1 Recursive Formulae

Suppose that we want to compute $2^n \mod M$ for some numbers $n \geq 0$ and $M \geq 2$. $2^n$ can require a lot of digits to write down for large $n$, and we want to avoid that, since the end result is $< M$. Our first attempt avoids multiplication and only uses addition modulo $M$. We use the fact that $2^n = 2^{n-1} + 2^{n-1}$ (mod $M$).

```python
def PowerOfTwo(n, M):
    if n == 0:
        return 1
    return (PowerOfTwo(n-1, M) + PowerOfTwo(n-1, M)) % M
```

What is the runtime of the above algorithm?
- $O(n)$
- $O(2^n)$
- $O(\log n)$

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

$A = \text{array indexed with } 0, \ldots, n$
$A[0] = 1$
for $i = 1, \ldots, n$ do
    $A[i] = (A[i-1] + A[i-1]) \mod M$
return $A[n]$

What is the runtime of the above algorithm?
- $O(n)$
- $O(2^n)$
- $O(\log n)$

If $n$ is a power of two? We can run the following slightly modified algorithm:

$B = \text{array indexed with } 0, \ldots, \log n$
$B[0] = 2$
for $i = 1, \ldots, \log n$ do
    $B[i] = (B[i-1] + B[i-1]) \mod M$
Let the binary representation of $n$ be $(x_{\log n-1}, \ldots, x_0)$. $B = [1]$
for $i = 0, \ldots, \log n$ do
    if $x_i = 1$ then
        $A = (B[B[i]]) \mod M$
return $A$

What is the runtime of this algorithm?
- $O(n)$
- $O(2^n)$
- $O(\log n)$

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 $n = n^\alpha n^\beta$, and we want to find the shortest path between some $u$ and $v$ algorithms which should we use? We prefer algorithms with the smallest worst-case runtime.
- $\alpha$ Dijkstra
- Bellman-Ford
- Floyd-Warshall
Two or more of the above algorithms are correct and have the smallest worst-case runtime.

What if all the edges have nonnegative weight?
- $\alpha$ Dijkstra
- Bellman-Ford
- Floyd-Warshall
Two or more of the above algorithms are correct and have the smallest worst-case runtime.

Correct

What is the runtime of the above algorithm?
- $O(n)$
- $O(2^n)$
- $O(\log n)$

Two or more of the above algorithms are correct and have the smallest worst-case runtime.

Correct

Suppose that we want to compute Fibonacci(0) modulo a desired number $M$, in time $O(n \log n)$. As a challenge, try to use the following identity involving Fibonacci numbers and matrix multiplication, to come up with this $O(\log n)$ algorithm:

$Fibonacci[0] = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{n}$

Critical

Suppose that we have a graph with $n \geq n^\alpha n^\beta$ edges that all have nonnegative weights. Which algorithm should we use to find the shortest path between all pairs of vertices?
- $\alpha$ Dijkstra
- $\beta$ Dijkstra
- $\beta$ Bellman-Ford
- $\alpha$ Bellman-Ford
- $\alpha$ Floyd-Warshall
Two or more of the above algorithms are correct and have the smallest worst-case runtime.

Correct