Discrete Structures, Homework n+1=9

Braeden Hunt

Due: 30 April 2021

This homework assignment should be submitted as a single PDF file to Gradescope.

Write a two-page paper describing to me how you have grown as a student, computer scientist, mathematician, engineer, or a researcher in this class, and more generally, in this semester. To support your argument, you should include your homework or writing samples (or excerpts from them) in an appendix as evidence (and reference them!)

If you do not feel that you've grown, explain why.

Remember, style counts. Use complete sentences.

This HW will be graded on the following scale:

- No submission (0%)
- Low pass (70%)
- Pass (90%)
- High Pass (100%)

1 My Growth

I have grown quite a bit as a computer scientist, mathematician, and software developer. The two classes that have given me the most growth this semester have been CSCI 246 (Discrete Structure) and CSCI 232 (Data Structures and Algorithms). I've learned new concepts and how to apply them, as well as skills, both theoretical and practical.

At the beginning of the semester, we were asked to define concepts that we knew before coming into class. As you can see by looking at my answers to H-0 (see Appendix A), I didn't know quite a few of the concepts. Even the ones that I did answer, my definitions weren't very technical or precise. For example, I had no idea what $f(n)$ is $\Theta(g(x))$ meant. However, now I know that it means there exists positive constants a and b such that $ag(x) \leq f(x) \leq bg(x)$ for all $x \geq n_0$. An example of a concept that I can define much better is a tree. I defined a tree at the beginning of the semester as "[a] data structure that can be in the shape of a tree. It can be used for binary searches, as well as sorting." I now define it as "an undirected graph such that there is one and only one path from any vertex to any other vertex. This is equivalent to the tree not having any cycles." This new definition shows my growth as a computer scientist and mathematician as I can more properly define terms rather than by the looks of it as I used to.

I've also learned how to apply these kinds of structures. In CSCI 232, our second outlab required us to implement our own binary search tree. I also learned how to implement other types of graphs, including weighted directed graphs, and then apply Dijkstra's algorithm to find the shortest path between two vertices in a mock GPS program. This is a concept that I first learned about in CSCI 246, but then was able to use that acquired knowledge and apply to solving a real world problem. I believe that one of the most important things you learn in college when it comes to software development and computer science is being able to apply those abstract concepts like graphs to real world problems like navigating streets in a city.

This class has also taught me much of the formal language of mathematics. Before this class, I didn't know the meaning of any of these symbols: $\in$, $\mathbb{R}$, $\forall$, $\exists$, $\implies$, $\iff$, and many more. I've learned that these mean "in," the set of all real numbers, "for all," "there exists," "implies," and "if and only if" respectively. Learning what symbols means not only allows me to read mathematical proofs and theorems, but it also helps me understand them as well as write my own mathematical proofs in a formal and proper manner.

It is easy to compare my first proof that I ever did from H-1 (see Appendix B) to the proofs that I have done more recently to see how much I have progressed in my mathematical skills. In H-1 we were asked to prove that $6\mathbb{Z} \subset 2\mathbb{Z}$. I remember spending quite a bit of time trying to figure this out, and all it ended up being was 4 lines of text and symbols. Nearly 3 months later, H-7 (see Appendix C) asked us to prove that the relation R was an equivalence relation (defined as the relationship between all simple graphs where two graphs g and h are related if and only if g and h have the same number of connected components). At the beginning of the semester, I didn't even know what a relationship was. But when this was assigned, I immediately knew what was needed to be shown. I knew how to formally define R and that R had to be shown to be reflexive, symmetric, and transitive. This proof was 25+ lines with three different parts and two subparts, but it probably took me about the same amount of time as my first proof. If proofs are like a language, I came into this class trying to sound words out to speaking the language fairly fluently.

Finally, I have learned a few practical skills this semester too. One of the biggest is being able to write documents in Latex. I had heard of Latex before this class, but had never used or even seen its use. It was definitely a challenge for me to learn, but I actually enjoy working in it now. I find it much easier to format mathematical language and symbols in it. I also recently learned about the algorithm/psuedocode environment after using it in H-7 (see Appendix D). I wish I knew about this earlier as I could have been using it for writing psuedocode algorithms for my CSCI 232 class. Now I know if I ever need to write it up algorithms to turn in in the future, Latex is the program that I will turn to.

The combination of taking CSCI 246 and CSCI 232 together has spurred on quite a bit of growth in me academically and professionally. My skills as a computer scientist and mathematician have really advanced due to the rigorous nature of this course. My professional skills have also improved by CSCI 232 applying the knowledge that I've learned in this class to the real world. I feel much more confident as a computer science professional now than I did before the start of the semester.

Appendices

A H-0: Baseline Knowledge

1. $f(n)$ is $O(n^2)$.

   Answer The time complexity of the functon f(n) is proportional to the square of n.
2. $f(n)$ is $O(g(n))$.

   Answer The time complexity of the function f(n) is proportional to g(n).
3. $f(n)$ is $\Omega(n^3)$.

   Answer I have not heard of this term
4. $f(n)$ is $\Theta(n \log n)$.

   Answer I have not heard of this term.
5. Binomial Coefficients

Answer I have not heard of this term.

6. Four Color Theorem

   Answer The minimum number of colors it takes to color a map of a planar surface subdivided into areas such that no two adjacent areas have the same color.

7. Graph

   Answer In Graph Theory, a graph is a collective of nodes and edges that connect nodes.

8. Modus Ponens

   Answer I have not heard of this term.

9. Proof by Counter-example

   Answer A mathematical proof that shows that either that a statement is false by showing an example that is can be true, or that a statement is true by showing that the opposite is impossible.

10. Proof by Example

    Answer A mathematical proof that shows that a statement can be true by showing an example of it.

11. Proof by Induction

    Answer I have not heard of this term.

12. Recurrence Relation

    Answer I have not heard of this term.

13. Recursive Algorithm

    Answer An algorithm that splits a problem into smaller and smaller parts, until it reaches a base case. The algorithm calls itself.

14. Searching Algorithms

    Answer An algorithm that searches a set for certain values.

15. Sorting Algorithms

    Answer An algorithm that orders a set of data based on a certain metric.

16. Tree

    Answer A data structure that can be in the shape of a tree. It can be used for binary searches, as well as sorting.

B H-1: First Proof

Prove that $6\mathbb{Z} \subset 2\mathbb{Z}$.

Let $n \in 6\mathbb{Z}$.

By definition of $6\mathbb{Z}$, we know that $n = 6 * i$, where i is an integer.

By factoring, we see that $n = 2 * (3 * i)$.

Therefore, $n = 2 * k$, where $k = 3 * i$, which means that $n \in 2\mathbb{Z}$, as was to be shown.

C H-7: More Advanced Proof

Define a relation R between all simple graphs where two graphs g and h are related (denoted gRh) if and only if g and h have the same number of connected components.

1. Prove that this is an equivalence relation.

   Answer Three properties need to be met in order for a relation to be an equivalence relation, reflexivity, symmetry, and transitivity. For the following proof, let $C(a)$ represent the number of connected components of some graph a. That allows us to define R as $(g, h) \in R \iff C(g) = C(h)$.

   (a) Reflexivity: $(g, g) \in R$
   In order for $(g, g) \in R$, then $C(g)$ has to equal $C(g)$ by definition of R.
   $C(g) = C(g)$ is true by the reflexive property of equals.
   Therefore, R is a reflexive relation.

   (b) Symmetry: $(g, h) \in R \iff (h, g) \in R$
   We need to show two things:
      i. $(g, h) \in R \implies (h, g) \in R$
      Suppose that $(g, h) \in R$. We need to show that $(h, g) \in R$.
      We know that $C(g) = C(h)$ by the definition of R.
      We can rearrange this by the symmetric property of equals to get $C(h) = C(g)$
      Which means $(h, g) \in R$ as was to be shown.
      ii. $(h, g) \in R \implies (g, h) \in R$
      Suppose that $(h, g) \in R$. We need to show that $(g, h) \in R$.
      We know that $C(h) = C(g)$ by the definition of R.
      We can rearrange this by the symmetric property of equals to get $C(g) = C(h)$
      Which means $(g, h) \in R$ as was to be shown.

   Since $(g, h) \in R \implies (h, g) \in R$ and $(h, g) \in R \implies (g, h) \in R$, then $(g, h) \in R \iff (h, g) \in R$ as was to be shown.

   (c) Transitivity: If $(g, h) \in R$ and $(h, i) \in R$, then $(g, i) \in R$
   Suppose $(g, h), (h, i) \in R$
   Then $C(g) = C(h)$ and $C(h) = C(i)$ by the definition of R.

We can substitute $C(i)$ for $C(h)$ in the first equation as $C(h) = C(i)$.
This leaves us with $C(g) = C(i)$, meaning $(g, i) \in R$, as was to be shown.

As R satisfies all properties of equivalence relations, it is an equivalence relation.

D H-7 Algorithm Environment

Algorithm 1 BinarySearchFor($A, value$)

```
$start = 0$
$end = |A| - 1$
**for** $i = 0, i <= \log_2 |A|, i++$ **do**                        ▷ Iterate up to log(n) times
    $mid = (start + end)//2$                                           ▷ Integer Division
    **if** $A[mid] == value$ **then**
        **return** true
    **else if** $A[mid] > value$ **then**                              ▷ If value is above mid,
        $start = mid + 1$                        ▷ reduce the search space to A[(mid+1)...end]
    **else**                                                           ▷ If value is below mid,
        $end = mid - 1$                          ▷ reduce the search space to A[start...(mid-1)]
    **end if**
**end for**
**return** false                                          ▷ If value wasn't found, return false
```