Data Structures
Quiz 1

Cain Susko
January 28, 2022

Queen's University
School of Computing

Review for Quiz 1

these are notes from a live tutorial reviewing the content covered in cisc235 Quiz 1. quiz entry is from 11:30 to 11:35, the quiz is 60 minutes in length. the format is Open Book with Five questions. The time should be tight given the the alleged difficluty of the question. They cover the following:

1. Program Analysis
2. big O-proof
3. coding linked list
4. coding stack
5. coding queue

for coding, be aware of: time, edge cases, and functions are correct

Complexity

we should be able to write the operational complexity as $T(n)$ where T is the function that models the number of operations with n size input.

Big O

we should be able to use Big-O notation which takes the most significant term (ignoring its coeficcient) t^s from $T(n)$ and prove that $T(n) \in O(t^s(n))$ which represents the upperbound of the complexity. we must be able to prove–with the smallest possible c–that $T(n) \leq c * t^s(n)$ for all $n \geq n_0$ where $n \to \infty$ to show that $T(n) \in O(t^s(n))$

$$
\begin{aligned}
T(n) &= 5n^2 - 3n + 20 \\
T(n) &\leq 5n^2 - 0 + 20 \\
20 &\leq 20 * n^2 \\
T(n) &\leq 5n^2 + 20 \leq 5n^2 + 20n^2 \\
T(n) &\leq 25n^2 \quad \forall n \geq n_0 \\
T(n) &\in O(n^2)
\end{aligned}
$$

generally we can ignore any terms with a coefficient < 0. big O is always striving for the lowest upper bound of the complexity.

Big Ω

big Ω defines the highest the lower bound of a program's time complexity that is to say, the highest value of c that satisfies $T(n) \geq c * t^s(n)$ for all $n \geq n_0$
some tricks include:

splitting a higher order term	$n^3 - n \geq n^2 * n - n = n^2$
remove positive term	$3n^2 + 2n \geq 3n^2$
multiply a negative term	$5n^2 - n \ \geq \ 5n^2 - n * n = 4n^2$

when proving Big O or Ω we do not explicitly mention $n_0 = 1$, instead say $n \geq 1$ Furthermore, we must present a complexity proof in the form of

$$T(n) \leq c * t^s(n) \text{ for all } n \geq n_0$$

$$n_0 = x \quad c = y$$

array based list

the key to this type is that it must be able to dynamically allocate consecutive swaths of memory inorder to allow for common & intuitive list like operations using indexes etc.

linked list

this datatype is made up of nodes in memory that store a value and a pointer to the next value. while these arent as intuitive they are computationally less complex. we also use head and tail pointers to define the start and end of a linked list.

stack & queue

a queue uses the FILO principle. an implementation of a queue is a circular queue; which is a fixed size and never ending. see A1/q3 for example.

[] → enqueue → tail[][][][][]head → dequeue → []

a stack uses the LIFO principle a stack can be used for symmetric operations like brace matching

[] → push → head[][] [] ← pop ← head[][]

Sample Questions

1. given $f(n) \in \Theta(n^{12}) \vee g(n) \in \Theta(n^2)$
   what Θ is $f(n) + g(n)$ in

2. write a program that compares 2 linked lists to see if theyre equal

3. implement stack ADT using queue

4. cal (int n):
   sum = 0
   i=1
   for i in range(n):
   sum += i
   return sum

   there are $2n + 3$ ops in this program, thus $t^s(n) = n$ and $T(n) \in O(n)$

   if there are 2 loops with the same (highest) significance but different values $(2m, 2n)$,then $t^s(n) = n + m$

5. how can you improve a program that iterates through a whole list and returns the index of a target value.

   by making it return the index as soon as its found rather than continuing through the list