1 Recursive Formulae

Suppose that we want to compute \( 2^n \mod M \) for some numbers \( n \geq 0 \) and \( M \geq 2 \). We can use a lot of digits to write down for large \( n \), and we want to avoid that, since the end result is \( n < M \).

Our first attempt avoids multiplication and only uses addition modulo \( M \). We use the fact that \( 2^n = 2^{n-1} \cdot 2 \) (mod \( M \)).

```plaintext
function PowerOfTwo(n, M):
  if n == 0 then return 1
  return (PowerOfTwo(n - 1, M) + PowerOfTwo(n - 1, M)) mod M
What is the runtime of the above algorithm?
  \( \Theta(2^n) \)
  \( \Theta(\log n) \)
```

Now let us replace this algorithm with an iterative one that stores the results:

```plaintext
function PowerOfTwo(n, M):
  A[0] = 1
  for i = 1, . . . , n do
  return A[n]
What is the runtime of the above algorithm?
  \( \Theta(n) \)
  \( \Theta(2^n) \)
  \( \Theta(\log n) \)
```

What if we are allowed to use multiplication? Suppose that \( n \) is a power of two.

```plaintext
function PowerOfTwo(n, M):
  if n == 0 then return 1
  if n is a power of two then return M
  return (PowerOfTwo(n, M) + PowerOfTwo(n, M)) mod M
What is the runtime of the above algorithm?
  \( \Theta(n) \)
  \( \Theta(2^n) \)
  \( \Theta(\log n) \)
```

Remark: A clever algorithm inspired by the above can compute Fibonacci numbers with a desired number \( M \), in time \( \Theta(\log n) \). As a challenge, try to use the following identity involving Fibonacci numbers and matrix multiplication, to come up with this \( \Theta(\log n) \) algorithm.

\[
\text{Fibonacci}(n) = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}^{n-1} \mod M
\]

2 Shortest Paths

Suppose that we have a weighted graph with \( n \) vertices and \( m \) edges and no negative cycles (so shortest paths are well-defined). Suppose for the below questions that our implementation of Dijkstra uses red-black trees (and not Fibonacci heaps).

If \( m = o(n^3) \), and we want to find the shortest path between some \( s \) and \( v \) which algorithm should we use? We prefer algorithms with the smallest worst-case runtime.

```plaintext
Bellman-Ford
Balloch-Wallshelt
Two or more of the above algorithms are correct and have the smallest worst-case runtime.
```

What if all the edges have nonnegative weight?

```plaintext
Dijkstra
Bellman-Ford
Floyd-Wallshelt
Two or more of the above algorithms are correct and have the smallest worst-case runtime.
```

Suppose that we have a graph with \( m = \Omega(n^2) \) edges that all have nonnegative weights. Which algorithm should we use to find the shortest path between all pairs of vertices?

```plaintext
Dijkstra
Bellman-Ford
Floyd-Wallshelt
Two or more of the above algorithms are correct and have the smallest worst-case runtime.
```

What if we are allowed to use matrix multiplication? Suppose that \( n \) is a power of two.

```plaintext
function ShortestPaths(M):
  A[0] = 1
  for i = 1, . . . , n do
  return A[n]
What is the runtime of the above algorithm?
  \( \Theta(n) \)
  \( \Theta(2^n) \)
  \( \Theta(\log n) \)
```

Remark: A clever algorithm inspired by the above can compute Fibonacci numbers modulo a desired number \( M \), in time \( \Theta(\log n) \). As a challenge, try to use the following identity involving Fibonacci numbers and matrix multiplication, to come up with this \( \Theta(\log n) \) algorithm.

\[
\text{Fibonacci}(n) = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}^{n-1} \mod M
\]

2 Shortest Paths

Suppose that we have a weighted graph with \( n \) vertices and \( m \) edges and no negative cycles (so shortest paths are well-defined). Suppose for the below questions that our implementation of Dijkstra uses red-black trees (and not Fibonacci heaps).

If \( m = o(n^3) \), and we want to find the shortest path between some \( s \) and \( v \) which algorithm should we use? We prefer algorithms with the smallest worst-case runtime.

```plaintext
Dijkstra
Bellman-Ford
Floyd-Wallshelt
Two or more of the above algorithms are correct and have the smallest worst-case runtime.
```

What if all the edges have nonnegative weight?

```plaintext
Dijkstra
Bellman-Ford
Floyd-Wallshelt
Two or more of the above algorithms are correct and have the smallest worst-case runtime.
```

Suppose that we have a graph with \( m = \Omega(n^2) \) edges that all have nonnegative weights. Which algorithm should we use to find the shortest path between all pairs of vertices?

```plaintext
Dijkstra
Bellman-Ford
Floyd-Wallshelt
Two or more of the above algorithms are correct and have the smallest worst-case runtime.
```