Discussion 2

- Induction proof
- Asymptotic Notation and Execution Time

Induction Proof

1. Use mathematical induction to show that: $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$ for all nonnegative integer n

Induction Proof

1. Use mathematical induction to show that: $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$ for all nonnegative integer n

Base case. For $n = 0$, we have LHS $= 2^0 = 1$ and RHS $= 2^1 - 1 = 1 =$ LHS

Induction Proof

1. Use mathematical induction to show that: $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$ for all nonnegative integer n

Base case. For $n = 0$, we have LHS $= 2^0 = 1$ and RHS $= 2^1 - 1 = 1 =$ LHS

Inductive step. Assume the claim holds for $n = k$, that is: $\sum_{i=0}^{k} 2^i = 2^{k+1} - 1$

Prove it holds for $n = k + 1$, that is: $\sum_{i=0}^{k+1} 2^i = 2^{k+2} - 1$

Induction Proof

1. Use mathematical induction to show that: $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$ for all nonnegative integer n

Base case. For $n = 0$, we have LHS $= 2^0 = 1$ and RHS $= 2^1 - 1 = 1 =$ LHS

Inductive step. Assume the claim holds for $n = k$, that is: $\sum_{i=0}^{k} 2^i = 2^{k+1} - 1$

Prove it holds for $n = k + 1$, that is: $\sum_{i=0}^{k+1} 2^i = 2^{k+2} - 1$

$$\text{LHS} = \sum_{i=0}^{k+1} 2^i$$

$$= \sum_{i=0}^{k} 2^i + 2^{k+1}$$

$$= 2^{k+1} - 1 + 2^{k+1} \quad \text{(apply inductive assumption)}$$

$$= 2 \cdot 2^{k+1} - 1 = 2^{k+2} - 1 = \text{RHS}$$

Induction Proof

1. Use mathematical induction to show that: $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$ for all nonnegative integer n

Base case. For $n = 0$, we have LHS $= 2^0 = 1$ and RHS $= 2^1 - 1 = 1 =$ LHS

Inductive step. Assume the claim holds for $n = k$, that is: $\sum_{i=0}^{k} 2^i = 2^{k+1} - 1$

Prove it holds for $n = k + 1$, that is: $\sum_{i=0}^{k+1} 2^i = 2^{k+2} - 1$

LHS $= \sum_{i=0}^{k+1} 2^i$

$= \sum_{i=0}^{k} 2^i + 2^{k+1}$

$= 2^{k+1} - 1 + 2^{k+1}$ (apply inductive assumption)

$= 2 \cdot 2^{k+1} - 1 = 2^{k+2} - 1 =$ RHS

Conclusion. The claim holds for $n = k + 1$. From the base case and the inductive step, the claim holds for $n \geq 0$

Induction Proof

2. Use mathematical induction to show that: $2 \cdot 4^n \geq n2^n + 3^{n+1}$ for all integer $n \geq 3$

Induction Proof

2. Use mathematical induction to show that: $2 \cdot 4^n \geq n2^n + 3^{n+1}$ for all integer $n \geq 3$

Base case. For $n = 3$, we have LHS $= 2 \cdot 4^3 = 128$, RHS $= 3 \cdot 2^3 + 3^{3+1} = 24 + 81 = 105$. Since $128 > 105$, the base case holds.

Induction Proof

2. Use mathematical induction to show that: $2 \cdot 4^n \geq n2^n + 3^{n+1}$ for all integer $n \geq 3$

Base case. For $n = 3$, we have LHS $= 2 \cdot 4^3 = 128$, RHS $= 3 \cdot 2^3 + 3^{3+1} = 24 + 81 = 105$. Since $128 > 105$, the base case holds.

Inductive step. Let k be any integer such that $k \geq 3$. Assume the inequality holds for $n = k$, that is: $2 \cdot 4^k \geq k2^k + 3^{k+1}$

Prove it holds for $n = k + 1$, that is: $2 \cdot 4^{k+1} \geq (k+1)2^{k+1} + 3^{k+2}$

Induction Proof

2. Use mathematical induction to show that: $2 \cdot 4^n \geq n2^n + 3^{n+1}$ for all integer $n \geq 3$

Base case. For $n = 3$, we have LHS $= 2 \cdot 4^3 = 128$, RHS $= 3 \cdot 2^3 + 3^{3+1} = 24 + 81 = 105$. Since $128 > 105$, the base case holds.

Inductive step. Let k be any integer such that $k \geq 3$. Assume the inequality holds for $n = k$, that is: $2 \cdot 4^k \geq k2^k + 3^{k+1}$

Prove it holds for $n = k + 1$, that is: $2 \cdot 4^{k+1} \geq (k+1)2^{k+1} + 3^{k+2}$

$$\begin{aligned}
\text{LHS} &= 2 \cdot 4^{k+1} \\
&= 4 \cdot 2 \cdot 4^k \\
&\geq 4 \cdot (k2^k + 3^{k+1}) \quad \text{(apply inductive assumption)} \\
&= 4 \cdot k2^k + 4 \cdot 3^{k+1} \\
&\geq 2 \cdot 2 \cdot k2^k + 3 \cdot 3^{k+1} = 2 \cdot k2^{k+1} + 3^{k+2} \\
&\geq (k+1)2^{k+1} + 3^{k+2} \quad \text{(because } 2k \geq k+1\text{, for } k \geq 1\text{)}
\end{aligned}$$

Induction Proof

2. Use mathematical induction to show that: $2 \cdot 4^n \geq n2^n + 3^{n+1}$ for all integer $n \geq 3$

Base case. For $n = 3$, we have LHS $= 2 \cdot 4^3 = 128$, RHS $= 3 \cdot 2^3 + 3^{3+1} = 24 + 81 = 105$. Since $128 > 105$, the base case holds.

Inductive step. Let k be any integer such that $k \geq 3$. Assume the inequality holds for $n = k$, that is: $2 \cdot 4^k \geq k2^k + 3^{k+1}$

Prove it holds for $n = k + 1$, that is: $2 \cdot 4^{k+1} \geq (k+1)2^{k+1} + 3^{k+2}$

$$
\begin{aligned}
\text{LHS} &= 2 \cdot 4^{k+1} \\
&= 4 \cdot 2 \cdot 4^k \\
&\geq 4 \cdot (k2^k + 3^{k+1}) \quad \text{(apply inductive assumption)} \\
&= 4 \cdot k2^k + 4 \cdot 3^{k+1} \\
&\geq 2 \cdot 2 \cdot k2^k + 3 \cdot 3^{k+1} = 2 \cdot k2^{k+1} + 3^{k+2} \\
&\geq (k+1)2^{k+1} + 3^{k+2} \quad \text{(because } 2k \geq k+1 \text{, for } k \geq 1)
\end{aligned}
$$

Conclusion. The claim holds for $n = k + 1$. From the base case and the inductive step, the claim holds for $n \geq 0$

Induction Proof

3. We define a sequence U_n as follows:

$$U_0 = U_1 = 1$$

$$U_n = \frac{1}{8} U_{n-1}^2 + \frac{1}{8} U_{n-2} + 1 \quad \text{for } n \geq 2$$

Use mathematical induction to prove that: $U_n < 2 \quad \text{for } n \geq 0$

Induction Proof

3. We define a sequence U_n as follows:

$$U_0 = U_1 = 1$$

$$U_n = \frac{1}{8}U_{n-1}^2 + \frac{1}{8}U_{n-2} + 1 \quad \text{for } n \geq 2$$

Use mathematical induction to prove that: $U_n < 2 \quad \text{for } n \geq 0$

Base case. For $n = 0$, we have $U_0 = 1 < 2$ and for $n = 1$, $U_1 = 1 < 2$

Induction Proof

3. We define a sequence U_n as follows:

$$U_0 = U_1 = 1$$

$$U_n = \frac{1}{8}\,U_{n-1}^2 + \frac{1}{8}\,U_{n-2} + 1 \quad \text{for } n \geq 2$$

Use mathematical induction to prove that: $U_n < 2 \quad \text{for } n \geq 0$

Base case. For $n = 0$, we have $U_0 = 1 < 2$ and for $n = 1$, $U_1 = 1 < 2$

Inductive step. Assume the claim holds for $n \leq k$. Prove it holds for $n = k + 1$, that is: $U_{k+1} < 2$

Induction Proof

3. We define a sequence U_n as follows:

$$U_0 = U_1 = 1$$

$$U_n = \frac{1}{8} U_{n-1}^2 + \frac{1}{8} U_{n-2} + 1 \quad \text{for } n \geq 2$$

Use mathematical induction to prove that: $U_n < 2 \quad \text{for } n \geq 0$

Base case. For $n = 0$, we have $U_0 = 1 < 2$ and for $n = 1$, $U_1 = 1 < 2$

Inductive step. Assume the claim holds for $n \leq k$. Prove it holds for $n = k + 1$, that is: $U_{k+1} < 2$

$$\text{LHS} = U_{k+1} = \frac{1}{8} U_k^2 + \frac{1}{8} U_{k-1} + 1$$

$$< \frac{1}{8} \cdot 2^2 + \frac{1}{8} \cdot 2 + 1 \text{(apply inductive assumption for } n = k \text{ and } n = k - 1\text{)}$$

$$= \frac{7}{4} < 2 = \text{RHS}$$

Induction Proof

3. We define a sequence U_n as follows:

$$U_0 = U_1 = 1$$

$$U_n = \frac{1}{8}U_{n-1}^2 + \frac{1}{8}U_{n-2} + 1 \quad \text{for } n \geq 2$$

Use mathematical induction to prove that: $U_n < 2 \quad$ for $n \geq 0$

Base case. For $n = 0$, we have $U_0 = 1 < 2$ and for $n = 1$, $U_1 = 1 < 2$

Inductive step. Assume the claim holds for $n \leq k$. Prove it holds for $n = k + 1$, that is: $U_{k+1} < 2$

$$\text{LHS} = U_{k+1} = \frac{1}{8}U_k^2 + \frac{1}{8}U_{k-1} + 1$$

$$< \frac{1}{8} \cdot 2^2 + \frac{1}{8} \cdot 2 + 1 \text{(apply inductive assumption for } n = k \text{ and } n = k - 1)$$

$$= \frac{7}{4} < 2 = \text{RHS}$$

Conclusion. The claim holds for $n = k + 1$. From the base case and the inductive step, the claim holds for $n \geq 0$

Asymptotic Notation and Execution Time

1. Give an O estimate for the number of operations (where an operation is an addition or a multiplication) used in this segment of an algorithm.

$t := 0$

for $i := 1$ to 3

  for $j := 1$ to 4

    $t := t + i*j$

Asymptotic Notation and Execution Time

1. Give an O estimate for the number of operations (where an operation is an addition or a multiplication) used in this segment of an algorithm.

$t := 0$

for $i := 1$ to 3

for $j := 1$ to 4

$\boxed{t := t + i*j}$

executes 12 times

Asymptotic Notation and Execution Time

1. Give an O estimate for the number of operations (where an operation is an addition or a multiplication) used in this segment of an algorithm.

$t := 0$

for $i := 1$ to 3

for $j := 1$ to 4

$\boxed{t := t + i*j}$

executes 12 times

The number of operations is $O(1)$. Specifically, there are 24 additions and multiplications.

Asymptotic Notation and Execution Time

2. Give asymptotic running time for the pseudo code below using O notation.

$$
\begin{aligned}
&\textbf{for } i := n/2 \text{ to } n \\
&\qquad x := 2x + 7 \\
&\textbf{for } j := 1 \text{ to } 3n \\
&\qquad x := 2x + 7
\end{aligned}
$$

Asymptotic Notation and Execution Time

2. Give asymptotic running time for the pseudo code below using O notation.

$$
\begin{aligned}
&\textbf{for } i := n/2 \text{ to } n \\
&\qquad x := 2x + 7 \\
&\textbf{for } j := 1 \text{ to } 3n \\
&\qquad x := 2x + 7
\end{aligned}
$$

The first for loop runs $n/2 + 1$ times. The second for loop runs $3n$ times. Total: $O(n)$

Asymptotic Notation and Execution Time

3. Give asymptotic running time for the pseudo code below using O notation.

$$
\begin{array}{l}
\textbf{for } i := 1 \text{ to } n \\
\quad j := 1 \\
\quad \textbf{while } j < n \\
\quad\quad \textbf{print}(\text{"boo"}) \\
\quad\quad j = 2j
\end{array}
$$

Asymptotic Notation and Execution Time

3. Give asymptotic running time for the pseudo code below using O notation.

$$
\begin{array}{l}
\textbf{for } i := 1 \text{ to } n \\
\quad j := 1 \\
\quad \textbf{while } j < n \\
\quad\quad \textbf{print}(\text{"boo"}) \\
\quad\quad j = 2j
\end{array}
\longrightarrow \quad j = 1, 2, 4, 8, \ldots, 2^k
$$

Asymptotic Notation and Execution Time

3. Give asymptotic running time for the pseudo code below using O notation.

$$
\begin{array}{l}
\textbf{for } i := 1 \text{ to } n \\
\quad j := 1 \\
\quad \textbf{while } j < n \qquad \longrightarrow \qquad j = 1, 2, 4, 8, \ldots, 2^k \\
\quad\quad \textbf{print}(\text{"boo"}) \\
\quad\quad j = 2j
\end{array}
$$

Inner loop runs $O(\log n)$ times. Outer loop runs n times. Total: $O(n \log n)$

Asymptotic Notation and Execution Time

4. For the following pseudo-code, give:

a. Exact value for the number of times "OK" is printed

b. Asymptotic value for the number of times "OK" is printed using Θ notation

```
for i := 1 to n
    for j := i to n
        print("OK")
```

Asymptotic Notation and Execution Time

4. For the following pseudo-code, give:

a. Exact value for the number of times "OK" is printed

b. Asymptotic value for the number of times "OK" is printed using Θ notation

for i := 1 to n

  for j := i to n → For any i, executes $n - i + 1$ times

    print("OK")

Asymptotic Notation and Execution Time

4. For the following pseudo-code, give:

a. Exact value for the number of times "OK" is printed

b. Asymptotic value for the number of times "OK" is printed using Θ notation

for $i := 1$ to n

  for $j := i$ to n → For any i, executes $n - i + 1$ times

    print("OK")

a. The number of times "OK":

$$T(n) = \sum_{i=1}^{n} (n - i + 1) = \sum_{i=1}^{n} (n+1) - \sum_{i=1}^{n} i = n(n+1) - \frac{n(n+1)}{2} = \frac{n(n+1)}{2}$$

Asymptotic Notation and Execution Time

4. For the following pseudo-code, give:

a. Exact value for the number of times "OK" is printed

b. Asymptotic value for the number of times "OK" is printed using Θ notation

for i := 1 to n

  for j := i to n → For any i, executes $n - i + 1$ times

    print("OK")

a. The number of times "OK":

$$T(n) = \sum_{i=1}^{n} (n - i + 1) = \sum_{i=1}^{n} (n + 1) - \sum_{i=1}^{n} i = n(n+1) - \frac{n(n+1)}{2} = \frac{n(n+1)}{2}$$

b. $T(n) = \dfrac{n(n+1)}{2} = \dfrac{1}{2}n^2 + \dfrac{1}{2}n$

Asymptotic Notation and Execution Time

4. For the following pseudo-code, give:

a. Exact value for the number of times "OK" is printed

b. Asymptotic value for the number of times "OK" is printed using Θ notation

for $i := 1$ to n

for $j := i$ to n → For any i, executes $n - i + 1$ times

print("OK")

a. The number of times "OK":

$$T(n) = \sum_{i=1}^{n} (n - i + 1) = \sum_{i=1}^{n} (n + 1) - \sum_{i=1}^{n} i = n(n+1) - \frac{n(n+1)}{2} = \frac{n(n+1)}{2}$$

b. $T(n) = \dfrac{n(n+1)}{2} = \dfrac{1}{2}n^2 + \dfrac{1}{2}n$

We have $1/2\, n^2 + 1/2\, n \leq 1/2\, n^2 + 1/2\, n^2 = n^2$ for $n \geq 0 \Rightarrow T(n) = O(n^2)$

Asymptotic Notation and Execution Time

4. For the following pseudo-code, give:

a. Exact value for the number of times "OK" is printed

b. Asymptotic value for the number of times "OK" is printed using Θ notation

```
for i := 1 to n
    for j := i to n        →  For any i, executes n − i + 1 times
        print("OK")
```

a. The number of times "OK":

$$T(n) = \sum_{i=1}^{n} (n - i + 1) = \sum_{i=1}^{n} (n + 1) - \sum_{i=1}^{n} i = n(n+1) - \frac{n(n+1)}{2} = \frac{n(n+1)}{2}$$

b. $T(n) = \dfrac{n(n+1)}{2} = \dfrac{1}{2}n^2 + \dfrac{1}{2}n$

We have $1/2\, n^2 + 1/2\, n \leq 1/2\, n^2 + 1/2\, n^2 = n^2$ for $n \geq 0 \Rightarrow T(n) = O(n^2)$

We also have $1/2\, n^2 + 1/2\, n \geq 1/2\, n^2$ for $n \geq 0 \Rightarrow T(n) = \Omega(n^2)$

Conclusion: $T(n) = \Theta(n^2)$

Asymptotic Notation and Execution Time

Useful summation formulas:

- $\sum_{i=1}^{n} 1 = n$
- $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$
- $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$
- $\sum_{i=0}^{n} a^i = \frac{a^{n+1}-1}{a-1}$ for $a \neq 1$