Algorithms

Part 3: Time Complexity Basics

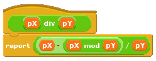
Constant, Linear, and Quadratic Time

Notes for CSC 100 - The Beauty and Joy of Computing
The University of North Carolina at Greensboro

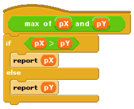
Constant time

We say a script (or part of a script or block definition) takes *constant time* if it is a constant (usually small) number of basic steps, regardless of input.

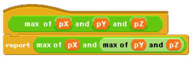
Question: Are all of these constant time?



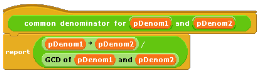
```
(pX) div (pY)
report ((pX) - ((pX) mod (pY))) / (pY)
```



```
max of (pX) and (pY)
if <(pX) > (pY)>
    report (pX)
else
    report (pY)
```

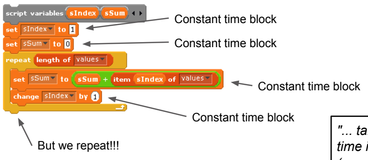


```
max of (pX) and (pY) and (pZ)
report max of (pX) and (max of (pY) and (pZ))
```



```
common denominator for (pDenom1) and (pDenom2)
report ((pDenom1) * (pDenom2)) / (GCD of (pDenom1) and (pDenom2))
```

What about loops?



```
script variables (sIndex) (sSum)
set sIndex to 1
set sSum to 0
repeat (length of values)
    set sSum to ((sSum) + (item (sIndex) of values))
    change sIndex by 1
```

Constant time block

Constant time block

Constant time block

Constant time block

But we repeat!!!

The number repetitions depends on length of "values"

- *So this is not constant time...*

"... takes constant time if it is a constant (usually small) number of basic steps, regardless of input"

Constant time operations, repeated "length of input" times is *linear time*

Mathematically: Constant time loop body is time "c"
Repeated "n" times where n is length of list

Total time is then c*n   (that's a linear function!)

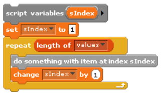
General list index iterator pattern

On previous slide:

- *Time was expressed as a function of input size*
- Could write time as T(n) = c*n

Very important "Big Idea"!!!

In general:



```
script variables sIndex
set sIndex to 1
repeat length of values
    do something with item at index sIndex
    change sIndex by 1
```

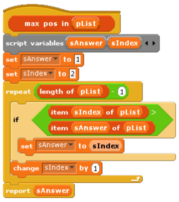
Mystery operation!!!

We know how many times it repeats, and all basic blocks are constant time except perhaps our "do something..." block

- In general, if time for "do something..." block is T(n), then time for complete script with loop is n*T(n)
- If "do something" is constant time, total time is c*n (linear)
- It "do something" is linear time, total time is c*n^2 (*quadratic*)

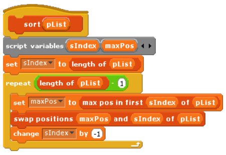
Two challenges

What's the time complexity?



```
max pos in pList
script variables sAnswer sIndex
set sAnswer to 1
set sIndex to 2
repeat (length of pList) - 1
    if (item sIndex of pList) > (item sAnswer of pList)
        set sAnswer to sIndex
    change sIndex by 1
report sAnswer
```

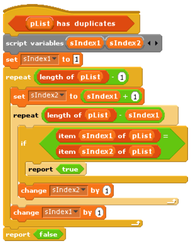
What's the time complexity?



```
sort pList
script variables sIndex maxPos
set sIndex to length of pList
repeat (length of pList) - 1
    set maxPos to max pos in first sIndex of pList
    swap positions maxPos and sIndex of pList
    change sIndex by -1
```

Another challenge

The following predicate tests whether a list has any duplicates:



```
pList has duplicates
script variables sIndex1 sIndex2
set sIndex1 to 1
repeat (length of pList) - 1
    set sIndex2 to sIndex1 + 1
    repeat (length of pList) - sIndex1
        if (item sIndex1 of pList) = (item sIndex2 of pList)
            report true
        change sIndex2 by 1
    change sIndex1 by 1
report false
```

Question: What's the time complexity?

Predicting Program Times - Linear

Basic idea: Given time complexity and sample time(s) can estimate time on larger inputs

Linear time: When input size doubles, time doubles
When input size triples, time triples
When input size goes up by a factor of 10, so does time

Example: A linear time algorithm runs in 10 sec on input size 10,000
How long to run on input size 1,000,000?

Answer: 1,000,000 / 10,000 = 100 times larger input
Therefore 100 times larger time, or 10 * 100 = 1,000 sec
Or 1,000 / 60 = 16.667 minutes

Predicting Program Times - Quadratic

Basic idea: Given time complexity and sample time(s) can estimate time on larger inputs

Quadratic time: When input size doubles (2x), time quadruples (4x)
Input size goes up by a factor of 10, time goes up 10^2=100 times
Input size goes up *k* times, time goes up k^2 times

Example: A quadratic time algorithm runs in 10 sec on input size 10,000
How long to run on input size 1,000,000?

Answer: 1,000,000 / 10,000 = 100 times larger input
Therefore 100^2 = 10,000 times larger time, or 100,000 sec
Or 100,000 / 60 = 1666.7 minutes (or 27.8 hours)

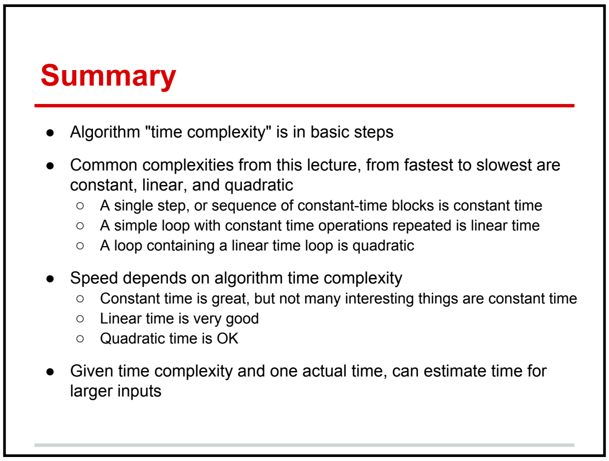
Predicting Program Times - Your Turn

Joe and Mary have created programs to analyze crime statistics, where the input is some data on each resident of a town

- Joe's algorithm is quadratic time
- Mary's algorithm is linear time
- Both algorithms take about 1 minute for a town of size 1000

Both would like to sell their program to the City of Greensboro (population 275,000)

Problem: Estimate how long each program would take to run for Greensboro



Summary

- Algorithm "time complexity" is in basic steps
- Common complexities from this lecture, from fastest to slowest are constant, linear, and quadratic
  - A single step, or sequence of constant-time blocks is constant time
  - A simple loop with constant time operations repeated is linear time
  - A loop containing a linear time loop is quadratic
- Speed depends on algorithm time complexity
  - Constant time is great, but not many interesting things are constant time
  - Linear time is very good
  - Quadratic time is OK
- Given time complexity and one actual time, can estimate time for larger inputs