МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ
МГПУ

Рекурсия

Определение рекурсии

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Процесс, в котором функция вызывает саму себя прямо или косвенно, называется рекурсией, а соответствующая функция называется рекурсивной функцией.

Используя рекурсивные алгоритмы, некоторые проблемы могут быть решены довольно легко.

Примерами таких проблем являются
Ханойские башни (ТОН),

обходы дерева,

обходы графа.

Определение рекурсии

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ
МГПУ

Термин рекурсия может быть определен как процесс определения чего-либо в терминах самого себя.

Это процесс, в котором функция вызывает саму себя прямо или косвенно.

Преимущества использования рекурсии

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

- **Сложную функцию** можно разбить на более мелкие подзадачи, используя рекурсию.

- **Создание последовательности** проще с помощью рекурсии, чем с использованием любой вложенной итерации.

- **Рекурсивные функции** делают код простым и эффективным.

Недостатки использования рекурсии

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

- **Рекурсивные вызовы** занимают много памяти и времени, что делает его дорогостоящим в использовании.

- **Рекурсивные функции** сложно отлаживать.

- **Обоснование рекурсии** иногда бывает сложно продумать.

Базовое условие в рекурсии

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

В **рекурсивной** программе предоставляется решение базовой задачи, а решение сложной задачи выражается в терминах более легкой задачи.

```
def fact(n):

 # base case
 if (n < = 1)
 return 1
 else
 return n*fact(n-1)
```

Базовое условие в рекурсии

Рекурсивная функция вызывает саму себя, память для вызываемой функции выделяется поверх памяти, выделенной для вызывающей функции, и для каждого вызова функции создается отдельная копия локальных переменных.
Когда достигается базовый вариант, функция возвращает свое значение функции, с помощью которой она вызывается, память освобождается, и процесс продолжается.

```
# A Python 3 program to
# demonstrate working of
# recursion

def printFun(test):

  if (test < 1):
    return
  else:

    print(test, end=" ")
    printFun(test-1) # statement 2
    print(test, end=" ")
    return

# Driver Code
test = 3
printFun(test)
```

Пример 1. Последовательность Фибоначчи - это целочисленная последовательность 0, 1, 1, 2, 3, 5, 8.

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

```
# Program to print the fibonacci series upto n_terms

# Recursive function
def recursive_fibonacci(n):
  if n <= 1:
    return n
  else:
    return(recursive_fibonacci(n-1) + recursive_fibonacci(n-2))

n_terms = 10

# check if the number of terms is valid
if n_terms <= 0:
  print("Invalid input ! Please input a positive value")
else:
  print("Fibonacci series:")
for i in range(n_terms):
  print(recursive_fibonacci(i))
```

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ
МГПУ

Динамическое программирование

Определение

Динамическое программирование - это в основном оптимизация по сравнению с простой рекурсией. Везде, где мы видим рекурсивное решение с повторяющимися вызовами для одних и тех же входных данных, мы можем оптимизировать его с помощью динамического программирования.

Идея состоит в том, чтобы просто сохранить результаты подзадач, чтобы нам не приходилось повторно вычислять их при необходимости позже. Эта простая оптимизация сокращает временные сложности с экспоненциальных до полиномиальных.

Например, если мы напишем простое рекурсивное решение для чисел Фибоначчи, мы получим экспоненциальную временную сложность, а если мы оптимизируем его, сохраняя решения подзадач, временная сложность уменьшается до линейной.

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Определение

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ
МГПУ

```
int fib(int n)
{
    if (n <= 1)
        return n;
    return fib(n-1) + fib(n-2);
}
```

Recursion : Exponential

```
f[0] = 0;
f[1] = 1;

for (i = 2; i <= n; i++)
{
    f[i] = f[i-1] + f[i-2];
}

return f[n];
```

Dynamic Programming : Linear

МОСКОВСКИЙ
ГОРОДСКОЙ
УНИВЕРСИТЕТ
МГПУ

129226, г. Москва, 2-й Сельскохозяйственный проезд, 4
info@mgpu.ru
+7 (499) 181-24-62
www.mgpu.ru

Университет твоих возможностей