Proving a bound by Induction

- Recurrence to solve: \( T(n) = 3T(n/3) + n \)
- Guess at a solution: \( T(n) = O(n \lg n) \)

**Proof steps:**
- Rewrite claim to remove big-O: \( T(n) \leq cn \lg n \) for some \( c \geq 0 \).
- “Assume” \( T(n') \leq cn' \lg n' \) for all \( n' < n \).
- Prove that the claim holds for \( n \). Here is the proof

\[
T(n) = 3T(n/3) + n \\
\leq 3(c(n/3) \lg(n/3)) + n \quad \text{(by inductive hypothesis since } n/3 < n) \\
= cn(\lg n - \lg 3) + n \\
= cn \lg n + n - cn \lg 3
\]

- Now we really want to choose \( c \) so that this last line is \( \leq cn \lg n \)
- Equivalently, we really want to choose \( c \) so that \( n - cn \lg 3 \leq 0 \)
- Equivalently, we really want to choose \( c \) so that \( c \lg 3 > 1 \)
- \( c = 1 \) works and completes the proof, as now \( n \lg n + n(1 - \lg 3) \leq n \lg n \)
Recurrences with a big-O in the $f(n)$

- Recurrences describing running times often have a big-O in the non-recursive term
- Consider $T(n) = 2T(n/2) + O(n)$
- What does the $O(n)$ mean?
Recurrences with a big-O in the $f(n)$

- Recurrences describing running times often have a big-O in the non-recursive term
- Consider $T(n) = 2T(n/2) + O(n)$
- What does the $O(n)$ mean?
- The $O(n)$ is describing an algorithm e.g. merge, that runs in time $kn$ for some $k > 0$ that we don’t get to pick.

Claim: $T(n) = O(n \lg n)$

Question: What do this big-O mean?
Recurrences with a big-O in the $f(n)$

- Recurrences describing running times often have a big-O in the non-recursive term
- Consider $T(n) = 2T(n/2) + O(n)$
- What does the $O(n)$ mean?
- The $O(n)$ is describing an algorithm e.g. merge, that runs in time $kn$ for some $k > 0$ that we don’t get to pick.

Claim: $T(n) = O(n \lg n)$

Question: What do this big-O mean?

Answer: $T(n) \leq cn \lg n$ for some $c > 0$, which we do get to pick.
Mechanics of Proof

Claim: The recurrence $T(n) = 2T(n/2) + kn$ has solution $T(n) \leq cn \lg n$.

Proof: Use mathematical induction. The base case (implicitly) holds (we didn’t even write the base case of the recurrence down).

Inductive step:

$$T(n) = 2T(n/2) + kn$$
$$\leq 2 \left( c \frac{n}{2} \lg \left( \frac{n}{2} \right) \right) + kn$$
$$= cn(\lg n - 1) + kn$$
$$= cn \lg n + kn - cn$$

Now we want this last term to be

$$\leq cn \lg n$$

, so we need $kn - cn \leq 0$

$$kn - cn \leq 0$$
$$\Leftrightarrow (k - c)n \leq 0$$
$$\Leftrightarrow (k - c) \leq 0$$
$$\Leftrightarrow k \leq c$$
Is $k \leq c$

- Recall that $k$ is given to us (we don’t choose it).
- We get to choose $c$.
- So if we choose $c = k$, then we have satisfied $c \leq k$, and the proof is complete.
Proof subtlety

Sometimes we have the correct solution, but the proof by induction doesn’t work

- Consider \( T(n) = 4T(n/2) + n \)
- By the master theorem, the solution is \( O(n^2) \)

Proof by induction that \( T(n) \leq cn^2 \) for some \( c > 0 \).

\[
T(n) = 4T(n/2) + n \\
\leq 4 \left( c \left( \frac{n}{2} \right)^2 \right) + n \\
= cn^2 + n
\]

Now we want this last term to be

\[ \leq cn^2 \]

, so we need \( n \leq 0 \)
Sometimes we have the correct solution, but the proof by induction doesn’t work.

- Consider $T(n) = 4T(n/2) + n$
- By the master theorem, the solution is $O(n^2)$

Proof by induction that $T(n) \leq cn^2$ for some $c > 0$.

$$T(n) = 4T(n/2) + n$$
$$\leq 4 \left( c \left( \frac{n}{2} \right)^2 \right) + n$$
$$= cn^2 + n$$

Now we want this last term to be
$$\leq cn^2$$
, so we need $n \leq 0$

UhOh No way is $n \leq 0$. What went wrong?
General Issue with proofs by induction

- Sometimes, you can’t prove something by induction because it is too weak. So your inductive hypothesis is not strong enough.
- The fix is to prove something stronger
- We will prove that $T(n) \leq cn^2 - dn$ for some positive constants $c,d$ that we get to chose.
- We chose to add the $-dn$ because we noticed that there was an extra $n$ in the previous proof.
The proof

Claim: \( T(n) \leq cn^2 - dn \) for some positive constants \( c, d \)

Proof:

\[
T(n) = 4T(n/2) + n \\
\leq 4 \left( c \left( \frac{n}{2} \right)^2 - d \frac{n}{2} \right) + n \\
= cn^2 - 2dn + n \\
= (cn^2 - dn) + (n - dn) \\
= (cn^2 - dn) + (1 - d)n
\]

Now we want this last term to be

\[
\leq cn^2 - dn
\]

so we need \( (1 - d)n \leq 0 \). Just choose \( d = 2 \). We can choose \( c \) to be anything, say 1

Conclusion

\[
T(n) \leq cn^2 - 2n = O(n^2)
\]