (int i = 0; i < N; i++)`
		`for (int j = i+1; j < N; j++)`
		`if (a[i] + a[j] == 0)`
		`cnt++;`
			double loop	*check all pairs*

cubic	N^3	`for (int i = 0; i < N; i++)`
		`for (int j = i+1; j < N; j++)`
		`for (int k = j+1; k < N; k++)`
		`if (a[i] + a[j] + a[k] == 0)`
		`cnt++;`
			triple loop	*check all triples*

exponential	2^N	[*see* CHAPTER 6]	*exhaustive search*	*check all subsets*

CS 315 College of Engineerin

11

Caveats

EMBRY-RIDDLE Aeronautical University

- Handling large constants in lower order terms
  - Simplify dropping lower order terms when their influence is low.
  - $2N^2 + cN$ is $\sim 2N^2$ if c is small, but ..
  - As c increases, say 10^3 or 10^6, it should not be dropped because including the term in the model better represents the problem's growth characteristics
- Nondominant inner loop
  - Cannot assume that inner loop will dominate the computation
  - Some problems include significant additional code at the same nested level as the inner loop
- Instruction time
  - Treat all instructions as approximately the same time
- System activity outside of analysis should be negligible
- Some cases will be too close to call, or not worth the effort of being more precise or knowing which of two algorithms is truly best
- Strong dependence on input
- Multiple problem parameters
  - Consider the whitelist example from Module 1. For each of the M items within the input, it takes log N runtime to find the solution using binary search

CS 315 College of Engineering, Daytona Beach, FL 12

12