Quick Sheet

Remember that \( \log(n) \) is assumed to be of base 2 in Computer Science unless otherwise stated.

1. Given fact you can use on exams and assignments: \( 1 < \log(n) < n < n^2 < \cdots < n^p \) for \( n > 3 \)
2. Definition of big-O: \( f(n) = O(g(n)) \) if and only if \( \exists k, n_o \) s.t. \( f(n) \leq k \cdot g(n) \) for \( n > n_o \)
3. Definition of Induction 1: if a property \( P \) holds true for \( k \), and it can be shown that \( P(n) \) implies \( P(n + 1) \) then for all \( n \geq k \) \( P(n) \) holds.
4. Definition of Induction 2: show a base case holds true, say \( P(k) \) is true. Assume \( P(n) \) is true up to some \( n \), using this show \( P(n + 1) \) is true. We now have that for all \( n \geq k \) \( P(n) \) is true.
5. Definition of Induction 3: \( \left( P(k) \text{ and } (P(n) \implies P(n + 1)) \right) \implies \forall n \left( n \geq k \implies P(n) \right) \)

Proof of big-O using Induction:

As we can see from the above definitions of induction, it is used to prove a property for all \( n \geq k \). How are we going to use this to prove \( \exists k, n_o \) s.t. \( f(n) \leq k \cdot g(n) \) for \( n \geq n_o \)? Well the definition of big-O has a for all, when we say for \( n \geq n_o \), we are really saying for all values of \( n \) that are bigger then \( n_o \). This is the part of big-O that benefits from the induction step. This is where the proofs will differ from Math 135, we also want to show \( \exists k, n_o \), and we will do this by placing restrictions on them throughout the induction proof, as once we are done the induction we can say. If I am given a \( k^* \) and \( n^* \) that satisfy these conditions the above proof of induction will prove \( f(n) \leq k^* \cdot g(n) \) for \( n \geq n^* \), and since I have value for \( k \) and \( n_o \) namely \( k^* \) and \( n^* \) they must exist, therefore \( \exists k, n_o \) s.t. \( f(n) \leq k \cdot g(n) \) for \( n \geq n_o \). Holds, therefore \( f(n) \) is \( O(g(n)) \) by definition.

Summary of Steps for Proving \( f(n) = O(g(n)) \)

1. Remove big-O notation from the question you are answering.
   ex. Prove: \( \exists k, n_o \) s.t. \( f(n) \leq k \cdot g(n) \) for \( n > n_o \)
2. Write out the function you are dealing with in terms of constants and recursion.
   ex. \( f(n) = a \text{ (for } n = 0\text{)}, f(n) = f(n - 1) + b \text{ (for } n > 0\text{)} \)
3. Base Case: Show \( f(n) \leq k \cdot g(n) \) holds for some start value of \( n \), likely to work for 0, 1, or 2.
   a. Keep in mind that you can restrict \( k > a + b \), or any other constants to make it work.
   b. Also remember that your base case is linked to \( n_o \), so if you prove it for \( n = 5 \) then \( n_o \geq 5 \) is the restriction you have to impose.
4. Induction Hypothesis: Assume \( f(n) \leq k \cdot g(n) \)
5. Induction Conclusion: Show \( f(n + 1) \leq k \cdot g(n + 1) \)
   a. For a proper proof you need exactly this statement, do not change the coefficient, ending up with \( f(n + 1) \leq 2k \cdot g(n + 1) \) or even \( f(n + 1) \leq (k + 1) \cdot g(n + 1) \) is incorrect.
   b. You can, however restrict \( k \) to be larger than given constants; don’t increase \( n_o \) as it’s linked to the base case of the induction.
6. Finish off by concluding with a statement that explains if the restrictions you have found are followed, the induction proof is complete, and because you have values for \( k \) and \( n_o \) they must exist. Therefore \( f(n) \) is \( O(g(n)) \).
1.
Prove \( f(n) \) is \( O(n^2) \), where \( f(n) = An^2 + Bn\log(n) + n \)

ie. Show \( \exists k, n_o \ f(n) \leq kn^2 \) for \( n > n_o \)

\[
f(n) = An^2 + Bn\log(n) + n
\]

\[
An^2 + Bn\log(n) + n \leq An^2 + Bn(n) + n^2 \quad \text{by fact (1), for } n > 3
\]

\[
An^2 + Bn(n) + n^2 = (A + B + 1)n^2
\]

Since, \( f(n) \leq kn^2 \) for \( n > n_o \) holds if \( k = A + B + 1 \) and \( n_o = 3 \)

There must exist \( k \) and \( n_o \) because we have values that work.

Therefore, \( \exists k, n_o \ f(n) \leq kn^2 \) for \( n > n_o \)

Therefore, \( f(n) \) is \( O(n^2) \)

2.

Prove \( f(n) \) is not \( O(n) \), where \( f(n) = An^2 \)

Assume it is true, ie. Assume \( \exists k, n_o \ f(n) \leq kn \) for \( n > n_o \)

\[
An^2 \leq kn
\]

\[
An \leq k
\]

\[
n \leq \frac{k}{A} \quad \text{there's a contradiction as } n \text{ can be as large as we want.}
\]

Pick a \( B > \frac{k}{A} \) and \( B \geq n_o \), now let \( n = B \)

\[
B \leq \frac{k}{A}
\]

but we picked \( B > \frac{k}{A} \) therefore we have a contradiction, therefore \( (n) \) is not \( O(n) \).

3.

Prove \( f(n) \) is not \( O(\log(n^n)) \), where \( f(n) = An^2 \) where \( A > 0 \)

Assume it is true, ie. Assume \( \exists k, n_o \ f(n) \leq k\log(n^n) \) for \( n > n_o \)
\[ f(n) = An^2 \leq k \log(n^n) \]

\[ An^2 \leq kn \log(n) \]

\[ An \leq k \log(n) \]

\[ \frac{A}{k} \leq \frac{\log(n)}{n} \]

from here we see that there is a contradiction. We can make \( n \) as large as we want; therefore, we can make \( \frac{\log(n)}{n} \) as small as we want by picking larger \( n \), and because we are dealing with efficiency this course will always be dealing with positive constants. Therefore we are safe to say \( k > 0 \) and \( A > 0 \), and hence \( \frac{A}{k} > 0 \), now we need to show this out right.

\[ \frac{A}{k} \leq \frac{\log(n)}{n} \]

let \( B = \frac{A}{k} \) and let \( n = B^B \)

\[ \log(B) \leq B \leq \frac{\log(B^B)}{B^B} \]

\[ B^B \leq \frac{B \log(B)}{\log(B)} \]

\[ B^B \leq B \quad \text{Contradiction by fact (1)} \]

Therefore \( f(n) \) is not \( O(\log(n^n)) \)

4.

Prove \( f(n) \) is \( O(1) \), where \( f(n) = \frac{A \log(n)}{n} \)

ie. Show \( \exists k, n_o \ f(n) \leq k(1) \) for \( n > n_o \)

\[ f(n) = \frac{A \log(n)}{n} \]

\[ 1 < \log(n) < n \text{ for } n > 3 \]

\[ \frac{1}{n} < \frac{\log(n)}{n} < 1 \]

\[ \frac{A \log(n)}{n} \leq A \quad (1) \]

Since, \( f(n) \leq k(1) \) for \( n > n_o \) holds if \( k = A \) and \( n_o = 3 \)
There must exist $k$ and $n_0$ because we have values that work.

Therefore, $\exists k, n_0 \ f(n) \leq k(1)$ for $n > n_0$

Therefore, $f(n)$ is $O(1)$

The other ones follow easily because $A(1) < A \frac{\log(n)}{n} < A \log(n) < A \ n$

5.

Find the Error in the reasoning below.

$O(n^2)$ is $O(n(n - 1))$

$O(n^2)$ is $O(n(n - 2))$

::

$O(n^2)$ is $O(n(n - p))$

::

$O(n^2)$ is $O(n(2))$

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

therefore $O(n^2)$ is $O(n)$.

this reasoning is correct up to $O(n^2)$ is $O(n(n - p))$, where $p$ is a constant. The problem after this is that $p$ is not a constant, $n(n - p) = n(2)$ implies $p = n - 2$ here we see that $p$ depends on a variable therefore making the big-O expression change and causing a problem in our logic.
6. I’d recommend reading the ‘Proof of big-O using Induction’ in the quick sheet before proceeding.

Prove $f(n)$ is $O(n)$, where $f(n) = a$ (if $n = 0$) and $f(n) = b + f(n-1)$ (if $n > 0$)

1. Prove $\exists k, n_0 \text{ st. } f(n) \leq kn$ for $n > n_0$
2. $f(n) = \begin{cases} a & \text{if } n = 0 \\ b + f(n-1) & \text{if } n > 0 \end{cases}$
3. Base Case: Try $n = 0$ here $n_0 \geq 0$
   
   $f(0) = a$, want to show $f(0) \leq k(0) = 0$, but $a > 0$ (represents a constant amount of work)
   
   This doesn’t work so try another base case.
   
   Try $n = 1$ here $n_0 \geq 1$
   
   $f(1) = b + f(1-1) = b + a$ want to show $f(1) \leq k(1)$. 

   This is easy if $k \geq a + b$ so we impose this restriction.
   
   Therefore base case holds if our restrictions are followed.
4. Induction Hypothesis: Assume $f(n) \leq k\ n$ holds up to some $n$.
5. Induction Conclusion: Show $f(n + 1) \leq k(n + 1)$
   
   $f(n + 1) = b + f(n)$
   
   $b + f(n) \leq b + k\ n$ by our assumption
   
   $b + k\ n \leq k + kn$ restrict $k > b$
   
   $k + kn = k(n + 1)$

   Therefore the Induction Conclusion holds if $k > b$.
6. If I am given a $k^* \text{ st. } k^* > a + b$ and $k^* > b$ (which is redundant) say $k^* = a + b + 1$ and a $n^* \text{ st. } n^* \geq 1$ say $n^* = 2$. The above induction proof concludes that $f(n) \leq k^* \ n \text{ for } n > n^*$, and because I have values $k^*$ and $n^*$ $\exists k, n_0 \text{ st. } f(n) \leq kn \text{ for } n > n_0.$

   Therefore $f(n)$ is $O(n)$. 

Prove $T(n)$ is $O(n^2)$, where $T(0)$ and $T(1)$ are $O(1)$, and $T(n) = T(n-2) + f(n)$, where $f(n)$ is $O(n)$.

1. Prove $\exists k, n_o \text{ st. } T(n) \leq kn^2$ for $n > n_o$
   Where $\exists a, n_a \text{ st. } T(0) \leq a(1)$ for $n > n_a$ but because $a$ does not include $n$, it’s simply $\forall n$.
   Where $\exists b, n_b \text{ st. } T(1) \leq b(1)$ for $n > n_b$ but because $b$ does not include $n$, it’s simply $\forall n$
   Where $\exists k_f, n_f \text{ st. } f(n) \leq k_f n$ for $n > n_f$
   
   2. $T(n) = \begin{cases} 
   a & \text{if } n = 0 \\
   b & \text{if } n = 1 \text{ note how } f(n) \text{ is left in the equation, this is because we only have information about } f(n) \text{ above } n_f, \text{ but we’ll use this later.} \\
   T(n-2) + f(n) & \text{if } n > 1 
   \end{cases}$

3. Base Case: try \( n = 2 \) here \( n_o \geq 2 \)
   $T(2) = T(2-2) + f(2) = a + f(2) \leq a + k_f(2)$
   Want $T(2) \leq k2^2 = 4k$, restrict $k > k_f$ and $k > a$ then
   $a + k_f(2) \leq k + 2k < 4k$
   Therefore $T(2) < k2^2$ holds with our restrictions.
   But we need two base cases because we are stepping down the recursion by two \((T(n-2))\)
   try \( n = 3 \)
   $T(3) = T(3-2) + f(3) = b + f(3) \leq b + k_f(3)$
   Want $T(3) \leq k3^2 = 9k$, restrict $k > k_f$ and $k > b$ then
   $b + k_f(3) \leq k + 3k < 9k$
   Therefore $T(3) < k3^2$ holds with one more restrictions.

7. Induction Hypothesis: Assume $T(n) \leq kn^2$ up to some $n$.

8. Induction Conclusion: Show $T(n + 1) \leq k(n + 1)^2 = kn^2 + 2kn + k$
   $T(n + 1) = T(n - 1) + f(n + 1)$
   $T(n - 1) + f(n + 1) \leq k(n - 1)^2 + f(n + 1)$ by Indo Hypo
   $k(n - 1)^2 + f(n + 1) \leq k(n - 1)^2 + k_f(n + 1)$
   $f(n)$ is $O(n)$
   $k(n - 1)^2 + k_f(n + 1) = kn^2 - 2kn + k + k_f n + k_f$
   If we add the restriction that $k > k_f$ (yes I’ve done this already) we get.
   $kn^2 - 2kn + k + k_f n + k_f \leq kn^2 - 2kn + k + kn + k$
   $kn^2 - 2kn + k + k_f n + k \leq kn^2 - kn + 2k$
   $kn^2 - kn + 2k \leq kn^2 - kn + 2kn$
   $kn^2 - kn + 2kn \leq kn^2 + kn$
   $kn^2 + kn \leq kn^2 + 2kn + k$
   $kn^2 + 2kn + k = k(n + 1)^2$
   This proves our Induction Conclusion under our restrictions.

9. Therefore if I’m given values $k^*$ and $n^*$ that follow the restrictions I’ve set down, the above induction proves $T(n) \leq k^*n^2$ for $n > n^*$. From here it follows that $T(n)$ is $O(n^2)$