Data Structures and Algorithms
HW #4

Nidhi Prajapati
Michael David

November 17, 2022

1

Write $P(\{a, b, c\})$ in a form in which all elements are listed.

Let S = {a, b, c}.
P(S) = {{}, {a}, {b}, {c}, {a, b}, {a, c}, {b, c}, {a, b, c} }

2

How many partitions are there of $\{x \in \mathbb{N} | 1 \leq x \wedge x \geq 10\}$?

115975

3

Let A be a set and m and n be positive integers. Would you say that $A^{m+n} = A^m * A^n$? Give arguments for or against accepting this equality as fact. (Hint: you may want to consider the sets $A * A^2$ and $A^2 * A$)

Yes, if we have the same base but different powers then $A^{m+n} = A^m * A^n$ would satisfy (e.g. $A^{12} = A^5 + A^7$ where which is equal to $A^5 * A^7 = A^{5+7} = A^{12}$). Explanation:

4

Express, in lambda notation, the function which when passed two functions f and g, returns the composition of f and g. (Don't use the predefined $\cdot$ symbol for composition. Actually define the meaning of composition in your answer.)

$\lambda x : f(g(x))$
$edit : \lambda_1 x.(\lambda_2 x.x)$

5

Let $f = \lambda x.4x - 3x^2$?
a. What is $f^0(0.01)$?
$f^0(0.01) = 0.01$
b. What is $f^1(0.01)$?
$f^1(0.01) = 0.0397$
c. What is $f^2(0.01)$?
$f^2(0.01) = 0.154$
d. What is $f^5(0.01)$?
$f^5(0.01) = 0.1713$
e. What is $f^{30}(0.01)$?
$f^{30}(0.01) = 1.286553251$
See image
f. What is $f^{50}(0.01)$ according to Java?
$f^{50}(0.01) = 1.3056006785398018$ according to Java
g. What is $f^{50}(0.01)$ according to your handheld calculator?
$f^{50}(0.01) = 1.063193699$ according to the calculator
h. Explain why we can never write out the exact value, as a decimal number, of $f^{75}(0.01)$ in 10 point Times New Roman on printer paper that is produced on Earth.
There are too many digits after the decimal for us to get an exact value for $f^{75}(0.01)$. We do not know the integers that when divided, result in $f^{75}(0.01)$ thus it is an irrational numbers; we cannot write to exact precision
i. Do you think it will be possible in our lifetime whether we will even know the first digit of the decimal expansion of $f^{100}(0.01)$? Why or why not?
Yes. The uncertainty with irrational numbers is in the latter decimal digits. So we should be able to determine the value of the first digit after the decimal.

6

What is the time complexity of this code fragment? (Use θ-notation)

$\theta(2^n)$

7

What is the time complexity of this code fragment? (Use θ-notation)

If n $\leq$ 1: There will be constant time complexity $\theta(1)$.
If n>1: It will run into an infinite loop; we cannot measure time complexity.

8

What is the time complexity of this code fragment? (Use θ-notation)

For loop: var i = 1; i*j <= n; i++: T(n)= n
For loop: var j= 1; j <= n; j+= j: T(n) = log(n) $\theta(nlogn)$

9

What is the time complexity of this code fragment? (Use θ-notation)

For loop: var i = 1; i<= n; i*= 2: T(n)=log(n)
For loop: var j= 1; j <= i; j++: T(n)=n
$\theta(nlogn)$

10

What is the time complexity of this code fragment? (Use θ-notation)

For loop i++: T(n)=n (Retrospect: should be n^2 because $i <= n * n$)
For loop j *= 2: $T(n) = log_2(n)$
$\theta(nlogn)$

11

What is the time complexity of this code fragment? (Use θ-notation)

For loop i++: T(n)=n
For loop j *= 2: $T(n) = log_2(n)$
$\theta(nlogn)$

12

What is the time complexity of this code fragment? (Use θ-notation)

For loop i/4: $T(n) = log_4(n)$
For loop j++: T(n)=n
$\theta(nlogn)$

13

Give both the best-case and worst-case time complexities of this code fragment. (Use θ-notation)

Best-case scenario: Code never enters if statement nor second for loop: $T(n) = log_2(n)$
B(n): $\theta(logn)$
Worst-case scenario: Program enters second for loop each time
W(n): $\theta(nlogn)$

14

Give both the best-case and worst-case time complexities of this code fragment. (Use θ-notation)

Ternary operator written as if-statements:
if(n==0){
return 1; } else {
if(n%2==0){
return power(x+x, $\frac{n}{2}$); } else {
return x*power(x+x, $\frac{n}{2}$; }
Best-case scenario: n=0
$B(n) : \theta(1)$
Worst-case scenario: $n! = 0, n\%2! = 0$
$W(n) : \theta(logn)$

15

What is the time complexity (in θ-notation) of a procedure to print out the exact value of 2^n, where n is a nonnegative (big) integer? The procedure described is supposed to print a result no matter how large the result may be.

2 is being multiplied n times, so n-1 operations have to take place (ie. $2^3 = 3 * 3 * 3$ is two multiplications) plus a print statement. Minus 1 and plus 1 dropped in $\theta - notation$.
$\theta(n)$

16

An algorithm with time complexity T(n) = n^3 can process a 100-element list on our PC in 10 seconds.
Run-time: $\frac{operations}{\frac{operations}{seconds}} = seconds$ with T(n)=operations
(a)How long would it take to process a 200-element list?
It would take 80 seconds for the algorithm to process a 200-element list.

(b) If we ran the algorithm on a machine that was 10 times faster than our PC, how large of a list could we process in 30 seconds?
We could process a 311 (310.7) element list if the computer was 10 times faster.

(c) How much faster than our PC would a computer have to be in order to process a 1000000000-element list in a time span of 1 hour?
Our PC would have to be $4.6296x10^{16}$ times faster to process a 1000000000-element list in 1 hour.

17

An algorithm with complexity function T(n) = n log(n) processes a 64-element list in three minutes and 12 seconds on our PC.
Run-time: $\frac{operations}{\frac{operations}{seconds}} = seconds$ with T(n)=operations
(a) How long does it take to process a 128-element list?
It would take our PC 7 minutes and 28 seconds to process a 128-element list.

(b) How large of a list could a computer that is 8 times faster than our PC process in 10 seconds?
A computer that is 8 times faster could process a 32-element list in 10 seconds.

(c) How much faster would a computer have to be than our PC to process a list of size 10 in a second?
A computer would have to be 17 (16.61) times faster to process a list of size 10 in one second.

18

An algorithm with time complexity function T(n) = 2n log(n) can process a 32-element list in 2 minutes and 40 seconds on our PC.
Run-time: $\frac{operations}{\frac{operations}{seconds}} = seconds$ with T(n)=operations
(a) How long would it take to process a 64 element list?
It would take 6 minutes and 24 seconds to process a 64 element list.

(b) If we ran the algorithm on a machine that was 4 times faster than our PC, how large of a list could we process in 16 seconds?
We would be able to process a 16-element list in 16 seconds.

19

We have seen that there is little hope of solving problems of size 100 or so with algorithm complexity $\lambda n.2^n$ even when billions of operations can be carried out per seconds. But what about algorithms of complexity $\lambda n.1.1^n$? How do these algorithms compete with quadratic algorithms? In particular, if a billion operations can be performed in one second, up to what problem size will the $\lambda n.1.1^n$ be faster than a $\lambda n.2^n$ algorithm?

Desmos used to graph functions $\lambda n.1.1^n$ increases in run-time significantly slower than $\lambda n.2^n$ for input ranges from 0 to 50 ($\{n|0 \leq n \leq 50\}$). After that, $\lambda n.1.1^n$ grows sharply and intersects with $\lambda n.n^2$ at in input of 96 (95.717). $\lambda n.1.1^n$ results in many more operations for subsequent input sizes (eg. For an input size of 200 (n=200), $\lambda n.n^2$ takes 40,000 operations while $\lambda n.1.1^n$ will take 189,905,277 operations).

20

Rank each of the following functions by growth rate:

* Desmos used to compare functions*
In descending order
1. $\lambda n.\frac{2}{n}$
2. $\lambda n.2^{nn}$
3. $\lambda n.n^n$
4. $\lambda n.n!$
5. $\lambda n.2^n$

6. $\lambda n.n^3$
7. $\lambda n.2^{\frac{n}{2}}$
8. $\lambda n.(log(n))^n$
9. $\lambda n.n^2 log(n)$
10. $\lambda n.n^2$
11. $\lambda n.n^{1.5}$
12. $\lambda n.n(log(n))^2$
13. $\lambda n.n(log(n^2)$
14. $\lambda n.n(log(n)) = \lambda n.log(n^n)$
15. $\lambda n.n$
16. $\lambda n.nlog(log(n))$
17. $\lambda n.\sqrt{n}$
18. $\lambda n.2^{log(n)}$
19. $\lambda n.2(log(n))$
20. $\lambda n.2$
21. $\lambda n.log(\sqrt{n})$