What's the big idea?

Come up with a rigorous method for figuring out work done in recursive algorithms. We added up all work in.

Proof of Master Method Page 1

\[
T(n) =
\begin{cases}
  \Theta(n^2) & \text{if } a = \frac{b^p}{p} \\
  \Theta(n \log n) & \text{if } a < \frac{b^p}{p} \\
  \Theta(n^p) & \text{if } a > \frac{b^p}{p}
\end{cases}
\]
Loop Invariants: Prove loops are correct

setup
while (condition) {
    Stuff
}
Great output

Parts of Proof: (guess)

1. State invariant: this is thing that is true after every

2. Base Case: Show invariant is true before loop starts

3. Maintenance: Show if invariant is true before an iteration, it is also true after.

4. Termination: argue loop ends. State what invariant tells us given ending conditions
Algorithm 1: Smallest($A, n$)

**Input**: Array $A$ of integers of length $n$

**Output**: Smallest value of $A$

1. $s = A[1]; i = 2$;
2. **while** $i \leq n$ **do**
3.   **if** $A[i] < s$ **then**
4.     $s = A[i]$;
5.   **end**
6.   $i++ = 1$;
7. **end**
8. return $s$;

**Solution** We will prove the following loop invariants: (i) $s \in A[1 : i - 1]$, (ii) $s \leq A[k]$ for $k$ such that $1 \leq k \leq i - 1$.

Base case: Before the loop starts, $i = 2$ and $s = A[1]$, so (i) and (ii) hold.

Maintenance: Let $i_a$, $s_a$ be initial values, and $i_b$, $s_b$ be new values at the end of the loop. Then

\[ i_b = i_a + 1, \]
\[ s_b = \min\{s_a, A[i_a]\}. \]  

By assumption (ii), $s_a$ is less than all values of $A[1 : i_a - 1]$, so combining this assumption with Eq. 2 and the fact that $A[i_a] = A[i_b - 1]$, (ii) is maintained. If $s_b = s_a$ then $s_b \in A[1 : i_a - 1] \subset A[1 : i_b - 1]$ by assumption (i). Otherwise, $s_b = A[i_a] = A[i_b - 1]$. Either way, (ii) holds.

Termination: $i$ increases with each loop, so the loop will terminate at $i = n$. Then we have $s$ is the smallest value of the whole array by (i) and (ii).
Q: Prove that \( y = c \) at end of loop:

\[
y = 0; \quad x = c;
\]

\[
\text{while (} x > 0 \text{) } \{
    x--; \quad y++;
\}
\]

Invariant: \( x + y = c \)

Base: \( y = 0, \ x = c \) ✓

Maintenance: \( y_{b} = y_{a} + 1, \ x_{b} = x_{a} - 1 \)

\[
x_{b} + y_{b} = y_{a} + 1 + x_{a} - 1 = x_{a} + y_{a} = c
\]

By inductive assumption

Termination: \( x \) gets smaller at each step, so terminates when \( x = 0 \). Since \( x + y = c \) \( \Rightarrow \) \( y = c \)