Fast Matrix Exponentiation

With Applications

Nurseiit Abdimomyn

UNIST

October 26, 2018

Outline

Matrix Multiplication

Consider two matrices:

Matrix Multiplication

Consider two matrices:

- Matrix A is $n * k$ dimensional.

Matrix Multiplication

Consider two matrices:

- Matrix A is $n * k$ dimensional.
- Matrix B is $k * m$ dimensional.

Matrix Multiplication

Consider two matrices:

- Matrix A is $n * k$ dimensional.
- Matrix B is $k * m$ dimensional.

Notice that A's columns and B's rows number are identical!

Matrix Multiplication

Consider two matrices:

- Matrix A is $n * k$ dimensional.
- Matrix B is $k * m$ dimensional.

Notice that A's columns and B's rows number are identical!

Then we define matrix $C = A * B$ as:

$$\begin{bmatrix} c_{11} & c_{12} & \dots & c_{1m} \\ c_{21} & c_{22} & \dots & c_{2m} \\ \dots & \dots & \dots & \dots \\ c_{n1} & c_{n2} & \dots & c_{nm} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1k} \\ a_{21} & a_{22} & \dots & a_{2k} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nk} \end{bmatrix} * \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1m} \\ b_{21} & b_{22} & \dots & b_{2m} \\ \dots & \dots & \dots & \dots \\ b_{k1} & b_{k2} & \dots & b_{km} \end{bmatrix}$$

Matrix Multiplication

Consider two matrices:

- Matrix A is $n * k$ dimensional.
- Matrix B is $k * m$ dimensional.

Notice that A's columns and B's rows number are identical!

Then we define matrix $C = A * B$ as:

$$\begin{bmatrix} c_{11} & c_{12} & \dots & c_{1m} \\ c_{21} & c_{22} & \dots & c_{2m} \\ \dots & \dots & \dots & \dots \\ c_{n1} & c_{n2} & \dots & c_{nm} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1k} \\ a_{21} & a_{22} & \dots & a_{2k} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nk} \end{bmatrix} * \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1m} \\ b_{21} & b_{22} & \dots & b_{2m} \\ \dots & \dots & \dots & \dots \\ b_{k1} & b_{k2} & \dots & b_{km} \end{bmatrix}$$

Thus, C is an $n * m$ dimensional matrix.

Which is calculated as: $c_{ij} = \sum_{r=1}^{k} a_{ir} * b_{rj}$

Couple of things to notice

- Matrix C has $n * m$ elements and each element is computed in k steps with given formula.

Couple of things to notice

- Matrix C has $n * m$ elements and each element is computed in k steps with given formula.
  Thus, we can obtain C in $O(n * m * k)$, given A and B.

Couple of things to notice

- Matrix C has $n * m$ elements and each element is computed in k steps with given formula.
  Thus, we can obtain C in $O(n * m * k)$, given A and B.
- If $n = m = k$ (i.e. both A and B have n rows and n columns), then C has n rows and n columns, and can be computed in $O(n^3)$.

Some useful properties of matrix multiplication

- It is *not commutative*: $A * B \neq B * A$ in general case.

Some useful properties of matrix multiplication

- It is *not commutative*: $A * B \neq B * A$ in general case.
- It is *associative*:
  $A * B * C = (A * B) * C = A * (B * C)$ in case $A * B * C$ exists;

Some useful properties of matrix multiplication

- It is *not commutative*: $A * B \neq B * A$ in general case.
- It is *associative*:
  $A * B * C = (A * B) * C = A * (B * C)$ in case $A * B * C$ exists;
- If you have a matrix with n rows and n columns,
  then multiplying it by I_n gives the same matrix.
  i. e. $I_n * A = A * I_n = A$.

Some useful properties of matrix multiplication

- It is *not commutative*: $A * B \neq B * A$ in general case.
- It is *associative*:
  $A * B * C = (A * B) * C = A * (B * C)$ in case $A * B * C$ exists;
- If you have a matrix with n rows and n columns,
  then multiplying it by I_n gives the same matrix.
  i. e. $I_n * A = A * I_n = A$.
  Where I_n is a matrix with n rows and n columns of such form:

$$I_n = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 \end{bmatrix}$$

Outline

Matrix exponentiation

Suppose we need to find some $n * n$ dimensional square matrix A to the power p or A^p

Matrix exponentiation

Suppose we need to find some $n * n$ dimensional square matrix A to the power p or A^p
We can do so via:

```
function matpow_naive(A, p):
  result = I_n
  for i = 1..p:
    result = result * A
  return result
```

Which will run in $O(n^3 * p)$

Fast Matrix exponentiation

Can we do it any faster?

Fast Matrix exponentiation

Can we do it any faster?

Yes, we can apply the *BinPower* algorithm here:

```
function matBinPow(A, p):
    result = I_n
    while p > 0:
        if p % 2 == 1:
            result = result * A
        A = A * A
        p = p / 2
    return result
```

Which will run in $O(n^3 * \log p)$

Outline

Finding Nth Fibonacci number

Fibonacci numbers, F_n are defined as:

- $F_0 = F_1 = 1$
- $F_i = F_{i-1} + F_{i-2}$ for $i > 1$.

Finding Nth Fibonacci number

Fibonacci numbers, F_n are defined as:

- $F_0 = F_1 = 1$
- $F_i = F_{i-1} + F_{i-2}$ for $i > 1$.

We want to calculate $F_n \ mod \ M$, where $n < 10^{18}$ and $M = 10^9 + 7$.

Finding Nth Fibonacci number

Suppose we have a vector (matrix with one row and several columns) of (F_{i-2}, F_{i-1}) and we want to multiply it by some matrix, so that we get (F_{i-1}, F_i) as a result.

Finding Nth Fibonacci number

Suppose we have a vector (matrix with one row and several columns) of (F_{i-2}, F_{i-1}) and we want to multiply it by some matrix, so that we get (F_{i-1}, F_i) as a result.
Let's call this matrix M:
$(F_{i-2}, F_{i-1}) * M = (F_{i-1}, F_i)$

Two questions we should answer arise immediately:

1. What are the dimensions of M?
2. What are the exact values in M?

Finding Nth Fibonacci number

We can answer them using the definition of matrix multiplication:

Finding Nth Fibonacci number

We can answer them using the definition of matrix multiplication:

- The *size*.
  We multiply (F_{i-2}, F_{i-1}), which has 1 row and 2 columns, by M.
  The result is (F_{i-1}, F_i), which has 1 row and 2 columns.

Finding Nth Fibonacci number

We can answer them using the definition of matrix multiplication:

- The *size*.
  We multiply (F_{i-2}, F_{i-1}), which has 1 row and 2 columns, by M.
  The result is (F_{i-1}, F_i), which has 1 row and 2 columns.

  By definition, if we multiply a matrix with N rows and K columns by a matrix with K rows and L columns, we get a matrix with N rows and L columns.

  Therefore, matrix M has $K = 2$ rows and $L = 2$ columns.

Finding Nth Fibonacci number

We can answer them using the definition of matrix multiplication:

- The *values*.

Finding Nth Fibonacci number

We can answer them using the definition of matrix multiplication:

- The *values*.
  We now know that M has 2 rows and 2 columns, 4 values overall.
  Lets denote them by letters, as we usually do with unknown variables:

$$M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Finding Nth Fibonacci number

We can answer them using the definition of matrix multiplication:

- The *values*.
  We now know that M has 2 rows and 2 columns, 4 values overall.
  Lets denote them by letters, as we usually do with unknown variables:

$$M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

  We want to find a, b, c and d.

Finding Nth Fibonacci number

We have,

$$M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Finding Nth Fibonacci number

We have,

$$M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

We want to find a, b, c and d.
Let's see what we get if we multiply (F_{i-2}, F_{i-1}) by M by definition:

Finding Nth Fibonacci number

We have,

$$M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

We want to find a, b, c and d.
Let's see what we get if we multiply (F_{i-2}, F_{i-1}) by M by definition:

$$(F_{i-2}, F_{i-1}) * \begin{bmatrix} a & b \\ c & d \end{bmatrix} = (a * F_{i-2} + c * F_{i-1}, b * F_{i-2} + d * F_{i-1})$$

Finding Nth Fibonacci number

We have,

$$M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

We want to find a, b, c and d.
Let's see what we get if we multiply (F_{i-2}, F_{i-1}) by M by definition:

$$(F_{i-2}, F_{i-1}) * \begin{bmatrix} a & b \\ c & d \end{bmatrix} = (a * F_{i-2} + c * F_{i-1}, b * F_{i-2} + d * F_{i-1})$$

Moreover, we know that the result of this multiplication must be (F_{i-1}, F_i):
$(a * F_{i-2} + c * F_{i-1}, b * F_{i-2} + d * F_{i-1}) = (F_{i-1}, F_i)$

Finding Nth Fibonacci number

Now we can write the system of equations:

- $a * F_{i-2} + c * F_{i-1} = F_{i-1}$
- $b * F_{i-2} + d * F_{i-1} = F_i$

Finding Nth Fibonacci number

Now we can write the system of equations:

- $a * F_{i-2} + c * F_{i-1} = F_{i-1}$
- $b * F_{i-2} + d * F_{i-1} = F_i$

The easiest solution is to set:
$a = 0, b = 1, c = 1, d = 1$

Finding Nth Fibonacci number

Now we can write the system of equations:

- $a * F_{i-2} + c * F_{i-1} = F_{i-1}$
- $b * F_{i-2} + d * F_{i-1} = F_i$

The easiest solution is to set:
$a = 0, b = 1, c = 1, d = 1$
Thus, we obtain:

$$(F_{i-2}, F_{i-1}) * \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} = (F_{i-1}, F_i)$$

Finding Nth Fibonacci number

You may wonder why on Earth would we need to overcomplicate the problem, introduce matrices or else?
Hang on, you'll see in a moment.

Finding Nth Fibonacci number

You may wonder why on Earth would we need to overcomplicate the problem, introduce matrices or else?
Hang on, you'll see in a moment.
Initially, we have F_0 and F_1.
$(F_0, F_1) = (1, 1)$

Finding Nth Fibonacci number

You may wonder why on Earth would we need to overcomplicate the problem, introduce matrices or else?
Hang on, you'll see in a moment.
Initially, we have F_0 and F_1.
$(F_0, F_1) = (1, 1)$
By multiplying this vector with the matrix M we get:
$(F_1, F_2) = (1, 2)$ or $(1, 1) * M = (1, 2)$.

Finding Nth Fibonacci number

You may wonder why on Earth would we need to overcomplicate the problem, introduce matrices or else?
Hang on, you'll see in a moment.
Initially, we have F_0 and F_1.
$(F_0, F_1) = (1, 1)$
By multiplying this vector with the matrix M we get:
$(F_1, F_2) = (1, 2)$ or $(1, 1) * M = (1, 2)$.
If we multiply $(1, 2)$ by M:
$(1, 2) * M = (2, 3)$

Finding Nth Fibonacci number

You may wonder why on Earth would we need to overcomplicate the problem, introduce matrices or else?
Hang on, you'll see in a moment.
Initially, we have F_0 and F_1.
$(F_0, F_1) = (1, 1)$
By multiplying this vector with the matrix M we get:
$(F_1, F_2) = (1, 2)$ or $(1, 1) * M = (1, 2)$.
If we multiply $(1, 2)$ by M:
$(1, 2) * M = (2, 3)$
But we could get the same result by multiplying $(1, 1)$ by M^2.
$(1, 1) * M^2 = (2, 3)$.

Finding Nth Fibonacci number

You may wonder why on Earth would we need to overcomplicate the problem, introduce matrices or else?
Hang on, you'll see in a moment.
Initially, we have F_0 and F_1.
$(F_0, F_1) = (1, 1)$
By multiplying this vector with the matrix M we get:
$(F_1, F_2) = (1, 2)$ or $(1, 1) * M = (1, 2)$.
If we multiply $(1, 2)$ by M:
$(1, 2) * M = (2, 3)$
But we could get the same result by multiplying $(1, 1)$ by M^2.
$(1, 1) * M^2 = (2, 3)$.
Now, can you see the pattern?

Finding Nth Fibonacci number

In general, we get:
$(1, 1) * M^k = (F_k, F_{k+1})$

Finding Nth Fibonacci number

In general, we get:
$(1, 1) * M^k = (F_k, F_{k+1})$
Computing M^k takes $O(2^3 * log(k))$ time.
Thus, we can now find N-th Fibonacci number in $O(log(N))$ time.

Finding Nth Fibonacci number

In general, we get:
$(1, 1) * M^k = (F_k, F_{k+1})$
Computing M^k takes $O(2^3 * log(k))$ time.
Thus, we can now find N-th Fibonacci number in $O(log(N))$ time.

You can find the reference code at *https://devnur.me/campunist*

General Solution for Recurrence Relations

We can generalize the Fibonacci problem's solution to solve linear recurrent sequences.

General Solution for Recurrence Relations

We can generalize the Fibonacci problem's solution to solve linear recurrent sequences.
Let's take a look at more general problem than before. Suppose sequence A_i satisfies the following:

- $A_0 = a_0, A_1 = a_1, ..., A_{k-1} = a_{k-1}$ for given $a_0, a_1, ..., a_{k-1}$.

General Solution for Recurrence Relations

We can generalize the Fibonacci problem's solution to solve linear recurrent sequences.
Let's take a look at more general problem than before. Suppose sequence A_i satisfies the following:

- $A_0 = a_0, A_1 = a_1, ..., A_{k-1} = a_{k-1}$ for given $a_0, a_1, ..., a_{k-1}$.
- $A_i = c_1 * A_{i-1} + c_2 * A_{i-2} + ... + c_k * A_{i-k}$ for given $c_1, c_2, ..., c_k$.

General Solution for Recurrence Relations

We can generalize the Fibonacci problem's solution to solve linear recurrent sequences.
Let's take a look at more general problem than before. Suppose sequence A_i satisfies the following:

- $A_0 = a_0, A_1 = a_1, ..., A_{k-1} = a_{k-1}$ for given $a_0, a_1, ..., a_{k-1}$.
- $A_i = c_1 * A_{i-1} + c_2 * A_{i-2} + ... + c_k * A_{i-k}$ for given $c_1, c_2, ..., c_k$.

We need to find $A_N \ mod \ 10^9 + 7$, for $N \leq 10^{18}$ and $k \leq 50$.

General Solution for Recurrence Relations

We can generalize the Fibonacci problem's solution to solve linear recurrent sequences.
Let's take a look at more general problem than before. Suppose sequence A_i satisfies the following:

- $A_0 = a_0, A_1 = a_1, ..., A_{k-1} = a_{k-1}$ for given $a_0, a_1, ..., a_{k-1}$.
- $A_i = c_1 * A_{i-1} + c_2 * A_{i-2} + ... + c_k * A_{i-k}$ for given $c_1, c_2, ..., c_k$.

We need to find $A_N \ mod \ 10^9 + 7$, for $N \leq 10^{18}$ and $k \leq 50$.

Side note: sequence A_i is called **recurrent**, because computing A_i requires computing A_j for some $j < i$. Also, A_i is called **linear**, because it depends linearly on A_j for some $j < i$.

Thank you!