Greedy Algorithms: Egyptian Fractions

Algorithms for Computing Egyptian Fraction Decomposition

Note:The algorithm takes as input two numbers M and N, representing the fraction $\frac{M}{N}$. The algorithm assumes that $M < N$. All division operations in the algorithms below are integer divisions.

Recursive Algorithm

```
ALGORITHM EgyptFractionRecursive(M,N)
begin

  if M= 1 then    //base case:  fraction is 1/N
    EF[0..1];
    EF[0]← 1;    //size of Egyptian Fraction decomposition is 1
    EF[1]← N ;    //fraction is 1/N
   return EF[0..1]
  else    //recursive step
    tmp← N/M + 1;    // M*tmp must be greater then N
   NewN ← N * tmp;    //computing denominator of M/N - 1/tmp
    NewM ← M * tmp − N;    // computing numerator of M/N - 1/tmp
        // Recursive call:  find Egyptian Fractions for NewM/NewN
   RecEF[]← EgyptFractionRecursive(NewM,NewN);
    EF[0..(RecEF[0]+1])];
    EF[0]←RecEF[0]+1;    // Update number of fractions in decomposition
    EF[1]←tmp;    // add new fraction
    EF[2..Check[0]]←RecEF[1..RecCheck[0]];    //copy all other fractions
    return EF[0..EF[0]];
  endif
end
```

Iterative Algorithm

```
ALGORITHM EgyptFraction(M,N)
begin
   EF[0]← 0;
   i ← 1;     //initialize iteration count
   while M> 0 do
     EF[i]← N/M + 1;    //compute next Egyptian Fraction component
    NewN ← N * EF[i];     //computing denominator of M/N - 1/EF[i]
     M ← M * EF[i] − N;    // computing numerator of M/N - 1/EF[i]
    EF[0]= EF[0]+1;
     N ← NewN;     // prepare for next iteration
     i ← i + 1;
  endwhile
   return EF[0..EF[0]];
end
```