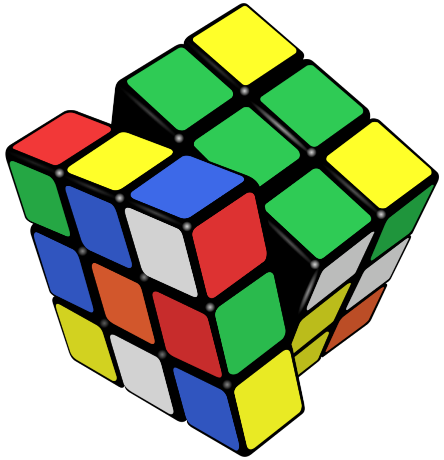
Quiz 1

Section 2 (C0201, LEC2000, LEC2201)



CSC263
Sep 27 – 2017

Question 1

There are three possible procedures A, B, C for running an application. Procedure A takes 3 seconds, procedure B takes 4 seconds and procedure C takes 6 seconds to run. This application to be run chooses algorithm A with the probabilty of 0.40, the algorithm B with the probabilty of 0.50 and C with probabilty of 0.10. What is the average case running time of the application?

A) 2 B) 4.5   C) 3.2 D)3

Players correct: 91%

Question 1

There are three possible procedures A, B, C for running an application. Procedure A takes 3 seconds, procedure B takes 4 seconds and procedure C takes 6 seconds to run. This application to be run chooses algorithm A with the probabilty of 0.40, the algorithm B with the probabilty of 0.50 and C with probabilty of 0.10. What is the average case running time of the application?

A) 3 B)3.8 C) 4 D)6.5

Players correct: 91%

Average-case running time = $\sum x\, p(x) = 3 * 0.4 + 4 * 0.5 + 6 * 0.1 = 3.8$

Question 2

Which of the following is the best upper bound for

$12n + 100\sqrt{n}\log n + \log(n^n)$?

$A)\ O(\sqrt{n}\log n)$

$B)\ O(n^2\log n)$

C) $O(\log n!)$

$D) O(n^n)$

Players correct: 15%

Question 2

Which of the following is the best upper bound for

$12n + 100\sqrt{n}\log n \;\; + \log(n^n)$?

$A)\ O(\sqrt{n}\log n)$

$B)\ O(n^2 \log n)$

C) $O(\log n!)$

$D) O(n^n)$

Players correct: 15%

The fast growth term is $\log(n^n)$ = n log n

The proof on the next slide.

Question 2

A proof for $logn! = \Theta(logn^n)$

It was the very first question on piazza (#30) (also exercise of the textbook)

- $\exists c_0$ such that for large values of n, $logn! \leq c_0 logn^n$
  $logn! = \log 1 + \log 2 + \cdots + logn < nlogn = logn^n \quad (c_0 = 1)$

- $\exists c_0$ such that for large values of n, $c_0 logn! \geq logn^n$
  $logn! = \log(1 \times 2 \ldots \times n) \geq \log\left(\frac{n}{2} \times \cdots \frac{n}{2}\right) = \log(\frac{n}{2}^{(\frac{n}{2})})$
  $= \frac{n}{2}\log(\frac{n}{2}) = \frac{n}{2}(logn - 1)$
  $logn! \geq n/2(logn - 1)$
  We can prove that for $c_0 = 4 \; and \; n > 4$
  $$c_0(\frac{n}{2})(logn - 1) \geq nlogn$$

Question 3

What is the worst case time complexity of insertion sort where position of the element to be inserted is calculated using binary search?

$A)\ O(n \log n)$

$B)\ O(n^2)$

C) $O(\log n)$

$D) O(n)$

Players correct: 25%

Question 3

What is the worst case time complexity of insertion sort where position of the element to be inserted is calculated using binary search?

$A)\ O(n \log n)$

$B)\ O(n^2)$

C) $O(\log n)$

$D) O(n)$

Players correct: 25%

You still need to shift the elements when you find the location of the element in $O(\log n)$

Question 4

What is the output of foo()?

$$
\begin{array}{l}
\textbf{int}\ \text{foo}\ (\textbf{int}\ n) \\
\{ \\
\quad \textbf{int}\ i, j, k = 0 \\
\quad \textbf{for}\ (i = \frac{n}{2};\ i \le n;\ i++) \\
\quad\quad \textbf{for}\ (j = 2;\ j \le n;\ j = j * 2) \\
\quad\quad\quad k = k + n/2; \\
\quad\quad \textbf{return}\ k; \\
\}
\end{array}
$$

Players correct: 9%

$A)\ O(n \log n)$ $\quad B)\ O(n^2 \log n)$ $\quad$ C) $O(n)$ $\quad D)\ O(n^2)$

Question 4

What is the output of foo()?

$$\textbf{int}\ \text{foo}\ (\textbf{int}\ n)$$
$$\{$$
$$\textbf{int}\ i, j, k\ =\ 0$$
$$\textbf{for}\ (i\ = \frac{n}{2};\ i \leq\ n;\ i++)$$
$$\textbf{for}\ (j\ =\ 2;\ j \leq\ n;\ j\ =\ j\ *\ 2)$$
$$k\ =\ k\ +\ n/2;$$
$$\textbf{return}\ k;$$
$$\}$$

- The outer loop: $O(n/2)$
- The inner loop: $O(\log n)$ why?
- $n/2$ is added to k in the inner loop

$\rightarrow k = O(\frac{n}{2} * \log n\ * n/2) = O(n^2 \log n)$

Players correct: 9%

$A)\ O(n \log n)$ $B) O(n^2 \log n)$ C) $O(n)$ $D) O(n^2)$

Question 5

What is the minimum number of comparisons required to find a minimum value stored in a max heap?

$A)\ n$ $B)\ n/2$ C) $O(\log n)$ $D)\ \frac{n}{2} - 1$

Players correct: 36%

Question 5

What is the minimum number of comparisons required to find a minimum value stored in a max heap?

$A)\ n$ $B)\ n/2$ C) $O(\log n)$ $D)\ \frac{n}{2} - 1$

Players correct: 36%

Answer: The minimum element in a max-heap can be anywhere in the leaves of the heap. (why?)

There are up to n/2 elements in the leaves, so finding the needs n/2 comparisons. (why?)

Question 6

A **ternary** max heap is defined similar to a **binary** max heap but each node has at most **3 children** instead of **2 children**. Let $A = \{15, 6, 8, 14, 3, 2\}$ be a ternary max heap with starting index at zero. We insert a new element 11. What is the index of 11 after being inserted to A. (A must remain a ternary max heap.)

$A)1$ $B)2$ C) 3 $D)$ 7

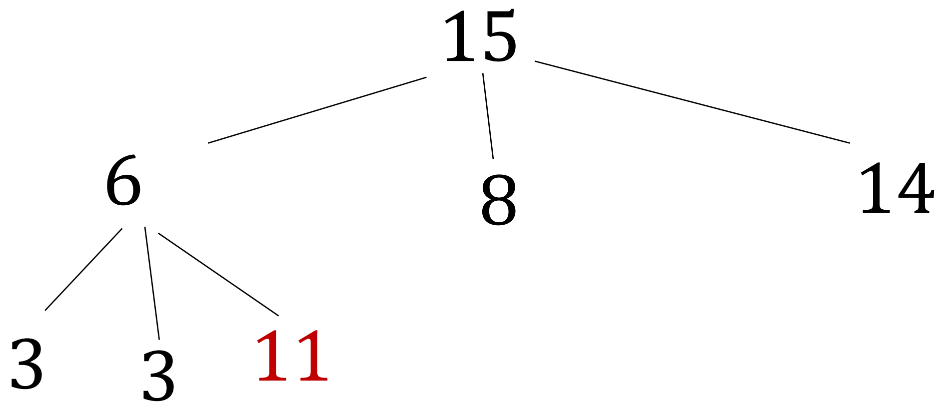
Players correct: 27%

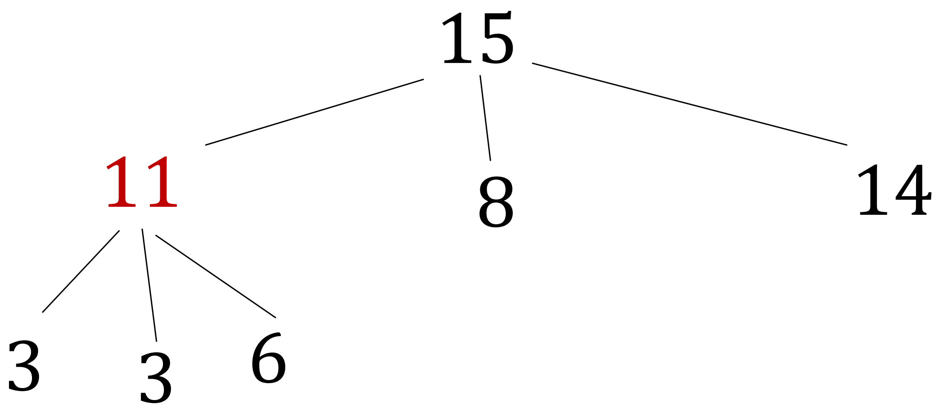
Question 6

A **ternary** max heap is defined similar to a **binary** max heap but each node has at most 3 **children** instead of 2 **children**. Let $A = \{15, 6, 8, 14, 3, 2\}$ be a ternary max heap with starting index at zero. We insert a new element 11. What is the index of 11 after being inserted to A. (A must remain a ternary max heap.)

A)1 $\quad$ B)2 $\quad$ C) 3 $\quad$ D) 7

Players correct: 36%





Question 7

What is time complexity of increasing and decreasing a value of an element in a max heap?

Players correct: 72%

A)O(n), O(n)

B)O(log n), O(n)

C) $O(\log n)$, $O(\log n)$

D) $O(n)O(logn)$

Question 7

What is time complexity of increasing and decreasing a value of an element in a max heap?

Players correct: 72%

A)O(n), O(n)

B)O(log n), O(n)

C) $O(\log n)$, $O(\log n)$

D) $O(n)O(logn)$

Answer:

- Increasing a value need a max-heapify-up operation
- Decreasing a value needs a max-heapify-down operation

Both operations are in $O(\log n)$ since needs to traverse the height of the tree in the worst case.

Question 8

In a heap of height h, what is the minimum number of elements?

(Assume the height of the root is zero).

Players correct: 30%

A) $2^{(h+1)} - 1$

B) 2^h

C) $2^{(h-1)}$

D) $2^{(h+1)}$

Question 8

In a heap of height h, what is the minimum number of elements?

(Assume the height of the root is zero).

Players correct: 30%

A) $2^{(h+1)} - 1$

B) 2^h

C) $2^{(h-1)}$

D) $2^{(h+1)}$

Heap is a complete binary tree, so it is full at all level bottom level which has at least 1 node. The number of nodes at level 0 is $1 = 2^0$, at level 1 is $2 = 2^1$, … , at level $h - 1$ is 2^{h-1} and at level h is 1.
Total number of nodes: $(1 + 2 + 2^2\ \ldots + 2^{h-1}) + 1 = (2^h - 1) + 1 = 2^h$.

Question 9

Let A be an array of size n. What is the best case and worst case time complexity of P?

Procedure $P(A)$
{
$m = 0$
for $i = 1$ to n
if ($A[i]$ is odd)
for $j = i$ to n
$m++$
}

A) Best case : $\Theta(1)$ Wort case: $\Theta(n)$

B) Best case : $\Theta(n)$ Wort case: $\Theta(n^2)$

C) Best case : $\Theta(1)$ Wort case: $\Theta(n^2)$

D) Best case : $\Theta(n^2)$ Wort case: $\Theta(n^2)$

Players correct: 57%

Question 9

Let A be an array of size n. What is the best case and worst case time complexity of P?

Procedure $P(A)$

$\{$

$m = 0$

for $i = 1$ to n

if $(A[i]$ is odd$)$

for $j = i$ to n

$m{+}{+}$

$\}$

A) Best case : $\Theta(1)$ Wort case: $\Theta(n)$

B) Best case : $\Theta(n)$ Wort case: $\Theta(n^2)$

C) Best case : $\Theta(1)$ Wort case: $\Theta(n^2)$

D) Best case : $\Theta(n^2)$ Wort case: $\Theta(n^2)$

Players correct: 30%

Best case happens when all the elements of A are even,
Worst case happens when all the elements of A are odd.