Summary

• Simple runtime problems

'Counting' instructions
 Detailed

• Terminology and notation:

 $-\log_2 N = \log N$

Estimate runtime

• Problem:

The total number of instructions in a program (or a piece of code) is 10¹² and it runs on a computer that executes 10⁹ instructions per second. How long will it take to run this program? Give the answer in seconds. If it is very large, transform it in larger units (hours, days, years).

• Summary:

- Total instructions: 10¹²
- Speed: 10⁹ instructions/second
- Answer:

– Time = (total instructions)/speed =

 $(10^{12} \text{ instructions}) / (10^9 \text{ instr/sec}) = 10^3 \text{ seconds} \sim 15 \text{ minutes}$

• Note that this computation is similar to computing the time it takes to travel a certain distance (e.g. 120miles) given the speed (e.g. 60 miles/hour).

Estimate runtime

- A slightly different way to formulate the same problem:
 - total number of instructions in a program (or a piece of code) is 10¹² and
 - it runs on a computer that executes one instruction in one nanosecond (10⁻⁹ seconds)
 - How long will it take to run this program? Give the answer in seconds. If it is very large, transform it in larger units (hours, days, years)
- Summary:
 - 10¹² total instructions
 - 10⁻⁹ seconds per instruction
- Answer:

Time = (total instructions) * (seconds per instruction) =
 (10¹² instructions)* (10⁻⁹ sec/instr) = 10³ seconds ~ 15 minutes

C conventions

- Body of loops :
 - Several instructions with curly braces
 - One instruction indented (with or without curly braces)
 - No instruction
 - With semicolon, or without

Counting instructions: detailed

Worksheet

Count in detail the total number of instructions executed by each of the following pieces of code:

```
// Example A. Notice the ; at the end of the for loop.
temp = 5; x = temp * 2;
for (i = 0; i < n; i++);
// Example B (source: Dr. Bob Weems)
for (i=0; i<n; i++)
   for (t=0; t<p; t++)</pre>
    {
        c[i][t]=0;
        for (k=0; k<r; k++)
              c[i][t]+=a[i][k]*b[k][t];
    }
```

Counting instructions: detailed

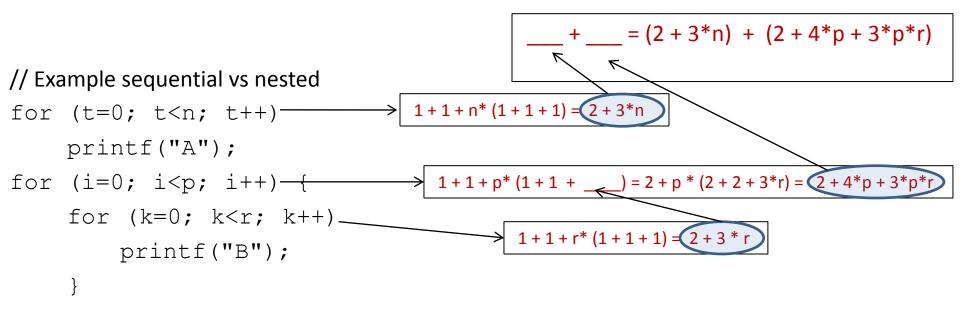
Answers

1 + 1 + n* (1 + 1 + body_instr_count) false true for (init; cond; update) // assume the condition is TRUE n times body // Example A. Notice the ; at the end of the for loop. temp = 5; x = temp * 2;for (i = 0; i<n; i++) / + 1 + n* (1 + 1 + 0) = 4 + 2n // Example B (source: Dr. Bob Weems) for (i=0; i<n; i++) → 1+1+n*(1+1+ <u>→</u>)=2+n*(2+2+5*p+3*p*r)=2+4*n+5*n*p+3*n*p*r for $(t=0; t<p; t++) \longrightarrow 1+1+p^*(1+1+1+ ___) = 2+p^*(3+2+3*r) = 2+5*p+3*p*r$ { c[i][t]=0;for $(k=0; k<r; k++) \longrightarrow | 1+1+r^*(1+1+1)=(2+3*)$ c[i][t] += a[i][k] * b[k][t];

Counting instructions: sequential vs nested loops Worksheet

```
// Example sequential vs nested
for (t=0; t<n; t++)
    printf("A");
for (i=0; i<p; i++) {
    for (k=0; k<r; k++)
        printf("B");
    }</pre>
```

Counting instructions: sequential vs nested loops Answers



CLRS

- The textbook, CLRS, provides a more detailed analysis that uses different costs (time) for instructions (based on their type).
 - Easy to adapt the above method to do that: just reapply the method, but count only specific instructions (assignments, additions, comparisons,...)
 - More details that we will want to skip over in the end (=> use Big-Oh).

Detailed instruction count Worksheet

- What does this code do?
- Give a detailed count of instructions for the case when the while loop stops because the condition (left <= right) is false.

```
/* code adapted from Sedgewick
  Assume A is sorted in increasing order and that left and
right are in range indexes for A.*/
1. int mistery(int A[], int v, int left, int
right) {
3.
     while (left <= right) {</pre>
4.
     int m = (left+right)/2;
5.
    if (v == A[m]) return m;
6. if (v < A[m])
7.
           right = m-1;
8.
      else
9.
       left = m+1;
10.
    }
11. return -1;
12. }
```