(/add_ques

Egyptian Fraction | Greedy Algorithm

In early Egypt, people only used unit fractions (fraction of the form $\frac{1}{n}$) to represent the fractional numbers instead of decimals, and fractions other than the unit fraction (like $\frac{2}{3}$) as we use today.

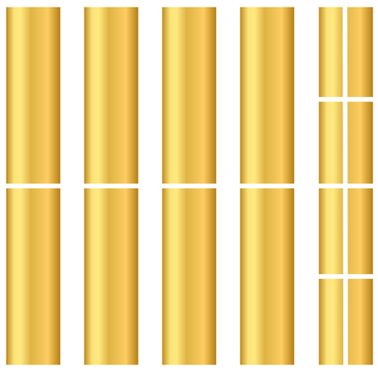
So, the Egyptians used a fraction of the form $\frac{1}{n}$, where the numerator is always 1 and denominator is a positive number and all other fractions were represented as the summation of the unit fractions. For example, $\frac{6}{7} = \frac{1}{2} + \frac{1}{3} + \frac{1}{42}$.

As per the algorithm is concerned, our task is to represent any fraction given to us in the form of a unit fraction but let's first take a look at an example of using Egyptian fraction.

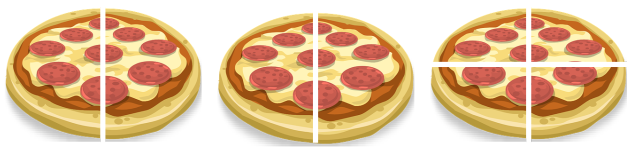
Also, take a note that in this chapter, our discussion will be for the fractions less than 1 i.e., $\frac{a}{b} < 1$ or $b > a$.

Example of Using Egyptian Fraction

A common example always given for the use of Egyptian fraction is something dividing equally among few people. For example, dividing 3 pizzas among 4 people or dividing 5 bars of gold among 8 people, etc.



Let's take the case of dividing 3 pizzas among 4 people.



To solve the problem, we can divide the first two pizzas into half and give one half to each person and then the remaining one pizza can be divided into 4 equal parts and then a quarter can be given again to each person.

We have basically represented $\frac{3}{4}$ into Egyptian fraction i.e., $\frac{3}{4} = \frac{1}{2} + \frac{1}{4}$.

Take note that there is not a unique way to represent a fraction into Egyptian fraction. For example, $\frac{5}{7}$ can be represented as $\frac{1}{2} + \frac{1}{5} + \frac{1}{70}$ as well as $\frac{1}{2} + \frac{1}{6} + \frac{1}{21}$ and there are other ways also.

Let's look at the algorithm which we can use to generate the Egyptian fraction of any fraction.

Greedy Algorithm for Egyptian Fraction

The greedy algorithm was developed by **Fibonacci** and states to extract the largest unit fraction first. Now for a fraction, $\frac{m}{n}$, the largest unit fraction we can extract is $\frac{1}{\lceil \frac{n}{m} \rceil}$. For example, for the fraction $\frac{2}{3}$, the largest unit fraction which we can extract is $\frac{1}{\lceil \frac{3}{2} \rceil} = \frac{1}{\lceil 1.5 \rceil} = \frac{1}{2}$. So, after extracting $\frac{1}{2}$ from $\frac{2}{3}$, we are left with $\frac{2}{3} - \frac{1}{2} = \frac{1}{6}$.

Let's take one more example of $\frac{4}{5}$. $\frac{1}{\lceil \frac{n}{m} \rceil} = \frac{1}{\lceil \frac{5}{4} \rceil} = \frac{1}{2}$.

Thus, the first unit fraction is $\frac{1}{2}$ and now we are left with $\frac{4}{5} - \frac{1}{2} = \frac{3}{10}$

Similarly, we can extract $\frac{1}{4}$ from $\frac{3}{10}$ which will leave us $\frac{1}{20}$. Thus, $\frac{1}{2}$, $\frac{1}{4}$ and $\frac{1}{20}$ are the required unit fractions.

We can repeat this process of extracting the largest unit fraction until the remaining fraction is also a unit fraction.

We have made a statement that the largest unit fraction we can extract from $\frac{m}{n}$ is $\frac{1}{\lceil \frac{n}{m} \rceil}$. Let's prove the same.

If you are not interested in the proof, you can directly skip to the algorithm.

Prove of $\lceil \frac{n}{m} \rceil$ Give Maximum Unit Fraction for $\frac{m}{n}$

We are stating that $\frac{1}{\lceil \frac{n}{m} \rceil}$ is the largest unit fraction which can be extracted from the fraction $\frac{m}{n}$. If we decrease the value of $\lceil \frac{n}{m} \rceil$ by 1 (i.e., $\frac{1}{\lceil \frac{n}{m} \rceil - 1}$), it will be the next larger unit fraction (we are decreasing the denominator and hence, increasing the value of the fraction).

Now, we will show that $\frac{1}{\lceil \frac{n}{m} \rceil - 1} > \frac{m}{n}$ and thus $\frac{1}{\lceil \frac{n}{m} \rceil}$ was the largest unit fraction we could have extracted.

We have to show,

$$\frac{1}{\lceil \frac{n}{m} \rceil - 1} > \frac{m}{n}$$

Cross multiplying,

L.H.S. = n

and R.H.S. = $m * \left(\lceil \frac{n}{m} \rceil - 1\right)$

$= \left(m * \lceil \frac{n}{m} \rceil\right) - (m)$

We can write $\lceil \frac{n}{m} \rceil$ as $\frac{n}{m} + \epsilon$, where $\epsilon \in (0, 1)$. For example, $\lceil \frac{3}{2} \rceil$ can be written as $\frac{3}{2} + 0.5$.

So, we can write the R.H.S. as $\left(m * \left(\frac{n}{m} + \epsilon\right)\right) - (m)$

$= n + m\epsilon - m$

Since ϵ is less than 1, so $m\epsilon < m$. Thus, $m\epsilon - m$ is negative.

Since $m\epsilon - m < 0$, it means we are subtracting some value from n in the equation $n + m\epsilon - m$. So, $n + m\epsilon - m < n$.

Thus, L.H.S. > R.H.S. and hence $\frac{1}{\lceil \frac{n}{m} \rceil - 1} > \frac{m}{n}$.

So, we have proved that our strategy of finding the largest unit fraction from a fraction is correct but we are left with one more task. We need to also show that extracting the largest unit fraction from a fraction will not give us any infinite series and will always terminate i.e., we will always get to a point where the remaining fraction after the subtraction will be a unit fraction.

Proof of Greedy Choice in Egyptian Fraction Always Terminate

We are going to use the method of induction to prove this. We have a fraction $\frac{m}{n}$ and our base case is when $m = 1$.

When m is 1, then we already have the unit fraction. Now, we will assume that it is true for $m \in \{1, 2, \ldots, k\}$, where $k \geq 1$ (it means integers between 1 to k) and will then prove that it is also true for $k + 1$. This is the method of induction i.e., we first check if it is true for 1 or not and then assume it is true for k and then prove that it is also true for k+1.

So, we have assumed that $\frac{k}{n}$ can be expressed as the sum of unit fractions and it will terminate. Now, let's test for $\frac{k+1}{n}$.

The largest fraction we can extract from $\frac{k+1}{n}$ is $\frac{1}{\lceil \frac{n}{k+1} \rceil}$ and thus we will be left with $\frac{k+1}{n} - \frac{1}{\lceil \frac{n}{k+1} \rceil}$

$$\frac{k+1}{n} - \frac{1}{\lceil \frac{n}{k+1} \rceil} = \frac{\lceil \frac{n}{k+1} \rceil (k+1) - n}{\lceil \frac{n}{k+1} \rceil n}$$

Considering numerator,

$$\left\lceil \frac{n}{k+1} \right\rceil (k+1) - n = \left(\left\lfloor \frac{n-1}{k+1} \right\rfloor + 1 \right) (k+1) - n$$

(As we can write $\lceil 1.5 \rceil = \lfloor 1.5 \rfloor + 1 = 2$)

Also,

$$\left(\left\lfloor \frac{n-1}{k+1} \right\rfloor + 1 \right) (k+1) - n \leq \left(\frac{n-1}{k+1} + 1 \right) (k+1) - n = (n-1) + (k+1) - n = k$$

(As $\lfloor x \rfloor \leq x$)

Since,

$$\left\lceil \frac{n}{k+1} \right\rceil (k+1) - n = \left(\left\lfloor \frac{n-1}{k+1} \right\rfloor + 1 \right) (k+1) - n$$

$$=> \left\lceil \frac{n}{k+1} \right\rceil (k+1) - n \leq k$$

So, we have proved that the numerator left after taking the largest unit fraction from $\frac{k+1}{n}$ is between 1 to k and we have also assumed that we can get unit fractions for the numerator $\leq k$, so it will have a unit fraction and will terminate.

Thus, we have proved this by the method of induction. If you are not familiar with the method of induction, you can check out Mathematical induction - Wikipedia (https://en.wikipedia.org/wiki/Mathematical_induction).

Let's look at the coding implementation of the algorithm.

Code for Egyptian Fraction

Our function is going to take the fraction i.e., the numerator and the denominator for the input - `GREEDY-EGYPTIAN-FRACTION(num, den)`.

We will make an array to store the denominators of the unit functions. If the 'num' is already 1, then it is a unit fraction, so we will just store its denominator in the array.

```
unit_den_array = []
GREEDY-EGYPTIAN-FRACTION(num, den)
  if num == 1
    unit_den_array.append(den)
```

Otherwise, we will follow the greedy strategy. So, we will first calculate $\left\lceil \frac{den}{num} \right\rceil$ i.e., `unit_den = ceil(den/num)` and then store this value in the array.

```
if num == 1
   ...
else
  unit_den = ceil(den/num)
  unit_den_array.append(unit_den)
```

Now, we will again extract the largest functions from $\frac{m}{n} - \frac{1}{\lceil \frac{n}{m} \rceil}$ or $\frac{num}{den} - \frac{1}{unit\_den}$ i.e., `GREEDY-EGYPTIAN-FRACTION((num*unit_den) - den, den*unit_den)`. But the fraction formed by the passed numerator ((num*unit_den) - den) and denominator (den*unit_den) must be in their lowest term i.e., we should divide the numerator and the denominator by their GCD (Greatest Common Divisor) -

```
gcd = GCD((num*unit_den) - den, (den*unit_den))
GREEDY-EGYPTIAN-FRACTION(((num*unit_den) - den)/gcd, (den*unit_den)/gcd)
```

```
unit_den_array = []
GREEDY-EGYPTIAN-FRACTION(num, den)
  if num == 1
     unit_den_array.append(den)
  else
     unit_den = ceil(den/num)
     unit_den_array.append(unit_den)
     gcd = GCD((num*unit_den) - den, (den*unit_den))
     GREEDY-EGYPTIAN-FRACTION(((num*unit_den) - den)/gcd, (den*unit_den)/gcd)
```

C **Python** **Java**

```
#include <stdio.h>
#include <math.h>

int unit_den_array[10];
int iter = 0;

int gcd(int a, int b) {
  int c = a%b;
  while(c > 0) {
    a = b;
    b = c;
    c = a % b;
  }
  return b;
}

void unit_den_init() {
  int i;
  for(i=0; i<10; i++) {
    unit_den_array[i] = 0;
  }
}

void greedy_egyptian_fraction(int num, int den) {
  if(num == 1) {
    //appending array unit_den_array
    iter++; // storing in unit_den_array from index 1 not 0
    unit_den_array[iter] = den;
  }
  else {
    int unit_den = ceil(den/(num*1.0));
    iter++;
    unit_den_array[iter] = unit_den;
    int gcd_of_numbers = gcd((num*unit_den) - den, den*unit_den);
    greedy_egyptian_fraction(((num*unit_den) - den)/gcd_of_numbers, (den*unit_den)/gcd_of_numbers);
  }
}

int main() {
  unit_den_init();
  greedy_egyptian_fraction(4, 5);
  int i;
  for(i=1; i<=iter; i++) {
    printf("%d\n",unit_den_array[i]);
  }
  return 0;
}
```

Till now, you must have a strong grip over greedy algorithms. So, let's finish this section of greedy algorithm by studying one last problem based on greedy algorithm.

“ I just wouldn’t give in, no way. ”

- S. Honda

PREV (/course/algorithms-activity-selection/) (/course/algorithms-huffman-codes/) NEXT

Download Our App.