COMP20010 - Math and Complexity Practice Questions - **Answers**

May 6, 2010

1 Big-O

1.1 Practice

What is big-O complexity of the following?

1. $100N + N^2$. $\boxed{\text{This is } O(N^2).}$

   The question did not ask for an argument why it is of this order. In case it did, here is one possible answer.

   - For all $N > 1$, it is the case that $N^2 > N$.
   - Therefore, for all $N > 1, 100N^2 > 100N$.
   - Thus,
     $$100N + N^2 < 100N^2 + N^2 = 101N^2.$$
   - So, for all $N > 1$, $100N + N^2 < cN^2$ with $c = 101$. From the definition of big-O, this is $O(N^2)$.

2. $3N^5 + 2N + 13N^2 + 9$. $\boxed{O(N^5)}$.

3. $\sqrt{N} + 3N + 100$. $\boxed{O(N)}$.

4. $10^6N + N^2$. $\boxed{O(N^2)}$.

5. $\frac{N}{1+N}$. $\boxed{O(1) \text{ Note for } N > 0 \text{ this is less than } 1}$

6. $\sin N$. $\boxed{O(1) \text{ for same reasons as above}}$.

1.2 Loops within algorithms

What is the big-O complexity of the following methods?

Algorithm 1 : Compute mean and variance of an array size N

```
mean ← 0
var ← 0
{Compute mean}
for i = 1 to N do
  mean ← mean + array[i]
end for
{Compute variance}
for i = 1 to N do
  var ← var + (array[i] - mean)*(array[i] - mean)
end for
print mean, var
```

Algorithm 1 is $O(N)$ because it has two loops both run up to N.

Algorithm 2:

```
for i = 1 to N do
  for j = 1 to 10 N do
    print i + j
  end for
end for
```

Algorithm 2 has complexity $O(N^2)$ because of the two nested loops, both going from 1 to N.

Algorithm 3:

```
for i = 1 to N do
  if factorial(i)> 100 then
    return i
    break
  end if
end for
```

Here `factorial()` is a method which computes the factorial of an number, i.e. `factorial(n)` $= 1 \times 2 \times \cdots \times n$.

Algorithm 3 is $O(1)$, because the loop stops when `factorial(i)` exceeds 100 which is independent of N. No matter how big N grows, this will stop after a fixed number of steps.

Algorithm 4:

```
for i = 1 to N do
  for j = 1 to 10N do
    for k = 1 to N do
      print  k * (i + j)
```

```
    end for
  end for
end for
```

Algorithm 4 is $O(N^3)$ because of three nested loops.

Algorithm 5:

```
for i = 1 to N do
  for j = 1 to i do
    print  i + j
  end for
end for
```

Algorithm 5 is $O(N^2)$. One way to see this is to argue as follows. If the upper limit to the inner loop is replaced with N, the algorithm would clearly be $O(N^2)$. This algorithm runs in less steps. Therefore it too is $O(N^2)$ (remember, big-O is an upper bound).

Algorithm 6:

```
for i = 1 to N do
  temp ← list[i]
  list[i] ← 0
  sort(list)
  list[i] ← temp
end for
```

where `sort` is an $O(N \ln N)$ sorting algorithm.

As stated in the notes, nested loops multiply, so this is $O(N^2 \ln N)$.

1.3 Exponential Growth

1.4 Logarithms

Find the following logarithms (without using a calculator).

1. $\log_2 128$. $\boxed{7}$
2. $\log_3 9$. $\boxed{2}$
3. $\log_{10}(10^5)$. $\boxed{5}$
4. $\log_{100} 100$. $\boxed{1}$
5. $\log_{10} 50$ lies between what two integers? Is it halfway between?

$\log_{10} 10 = 1$ and $\log_{10} 100 = 2$ so $\log_{10} 50$ lies between 1 and 2. It is *not* halfway between, because it is on a log scale. In fact, $\log_{10} 50 = 1.698\cdots$.

Let `database` be a database of names and phone numbers stored in an array, where the array is alphabetically by names. In other words, if $i < j$, then `database[i].name`<=`database[j].name`, where here less than means in alphabetical order.

The following is an algorithm to find the phone number of a particular name. What is the big-O complexity of this algorithm? (`sname` is the string containing the name being searched for.)

Algorithm BinarySearch

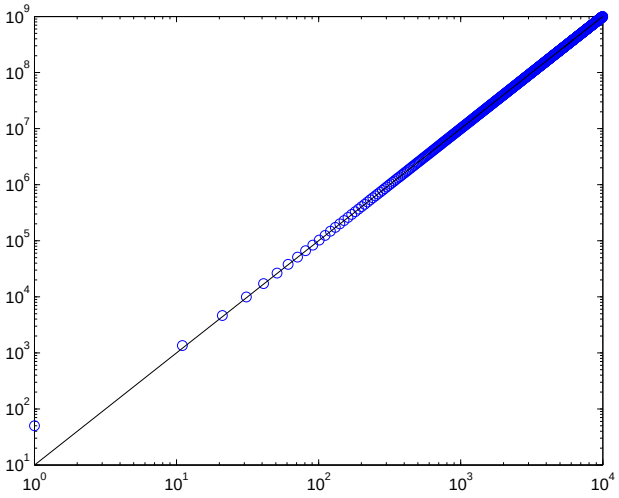
```
min ← 1
max ← N
repeat
  mid ← (min + max) div 2
  if sname > database[mid].name then
    min ← mid +1
  else
    max ← mid −1
  end if
until (database[mid].name = sname) or(min > max);
if (min > max) then
  print ''name not found''
else
  print database[mid].number
end if
```

Hint: How big is the search interval at the start of each iteration of the `repeat` loop? How many times does the `repeat` loop run.

The size of the interval is always `max - min + 1`. At the start this is N, and it is reduced by within 1 of half in each iteration. It is hard to say exactly how many times it runs, but it will be $O(\ln N)$.
To see this, suppose N is a power of 2. Then each iteration reduces the size of the interval by 1/2 and the number of iterations will be less than or equal to $\log_2 N + 1$ (the 1 comes because it may take one more step for `min` to get bigger than `max`. If N is not a power of 2, it will certainly run in less steps than if N were replaced by the next larger power of 2, i.e. less steps than $\log_2 N + 2$.

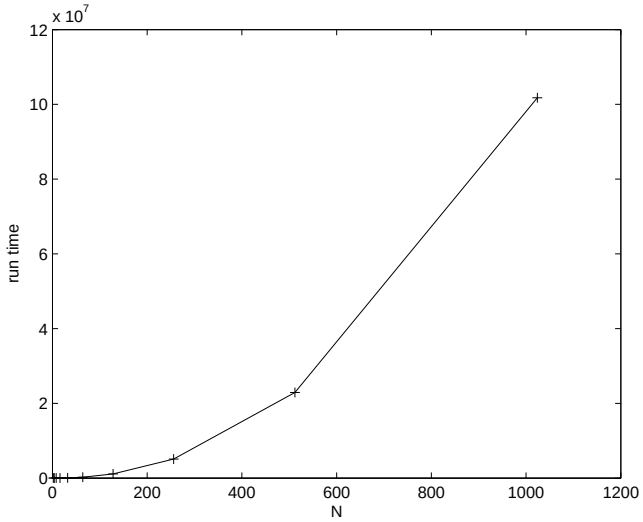
1.5 Log-log plots

1. The plot below shows some data plotted on a log-log plot, and a best-fit line. Is it a power law? What is the formula?

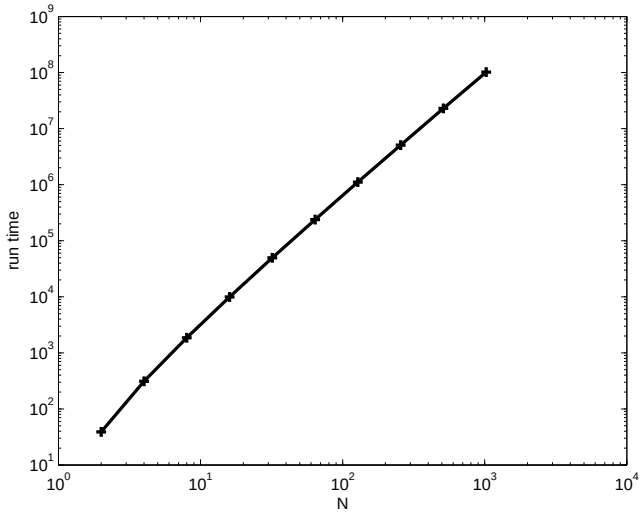


This is a powerlaw, because it is a straight line on a log-log plot. The slope is $\frac{9-1}{4-0} = 2$. The intercept is at 1. Thus, the formula is $y = 10x^2$.

2. The plot below shows the results of some experiments on run times. This could follow a powerlaw.



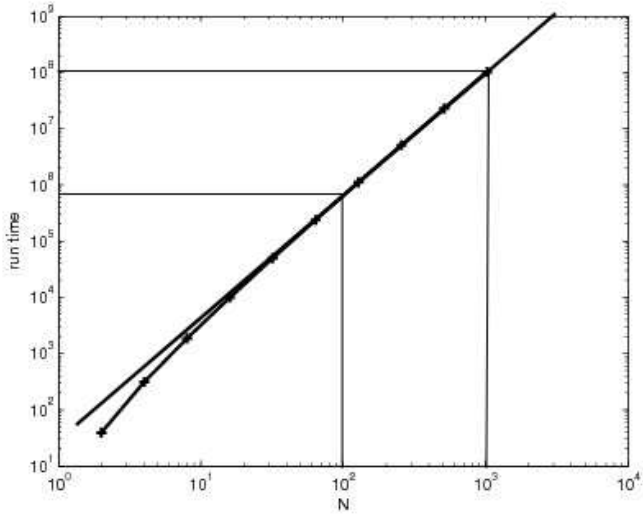
The plot below shows the same data plotted on a log-log plot. Does it exactly follow a power law. Give an approximate formula for the power law for large N.



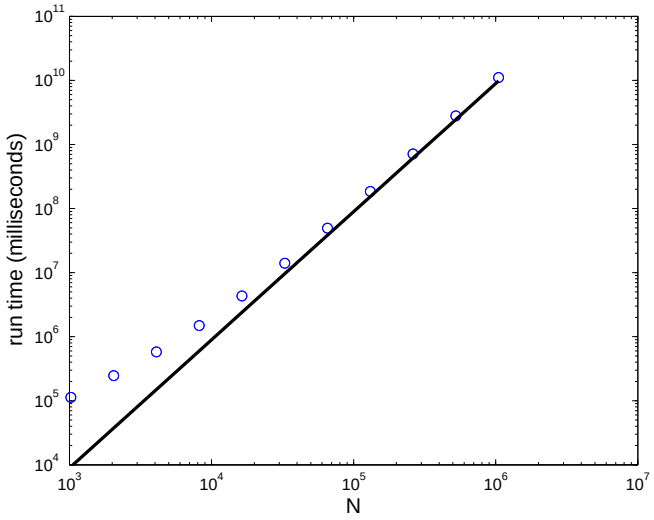
It is not an exact power law, because the line is not straight. However, as N increases, it approaches a straight line. I have sketched the line below, and we can estimate the equation. The slope appears to be about 2.3, and the intercept is at 1.2 on the log scale. Therefore the equation is approximately,

$$\text{run time} = 10^{1.2} N^{2.3}.$$

Note there are hash marks between the labelled points on the axes. These are spaced 0.1 apart on the log scale.



3. Here is the data from the January Mock exam. Give the formula of the best fit powerlaw.



The slope is $\frac{10-4}{6-3} = 2$, the intercept is at 4 on the log scale. The equation is,

$$\text{run time} = 10^4 N^2.$$