Recipe

When asked to design an algorithm you must always include the following 4 things unless noted otherwise:

1. Algorithm: Briefly summarize the key ideas of your algorithm in English. What is the main idea or insight of your algorithm? What approach did you use to solve the problem?
2. Write pseudo-code. This is not C or Java code but rather a high level language that illustrates the details of your algorithm.
3. State and justify the time complexity of your algorithm using big-O notation. If your algorithm has multiple parts (e.g. sort all data and then do some work) analyze each part and then make a final statement about the overall time complexity.
4. Give a proof of correctness. You may use any proof technique as long as your logic is correct (i.e. induction, direct proof, proof by contradiction, etc). It's not always immediately clear that algorithms work. The purpose of the proof is to rigorously convince your reader that your algorithm produces the correct result.

Examples

A) Fast multiplication: easy case

First we consider that n is a power of 2.

1. Algorithm. We recursively calculate $a^n = \left(a^{\frac{n}{2}}\right)^2$ until $n = 1$ where $a^1 = a$.

2. Pseudo-code.

```
procedure mult(a, n):
  if n == 1:
    return a
  else:
    b = mult(a, n / 2)
    return b * b
```

3. Time complexity. Let $T(n)$ be the number of multiplications. At each step n is divided by 2 and we do one multiplication:

$$T(n) = T\left(\frac{n}{2}\right) + 1$$

So, by the Master Theorem, $T(n) = O(\log n)$.

4. Proof of correctness. If $n = 2^p$, let's prove by induction on p that $\texttt{mult}(a, 2^p) = a^{2^p}$.

- *Base case* $p = 0$. We have $n = 2^0 = 1$ and $\texttt{mult}(a, 1) = a = a^1$. ok!
- *Induction hypothesis.* Let's assume that $\texttt{mult}(a, 2^p) = a^{2^p}$ for a random $p \geq 0$.
- *Induction step.* Let's prove $\texttt{mult}(a, 2^{p+1}) = a^{2^{p+1}}$.

  $\texttt{mult}(a, 2^{p+1}) = b^2$ where $b = \texttt{mult}(a, 2^p)$. And we know from the induction hypothesis that $\texttt{mult}(a, 2^p) = a^{2^p}$. Therefore $\texttt{mult}(a, 2^{p+1}) = b^2 = \left(a^{2^p}\right)^2 = a^{2^{p+1}}$.
- *Conclusion.* We have proven that $\forall p \geq 0$, $\texttt{mult}(a, 2^p) = a^{2^p}$.

B) Fast multiplication: general case

Here we no longer assume n to be a power of 2.

1. Algorithm. Recursively calculate:

$$a^n = \begin{cases} \left(a^{\frac{n}{2}}\right)^2 & \text{if } n \text{ is even} \\ a \times a^{n-1} & \text{if } n \text{ is odd} \\ 1 & \text{if } n = 1 \end{cases}$$

2. Pseudo-code.

```
procedure mult(a, n):
  if n == 1:
    return a
  if n is odd
    return a * mult(a, n - 1)
  else:
    b = mult(a, n / 2)
    return b * b
```

3. Time complexity. n is at most a multiplicative factor away from some power of 2, so the time complexity is not affected by n not being a power of 2. Therefore, we still have $T(n) = O(\log n)$.

4. Proof of correctness. If $n = 2^p$, let's prove by induction on p that $\texttt{mult}(a, 2^p) = a^{2^p}$.

- *Base case* $p = 0$. We have $n = 2^0 = 1$ and $\texttt{mult}(a, 1) = a = a^1$. ok!
- *Induction hypothesis.* Let's assume that we have $\forall k \in \{1, 2, ..., n\}$, $\texttt{mult}(a, k) = a^k$ (strong induction).
- *Induction step.* Let's prove $\texttt{mult}(a, n+1) = a^{n+1}$.
  - If n is even, then

$$\begin{aligned} \texttt{mult}(a, n+1) &= \left(\texttt{mult}\left(a, \frac{n+1}{2}\right)\right)^2 \\ &= \left(a^{\frac{n+1}{2}}\right)^2 \quad \text{(ind. hyp.)} \\ &= a^{n+1} \end{aligned}$$

  - If n is odd, then

$$\begin{aligned} \texttt{mult}(a, n+1) &= a \times \texttt{mult}\,(a, n) \\ &= a \times a^n \quad \text{(ind. hyp.)} \\ &= a^{n+1} \end{aligned}$$

- *Conclusion.* We have proven that $\forall n \geq 1$, $\texttt{mult}(a, n) = a^n$.