Q Overall Asymptotic Runtime Bound for dup1

Demo

$$R_{\text{best}}(N) = 2$$

$$R_{\text{worst}}(N) = \frac{N^2 + 3N + 2}{2}$$

Give an overall asymptotic runtime bound for R as a combination of **Θ**, **O**, and/or **Ω** notation. Take into account both the best and the worst case runtimes (R_{best} and R_{worst}).

Q1: Give an overall asymptotic runtime bound for R as a combination of **Θ**, **O**, and/or **Ω** notation. Take into account both the best and the worst case runtimes (R_{best} and R_{worst}).

Q Print Party: Attempt 1

Find a simple f(N) such that the runtime R(N) ∈ Θ(f(N)).

A. 1

B. log N

C. N

D. N log N

E. N^2

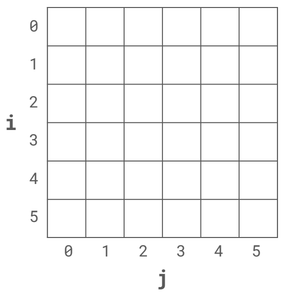
F. Other

```java
void printParty(int N) {
  for (int i = 1; i <= N; i *= 2) {
    for (int j = 0; j < i; j += 1) {
      System.out.println("hello");
    }
  }
}
```

Note that there's only one case.
No separate case analysis!

Q1: Find a simple f(N) such that the runtime R(N) ∈ Θ(f(N)).

?: How do we know that there's only one case to consider?




```java
void printParty(int N) {
  for (int i = 1; i <= N; i *= 2)
    for (int j = 0; j < i; j += 1)
      System.out.println("hello");
}
```

Find a simple f(N) s.t. the runtime R(N) ∈ Θ(f(N)).

N :	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
C(N):																		

Print Party: Attempt 2

Let the cost model C(N) be the number of calls to println for a given N. This is our representative operation for figuring out the runtime.

?: For each N, predict C(N).

Repeat After Me...

Demo

There is no magic shortcut for these problems (except in a few well-behaved cases).

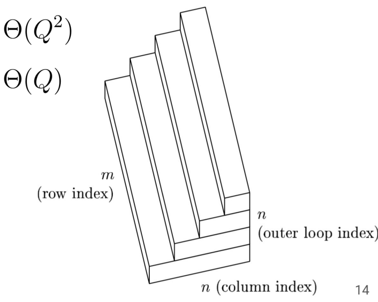
We'll expect you to know these two summations since they're common patterns.

$$1 + 2 + 3 + 4 + \cdots + Q = \frac{Q(Q+1)}{2} \quad \in \Theta(Q^2)$$

$$1 + 2 + 4 + 8 + \cdots + Q = 2Q - 1 \quad \in \Theta(Q)$$

Strategies.

1. Find the exact count of steps.
2. Write out examples.
3. Use a geometric argument–visualizations!



Numerical Linear Algebra (Lloyed N. Trefethen, David Bau, III/SIAM)

Real world programs are often messy and difficult to model.

?: What's different between these two summations?

?: How did we apply these strategies to analyze printParty?

Q Informal Recursion Analysis

```
public static int f3(int n) {
  if (n <= 1)
    return 1;
  return f3(n-1) + f3(n-1);
}
```

Find a simple f(N) such that the runtime R(N) ∈ Θ(f(N)).

Inspect the example and give the order of growth of the runtime as a function of N.

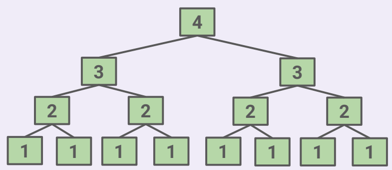
A. 1

B. log N

C. N

D. N^2

E. 2^N



?: What does each node represent in the tree on the right?

Q1: Find a simple f(N) such that the runtime R(N) ∈ Θ(f(N)).

Q Recursion and Exact Counts

```
public static int f3(int n) {
  if (n <= 1)
    return 1;
  return f3(n-1) + f3(n-1);
}
```

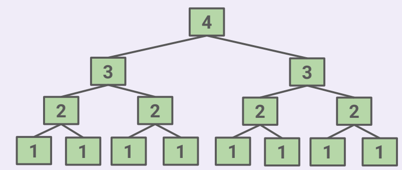
Find a simple f(N) such that the runtime R(N) ∈ Θ(f(N)).

Approach 2: Count number of calls to f3, given by C(N).

$$C(4) = 1 + 2 + 4 + 8$$
$$C(N) = 1 + 2 + 4 + 8 + \cdots + ?$$

Give a simple, exact expression for C(N).

$$C(N) = \boxed{}$$

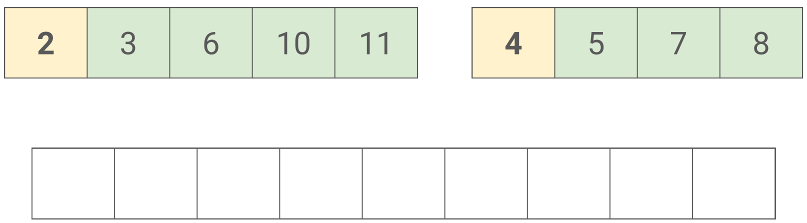


?: What is the exact value of the last term in the sum for C(N)?

Q1: Give a simple, exact expression for C(N).

The Merge Operation

Given **two sorted arrays**, the merge operation combines them into a single sorted array by successively copying the smallest item from the two arrays into a target array.



?: What is a cost model that we can use to evaluate the runtime of the merge operation?

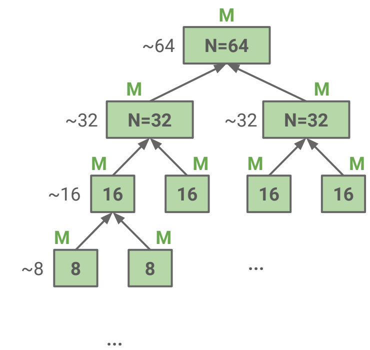
Merge Sort

Merge sort algorithm merges **every layer**.

1. If array is of size 1, return.
2. Merge sort the left half.
3. Merge sort the right half.
4. Merge the two sorted halves.

For N = 64, the total runtime is ~384 AU.

- **Top layer**: ~64 AU
- **Second layer**: 2(~32 AU) = ~64 AU
- **Third layer**: 4(~16 AU) = ~64 AU
- **i^{th} layer**: 2^{i-1}(~64 AU / 2^{i-1}) = ~64 AU



?: How does the call tree for merge sort differ from the example we saw in f3?

?: How do these differences affect our runtime analysis?