Рекурсия

Дисциплина «Алгоритмы дискретной математики»
2 курс 4 семестр

Объект называется рекурсивным, если он содержит сам себя или определен с помощью самого себя

Если процедура p содержит явное обращение к самой себе, то она называется явно рекурсивной. Если процедура содержит обращение к некоторой процедуре q, которая в свою очередь содержит прямое или косвенное обращение к p, то p - называется косвенно рекурсивной.

Рекурсию можно применять во всех случаях, когда в задаче можно выделить самоподобие. Например,

1. Сумма элементов массива равна сумме сумм элементов его подмассивов.
2. Нахождение максимального элемента массива является результатом сравнения первого элемента с максимальным элементом оставшейся части массива.

Использование рекурсии для решения этих задач не всегда эффективно.

Существуют специальные задачи, решаемые обычно с помощью рекурсии:

1. Вычисление факториала целого числа.
2. Вычисление чисел Фибоначчи.
3. Решение головоломки "Ханойские башни".
4. Генерация фракталов.

Мощь рекурсивного определения объекта в том, что такое конечное определение способно описывать бесконечно большое число объектов. С помощью рекурсивной программы же возможно описать бесконечное вычисление, причём без явных повторений частей программы.

Но рекурсивная программа не может вызывать себя бесконечно, иначе она никогда не остановится, таким образом в программе (функции) должна присутствовать еще один важный элемент - так называемое терминальное условие, то есть условие при котором программа прекращает рекурсивный процесс.

Рассмотрим это на примере факториала.

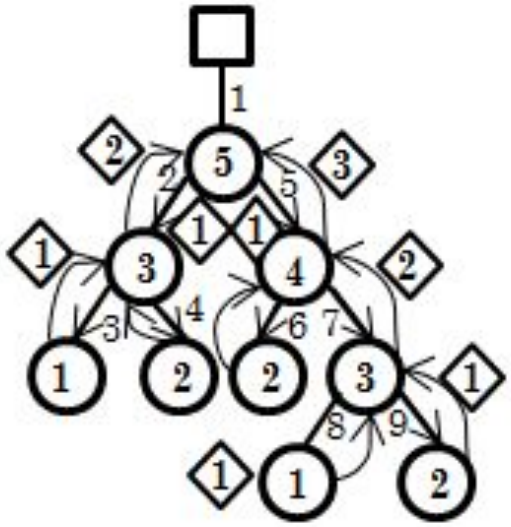
```
int fact(int N)
{
      if(N==1)
         return 1;
      else
         return N*fact(N-1);
 }
```

Числа Фибоначчи определяются с помощью рекурсии:

Первое и второе числа Фибоначчи равны 1 Для $n>2$, n-е число Фибоначчи равно сумме (n-1)-го и (n-2)-го чисел Фибоначчи

```
int  fibonacci (int N)
{
    if (N<=1)
        return 1;
  else
        return fibonacci(N-1) + fibonacci(N-2);
}
```

Состояние стека и дерево рекурсии при вычислении чисел Фибоначчи

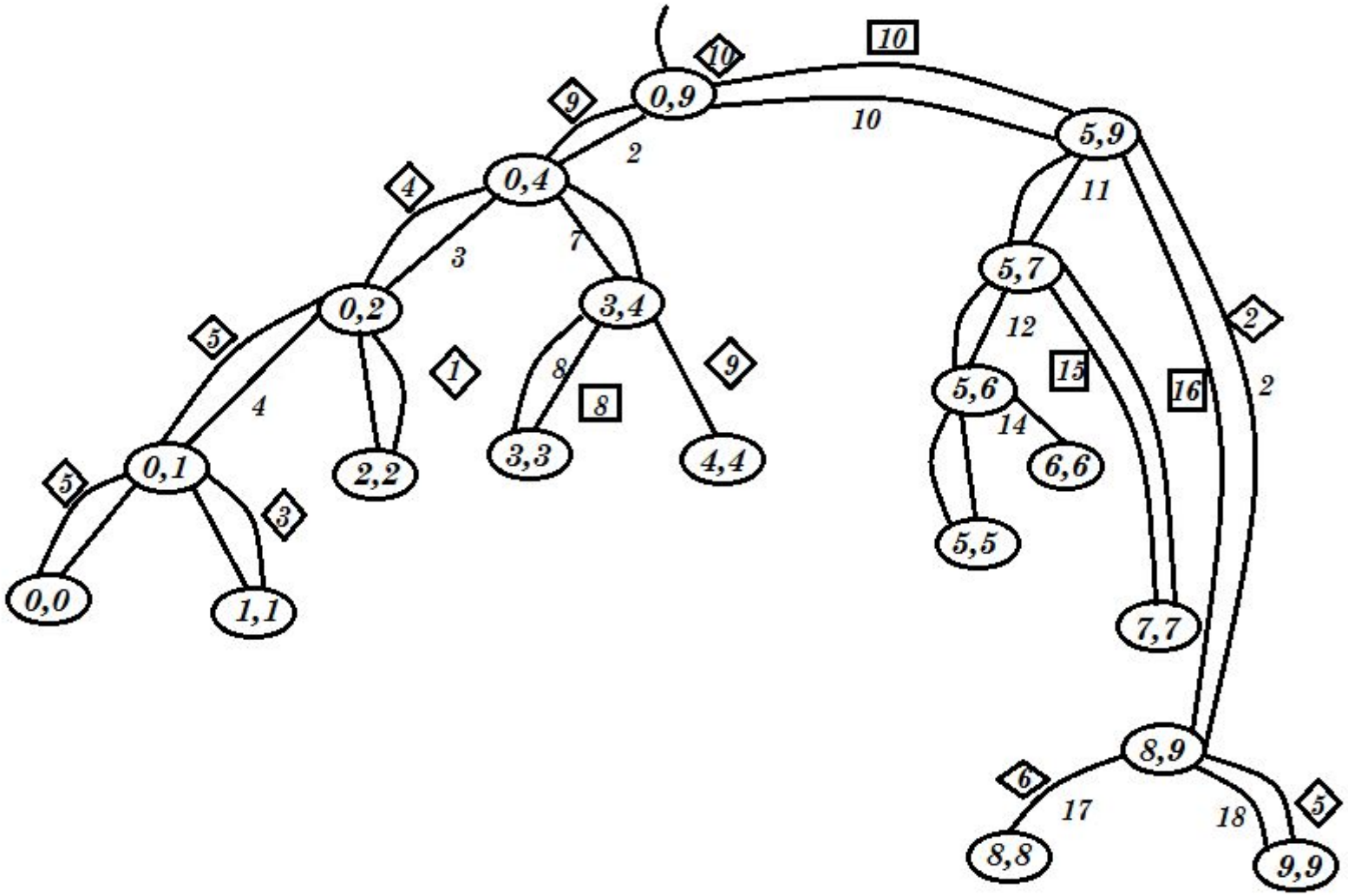



```
{
    ……
f = fibon(n)
fprintf(…f…)
}
intfibon(int n)
{
     if(n<1) return 0;
     if(n==1) return 1;
     return fibon (n-2) + fibon(n-1);
}
```

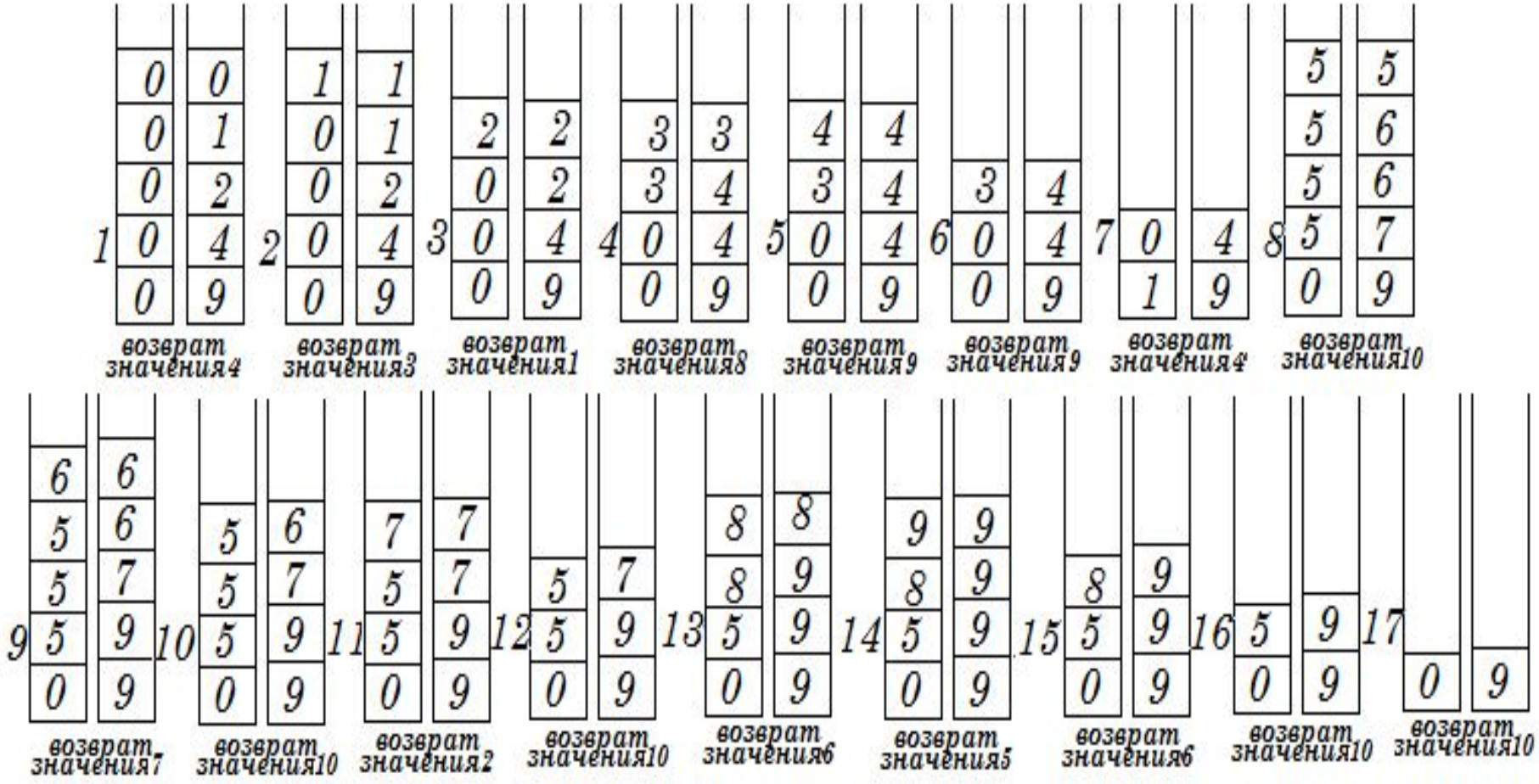
Применение принципа «Разделяй и властвуй» для поиска максимума массива

```
int q[10]= {4, 3, 1, 8, 9, 10, 7, 2, 6, 5};
……….
int max (int, int);   -  прототип функции
   void main (void)
   {
      int m, l = 0, r = 9;
      ……..
      for (int i = 0; i<10; i++) fprintf (fout, “%d…”, a[i]);
      m = max (l, r);
      ……..
    }
int max (int l, int r);
     {
        int mid, n, v
        if (l == r)  return a[l];  - ограничение рекурсии
        mid = (l + r)/2;
        n = max (l, mid);
        v = max (mid+1, r);
        if (n > v)   return n;
        else  return v;
     }
```

Дерево вызовов функции



Состояние стека при работе программы



Ханойские башни

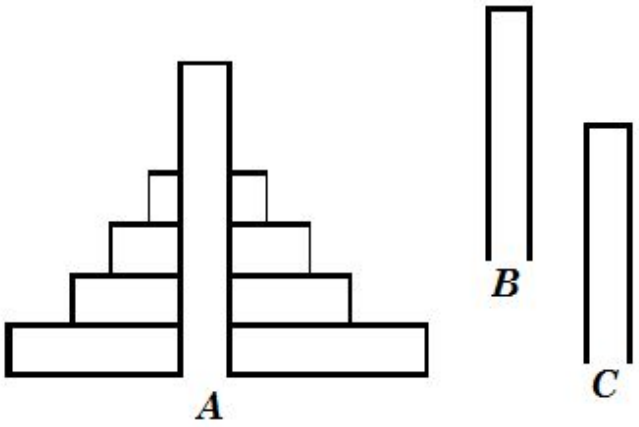
Задача

В одном из буддийских монастырей монахи уже тысячу лет занимаются перекладыванием колец. Они располагают тремя пирамидами, на которых надеты кольца разных размеров. В начальном состоянии 64 кольца были надеты на первую пирамиду и упорядочены по размеру. Монахи должны переложить все кольца с первой пирамиды на вторую, выполняя единственное условие - кольцо нельзя положить на кольцо меньшего размера. При перекладывании можно использовать все три пирамиды. Монахи перекладывают одно кольцо за одну секунду. Как только они закончат свою работу, наступит конец света.

Рекурсивный вариант решения задачи :

Решение

Нужно применить рекурсивно алгоритм, переложив $n-1$ кольцо с первой пирамиды на третью пирамиду. Затем сделать очевидный ход, переложив последнее самое большое кольцо с первой пирамиды на вторую. Затем снова применить рекурсию, переложив $n-1$ кольцо с третьей пирамиды на вторую пирамиду.



Стержни: ***A*** – исходный, ***C*** – целевой, ***B*** – дополнительный, ***n*** – количество дисков.

n=1 => *p*1 = 1 *n* = 3 => (*A*→*C*; *A*→*B*; *C*→*B*; *A*→*C*; *B*→*A*; *B*→*C*; *A*→*C)* => *p*3=7

n=2 => *p*2 = 3 *n* = 4 => (*A*→*B*; *A*→*C*; *B*→*C*; *A*→*B*; *C*→*A*; *C*→*B*; *A*→*B*; *A*→*C*; *B*→*A*; *B*→*C*; *A*→*C*; *C*→*B*; *C*→*B*; *C*→*A*;*B*→*A*; *B*→*C*; *A*→*B*; *A*→*C*; *B*→*C*)

*p*4 = 15

***Pn*=2*Pn*-1 +1.**

Программная реализация

```
void main()
{      .   .   .
       put_disk( n, ’A’, ‘B’, ‘C’);
       .   .   .
}

void put_disk(int n, char s1, char s2, char s3)
{      if (n == 1)        //последняя перестановка единственного диска
       printf (“%c --->%c \n", s1, s3);
       else
{//Всю верхушку башни, кроме самого большого диска,
перекладываем
            // на свободный стержень (временно используется для этого
s2)
put_disk(n - 1, s1, s3, s2); //s1->s2 через s3
//Самый большой диск кладём на 'место назначения'
printf (“%c --->%c \n", s1, s3);
//Перекладываем на него все остальные, используя
освободившийся стержень
put_disk(n-1,s2,s1,s3); //s2->s3 через s1
}
```