1. メモ化再帰

さて、フィボナッチ数列を求める関数だが、これはｎの値が大きくなると非常に時間がかかってしまう。時間計算量についての詳しい話は後期に回す。

フィボナッチ数列の再帰の流れを追ってみよう。次のような木構造で図式化する。

木の根から関数の最初の呼び出しがある。そこから下へと伸びるのが、再帰呼び出しされた関数である。さて、よく見ると同じ引数の呼び出しがある。同じ項を何度も求めていることになり、処理に無駄がある。

そこで、同じ処理をさせないために、「以前に求めた項はどこかにメモしておいて、必要なときにそれを利用する」ようにすれば、同じ項を何度も求める処理が省け、処理時間が早くなる。

このように、ある状態をメモしておいて必要なときにそれを使う手法をメモ化再帰という。フィボナッチ数列のメモ化再帰は非常に簡単な例である。

メモ化再帰で書いたフィボナッチ数列の第n項を求めるプログラムは次のようになる。

|  |
| --- |
| #include <stdio.h>  int memo[40];  int fib(int n)  {  if(memo[n] != 0) return memo[n];  if(n == 0) return 0;  if(n == 1) return 1;    memo[n] = fib(n - 2) + fib(n - 1);  return memo[n];  }  int main(void)  {  int n;  scanf("%d", &n);  printf("%d\n", fib(n));  return 0;  } |

memoという配列によって、既に計算した項をメモしている。同じ項を計算したくなったとき、memoからその値を取り出している。