Алгоритми та структури даних
Модуль 2: Методи розробки алгоритмів
Лекція 7. Рекурсія. Метод грубої сили.

Кудін Олексій Володимирович
avk256@gmail.com

План

1. Основні поняття рекурсії.
2. Поняття методу розробки алгоритмів.
3. Базові алгоритми сортування.

Питання по попередній лекції

- *Що таке хеш-таблиця?*
- *Які ви знаєте хеш-функції?*
- *Які вимоги висуваються до хеш-функцій?*
- *Що таке колізія?*
- *Методи розв’язання коллізій?*

Рекурсивні об’єкти

Об'єкт називається **рекурсивним**, якщо він містить сам себе або визначений за допомогою самого себе.

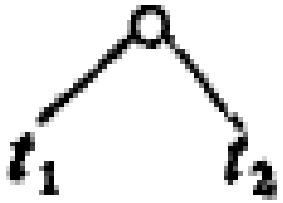
Приклади:

Натуральні числа

- (a) 1 є натуральне число;
- (b) ціле число, що слідує за натуральним, є натуральне число.

Деревоподібні структури

- (a) O є дерево (називане порожнім деревом);
- (b) якщо t1 і t2 - дерева, то



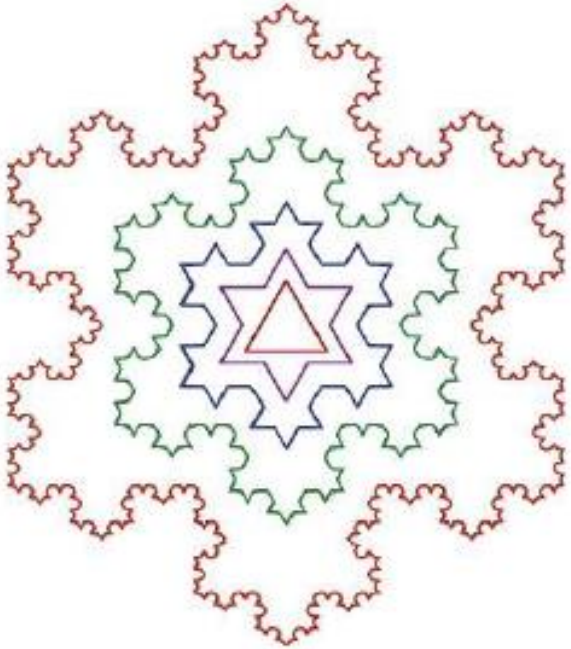
- є дерево (побудоване зверху вниз).

Функція факторіал *n!* для ненегативних цілих чисел:

- (a) 0! = 1,
- (b) якщо n> 0, то n! = n*(n-1)!

Фрактали (самоподібні множини нецілої розмірності)

Треугольники Серпинского



Кривые Коха

Числа Фібоначчі

$$f(n) = \begin{cases} 1, & n \le 2 \\ f(n-1) + f(n-2), & n > 2 \end{cases}$$

```
function Fib(n: Integer): Integer;
begin
        if (n=1) or (n=2) then
                Result:=1
        else
                Result:=Fib(n-
1)+Fib(n-2);
end;
```

Розрізняють пряму.

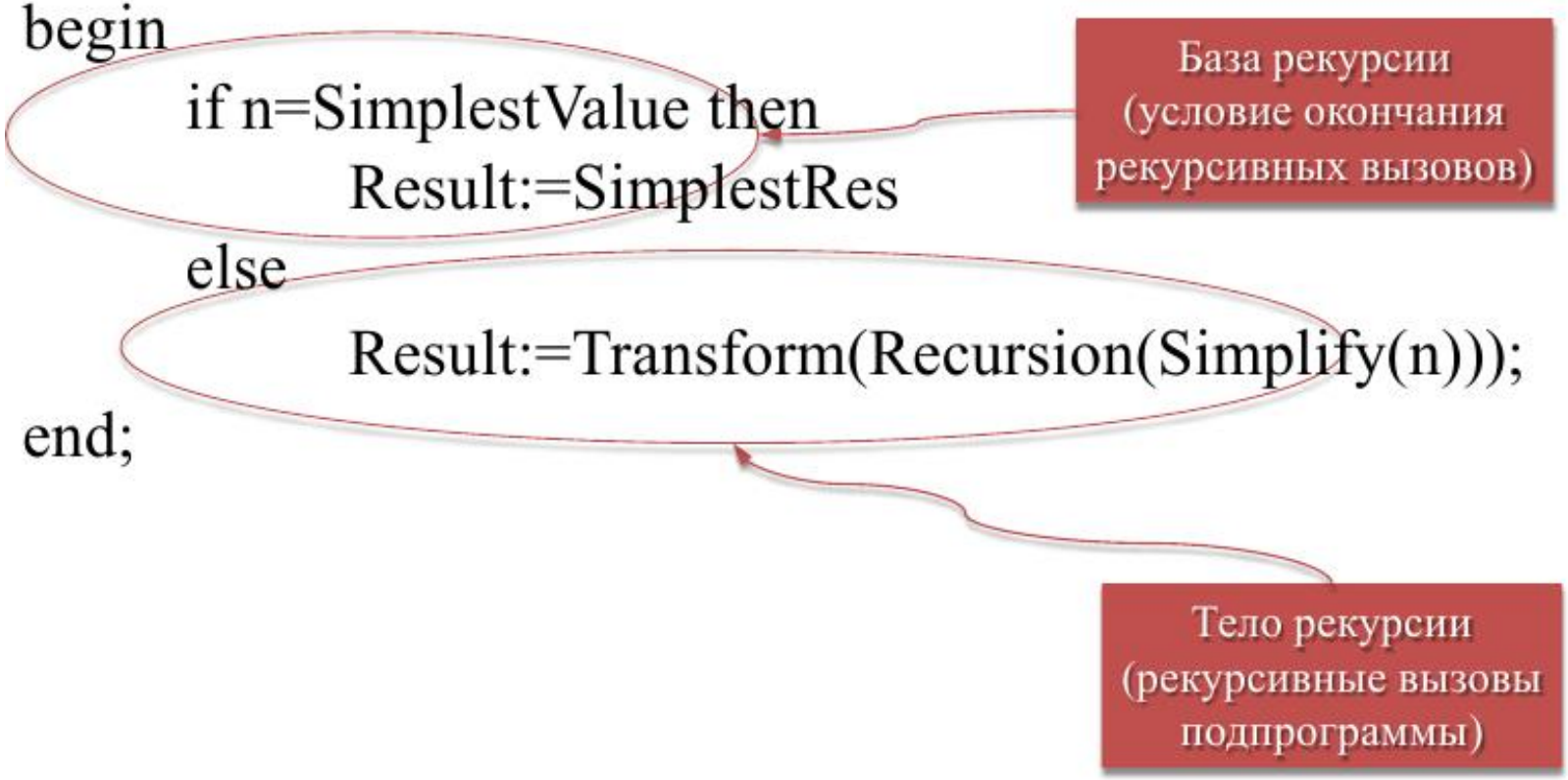
```
procedure Recursion(n: TParam; var Res: TOutput);
begin
        if n=SimplestValue then
                Res:=SimplestRes
        else begin
                Simplify(n);
                Recursion(n, Res);
                Transform(Res);
        end;
end;
```

База рекурсии (условие окончания рекурсивных вызовов)

Тело рекурсии (рекурсивные вызовы подпрограммы)

Розрізняють побічну рекурсію.

```
function Recursion(n: TParam): TOutput;
begin
        if n=SimplestValue then
                Result:=SimplestRes
        else
                Result:=Transform(Recursion(Simplify(n)));
end;
```



Прямий хід рекурсії

Прямий хід рекурсії. Рекурсивний виклик підпрограми залишає в сегменті стека дані, поміщені туди на попередньому виклику підпрограми, і записує на вершину стека дані поточного виклику.

```
function Factorial(n: Integer): Integer;
begin
        if (n=0) or (n=1) then Result:=1
        else
                Result:=n*Factorial(n-1);
end;
```

Factorial(5)

5*Factorial(4)

5*4*Factorial(3)

5*4*3*Factorial(2)

5*4*3*2*Factorial(1)

Зворотний хід рекурсії

Зворотний хід рекурсії. При досягненні бази рекурсії рекурсивні виклики припиняються й починається серія завершень викликів з добуванням зі стека збережених значень і використанням їх для продовження обчислень. Останнім завершується перший виклик.

```
function Factorial(n: Integer): Integer;
begin
        if (n=0) or (n=1) then Result:=1
        else
                Result:=n*Factorial(n-1);
end;
```

Factorial(5)

5*Factorial(4)

5*4*Factorial(3)

5*4*3*Factorial(2)

5*4*3*2*Factorial(1)

1

2*1=2

3*2=6

4*6=24

5*24=120

Приклад обчислення факторіала

```
function Factorial(n: Integer):
Integer;
begin
        if (n=0) or (n=1) then
                Result:=1
        else

        Result:=n*Factorial(n-1);
end;
```

```
function Factorial(n: Integer):
Integer;
var i, Res: Integer;
begin
        Res:=1;
        for i:=2 to n do
                Res:=i*Res;
        Result:=Res;
end;
```

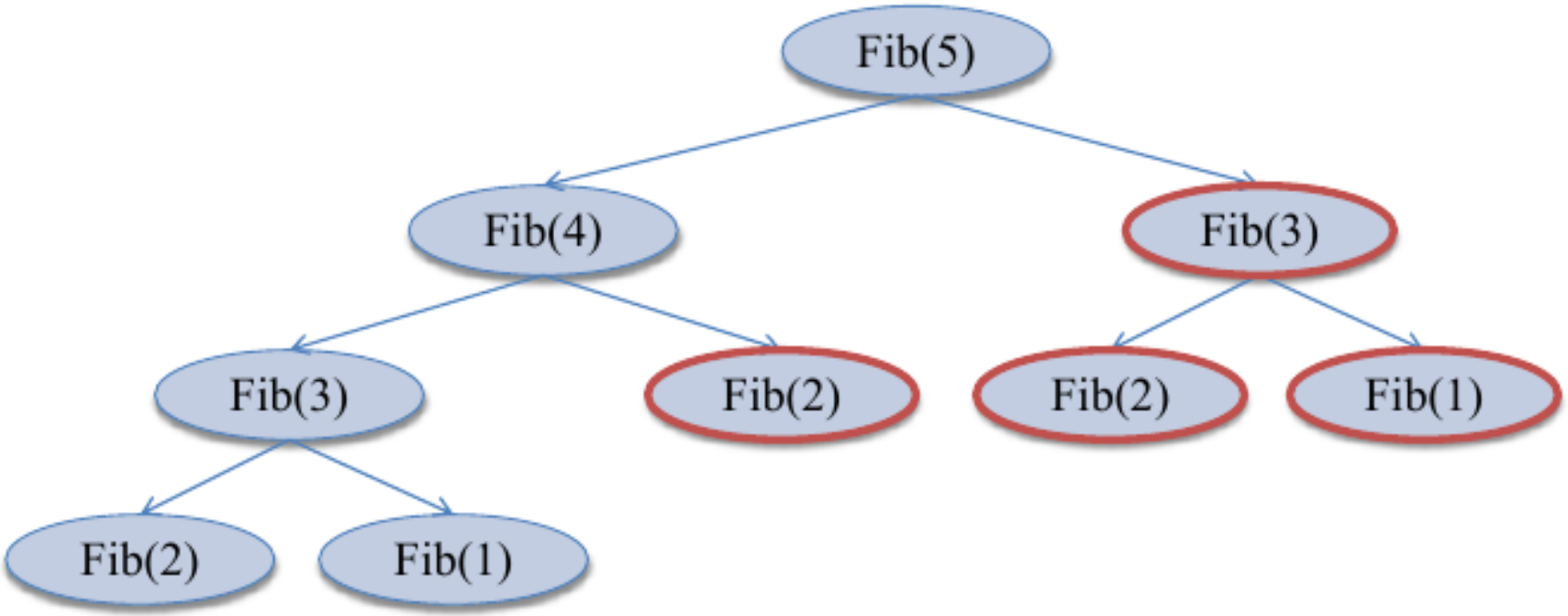
Приклад обчислення послідовності Фібоначчі

```
function Fib(n: Integer): Integer;
begin
    if (n=1) or (n=2) then
        Result:=1
    else
        Result:=Fib(n-1)+Fib(n-2);
end;
```

$$Fib(n) = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right)$$

```
function Fib(n: Integer): Integer;
var i, n1, n2, Res: Integer;
begin
    N1:=1; N2:=1;
    Result:=1;
    for i:=3 to n do begin
        Res:=n1+n2;
        n2:=n1;
        n1:=Res;
    end;
    Result:=Res;
end;
```

Дерево рекурсивних викликів



Переваги та недоліки рекурсивних алгоритмів???

Метод проектування алгоритму (algorithm design technique) (або "стратегія", або "принцип") — це універсальний підхід, застосовуваний для алгоритмічного рішення широкого кола задач, що ставляться до різних областей обчислювальної техніки.

Метод грубої сили являє собою прямий підхід до розв’язання задачі, звичайно заснований безпосередньо на формулюванні задачі й визначеннях використовуваних нею концепцій.

Переваги метода грубої сили

1. Широкий діапазон задач.
2. Для деяких важливих задач метод дає ефективні алгоритми.
3. Вартість розробки більш ефективного алгоритму може виявитись неприйнятною.
4. Метод може виявитись корисним для розв’язання задач невеликого розміру.

Базові алгоритми сортування

До базових алгоритмів сортування відносяться:

1. Сортування бульбашками.
2. Сортування вибором.
3. Сортування вставками.

Сортування бульбашкою

```
template<class T>
void bubbleSort(T a[], long size) {
  long i, j;
  T x;

  for( i=0; i < size; i++) {
    for( j = size-1; j > i; j-- ) {
      if ( a[j-1] > a[j] ) {
      x=a[j-1]; a[j-1]=a[j]; a[j]=x;
    }
  }
}
}
```

Сортування вибором

```
template<class T>
void selectSort(T a[], long size) {
  long i, j, k;
  T x;
  for( i=0; i < size; i++) {
    k=i; x=a[i];
    for( j=i+1; j < size; j++)
      if (  a[j] < x ) {
        k=j; x=a[j];
      }
    a[k] = a[i]; a[i] = x;
  }
}
```

Сортування вставками

```
template<class T>
void insertSort(T a[], long size) {
  T x;
  long i, j;

  for ( i=0; i < size; i++) {
    x = a[i];

    for ( j=i-1; j>=0 && a[j] > x; j--)
      a[j+1] = a[j];
    a[j+1] = x;
  }
}
```

Оцінка часу роботи

Сортування бульбашкою:

$$C(n) = \sum_{i=0}^{n-2} \sum_{j=0}^{n-2-i} 1 = \sum_{i=0}^{n-2} [(n-2-i) - 0 + 1] =$$

$$= \sum_{i=0}^{n-2} (n-1-i) = \frac{(n-1)\,n}{2} \in \Theta\left(n^2\right).$$

Сортування вибором:

$$C(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{n-2} (n-i-1) = \frac{(n-1)\,n}{2}.$$

Сортування вставками ???

Резюме

Резюме (що встигли за лекцію)

- Розглянули основні поняття рекурсії.
- Дали визначення методу розробки алгоритмів.
- Почали розглядати метод грубої сили.
- Розглянули базові алгоритми сортування.

До наступної лекції.

Виконати математичний аналіз сортування вставками.