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 $\leq M$.

Our first attempt avoids multiplication and only uses addition modulo $M$. We use the fact that $2^n \equiv 2^{n \mod \log_2 M} \pmod M$.

function PowerOfTwo(n, M):
  if $n = 0$ then
    return 1
  return $(\text{PowerOfTwo}(n-1, M) + \text{PowerOfTwo}(n-1, M)) \mod M$

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

Correct

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 $i = 1, \ldots, \log n$ do
  $A[i] = (A[i-1] + A[i-1]) \mod M$
return $A[\log n]$

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

Correct

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

$$A = \text{array indexed with } 0, \ldots, \log n$$
$$A[0] = 2$$
for $i = 1, \ldots, \log n$ do
  $A[i] = (A[i-1] + A[i-1]) \mod M$
return $A[\log n]$

What is the value of $A[i]$ in the above algorithm?
- $2^i \mod M$
- $2^i \mod M$
- $2^i \mod M$

Correct

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

Correct

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

$$\begin{bmatrix} \text{Fibonacci}(n) \\ \text{Fibonacci}(n-1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \text{Fibonacci}(n-1) \\ \text{Fibonacci}(n-2) \end{bmatrix}$$

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).

Suppose that we have a weighted graph with $n$ vertices and $m$ edges that all have nonnegative weights. Which algorithm should we use to find the shortest path between all pairs of vertices?
- Dijkstra
- Bellman-Ford
- Floyd-Warshall

Correct

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.
- Dijkstra
- Bellman-Ford
- Floyd-Warshall

Correct

What if all the edges have nonnegative weight?
- Dijkstra
- Bellman-Ford
- Floyd-Warshall

Correct

Suppose that we have a graph with $n \geq o(n^3)$ edges that all have nonnegative weights. Which algorithm should we use to find the shortest path between all pairs of vertices?
- $O^* n$ runs of Dijkstra
- $O^* n$ runs of Bellman-Ford
- $O^* n$ runs of Floyd-Warshall

Correct

Suppose that we have a graph with $n \geq o(n^3)$ edges that all have nonnegative weights. Which algorithm should we use to find the shortest path between all pairs of vertices?
- $O^* n$ runs of Dijkstra
- $O^* n$ runs of Bellman-Ford
- $O^* n$ runs of Floyd-Warshall

Correct