Алгоритми та структури даних
Модуль 1: Структури даних
Лекція 3. Основи аналізу ефективності алгоритмів(2).

Кудін Олексій Володимирович
avk256@gmail.com

План

1. Математичний аналіз нерекурсивних алгоритмів.
2. Математичний аналіз рекурсивних алгоритмів.
3. Візуалізація алгоритмів.

Питання по попередній лекції

- *Два параметра за якими аналізується ефективність.*
- *Як визначити розмір вхідних даних для задачі визначення простоти числа?*
- *Що таке базова операція алгоритма?*
- *Що таке асимптотичні позначення?*
- *Які класи ефективності ви знаєте?*

Математичний аналіз алгоритмів

Математичний аналіз алгоритмів дозволяє оцінити ефективність роботи алгоритму до його реалізації, тобто теоретично.

План математичного аналізу

Загальний план математичного аналізу ефективності нерекурсивних алгоритмів

1. Виберіть параметр (або параметри), по якому буде оцінюватися розмір вхідних даних алгоритму.

2. Визначите основну операцію алгоритму. (Як правило, вона перебуває в найбільше глибоко вкладеному внутрішньому циклі алгоритму.)

3. Перевірте, чи залежить число виконуваних основних операцій тільки від розміру вхідних даних. Якщо воно залежить і від інших факторів, розглянете при необхідності, як міняється ефективність алгоритму для найгіршого, середнього й найкращого випадків.

4. Запишіть суму, що виражає кількість виконуваних основних операцій алгоритму.

5. Використовуючи стандартні формули й правила підсумовування, спростите отриману формулу для кількості основних операцій алгоритму. Якщо це неможливо, визначите хоча б їхній порядок росту.

Приклад 1. Розглянемо задачу пошуку найбільшого елемента в списку з *n* чисел. Для простоти припустимо, що цей список реалізований у вигляді масиву. Нижче наведений псевдокод алгоритму рішення цієї задачі.

Приклад 1

АЛГОРИТМ $MaxElement\,(A\,[0..n-1])$

// **Входные данные:** массив вещественных чисел $A[0..n-1]$
// **Выходные данные:** возвращается значение наибольшего
// элемента массива A

$maxval \leftarrow A[0]$
for $i \leftarrow 1$ **to** $n-1$ **do**
  if $A[i] > maxval$
    $maxval \leftarrow A[i]$
return $maxval$

Приклад 2. Розглянемо задачу перевірки одиничності елементів. Інакше кажучи, потрібно переконатися, що всі елементи масиву різні. Цю задачу можна вирішити за допомогою наведеного нижче нескладного алгоритму.

Приклад 2

Алгоритм $UniqueElements\,(A\,[0..n-1])$

```
// Входные данные:   массив вещественных чисел A[0..n − 1]
// Выходные данные: возвращается значение "true", если все
//                  элементы массива A различны, и "false"
//                  в противном случае
for i ← 0 to n − 2 do
    for j ← i + 1 to n − 1 do
        if A[i] = A[j]
            return false
return true
```

Приклад 3. Для двох заданих матриць *A* и *B* розміром *n*x*n* визначите часову ефективність алгоритму їхнього множення *C* = *AB*, заснованого на визначенні цієї операції.

Приклад 3

Алгоритм $MatrixMultiplication\,(A\,[0..n-1, 0..n-1]\,, B\,[0..n-1, 0..n-1])$

// Выполняется умножение двух квадратных матриц размером
// $n \times n$. Используется алгоритм, основанный на определении
// этой операции
// **Входные данные:** две квадратные $n \times n$ матрицы A и B
// **Выходные данные:** матрица $C = AB$

for $i \leftarrow 0$ **to** $n-1$ **do**
  for $j \leftarrow 0$ **to** $n-1$ **do**
    $C[i, j] \leftarrow 0.0$
    for $k \leftarrow 0$ **to** $n-1$ **do**
      $C[i, j] \leftarrow C[i, j] + A[i, k] * B[k, j]$
return C

Загальний план математичного аналізу ефективності рекурсивних алгоритмів

1. Виберіть параметр (або параметри), по якому буде оцінюватися розмір вхідних даних алгоритму.
2. Визначите основну операцію алгоритму.
3. Перевірте, чи залежить число виконуваних основних операцій тільки від розміру вхідних даних. Якщо воно залежить і від інших факторів, розглянете при необхідності, як міняється ефективність алгоритму для найгіршого, середнього й найкращого випадків.
4. Складіть рекурентне рівняння, що виражає кількість виконуваних основних операцій алгоритму, і вкажіть відповідні початкові умови.
5. Знайдіть рішення рекурентного рівняння або, якщо це неможливо, визначите хоча б його порядок росту.

Приклад 1

Приклад 1. Обчислимо значення функції факторіала для довільного цілого ненегативного значення *n.*

Так як *n!=1*…*(n-1)*n* для всіх *n*>= *1* і по визначенню 0! = 1, ми можемо обчислити *F(n) = F(n-1)*n* за допомогою наступного рекурсивного алгоритму.

Приклад 1

АЛГОРИТМ $F(n)$

// Рекурсивное вычисление факториала
// **Входные данные:** Целое неотрицательное число n
// **Выходные данные:** Значение $n!$
if $n = 0$
    return 1
else
    return $F(n-1) * n$

Приклад 1

Приклад 1. Обчислимо значення функції факторіала для довільного цілого ненегативного значення *n.*

Так як *n!=1*…*(n-1)*n* для всіх *n*>= *1* і по визначенню 0! = 1, ми можемо обчислити *F(n) = F(n-1)*n* за допомогою наступного рекурсивного алгоритму.

Метод зворотної підстановки

$$M(n) = M(n-1) + 1 =$$
$$= [M(n-2) + 1] + 1 =$$
$$= M(n-2) + 2 =$$
$$= [M(n-3) + 1] + 2 =$$
$$= M(n-3) + 3.$$

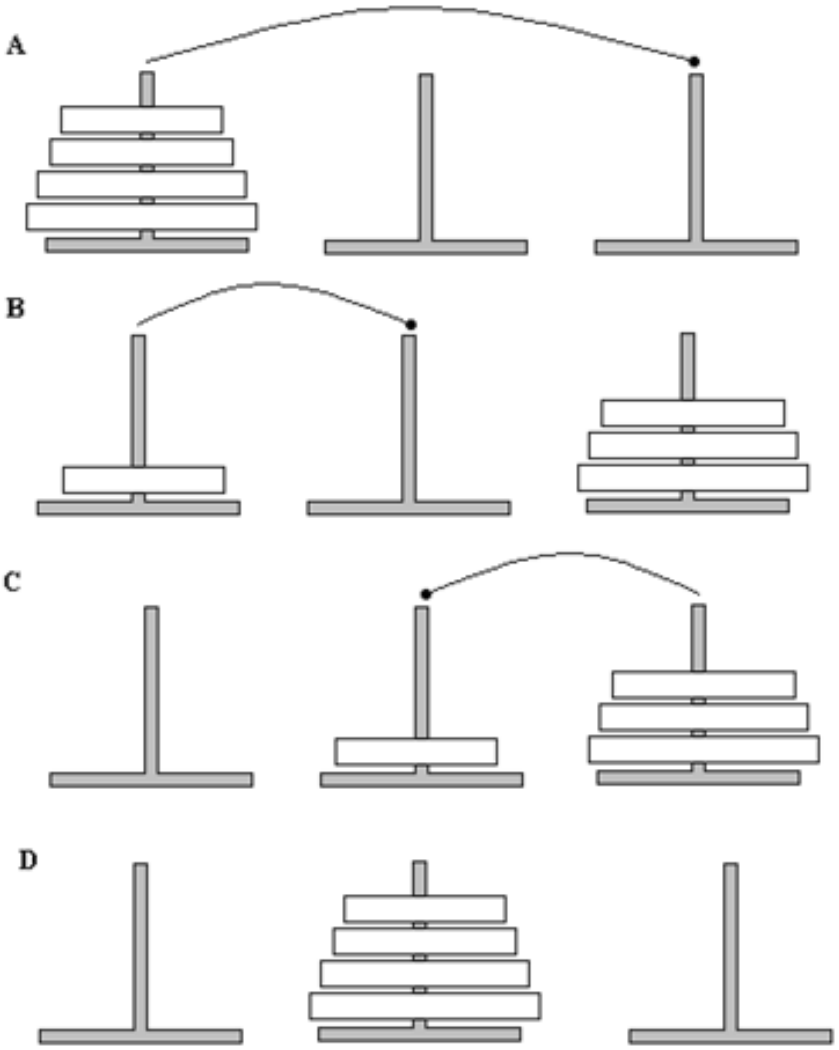
подставляем $M(n-1) = M(n-2) + 1$

подставляем $M(n-2) = M(n-3) + 1$

$$M(n) = M(n-i) + i.$$

$$M(n) = M(n-1) + 1 = \cdots = M(n-i) + i = \cdots = M(n-n) + n = n$$

Приклад 2. Задача про Ханойські башти.



Приклад 2

$$M(n) = 2M(n-1) + 1 \quad \text{для } n > 1,$$
$$M(1) = 1.$$

$$M(n) = 2M(n-1) + 1 = \quad \text{подставляем } M(n-1) = 2M(n-2) + 1$$
$$= 2[2M(n-2) + 1] + 1 =$$
$$= 2^2 M(n-2) + 2 + 1 = \quad \text{подставляем } M(n-2) = 2M(n-3) + 1$$
$$= 2^2[2M(n-3) + 1] + 2 + 1 =$$
$$= 2^3 M(n-3) + 2^2 + 2 + 1$$

$$M(n) = 2^i M(n-i) + 2^{i-1} + 2^{i-2} + \cdots + 2 + 1 = 2^i M(n-i) + 2^i - 1.$$

$$M(n) = 2^{n-1} M(n-(n-1)) + 2^{n-1} - 1 =$$
$$= 2^{n-1} M(1) + 2^{n-1} - 1 = 2^{n-1} + 2^{n-1} - 1 = 2^n - 1.$$

Ефективність алгоритма для різних вхідних даних

Приклад візуалізації алгоритмів сортування - http://www.sorting-algorithms.com/

Найбільш популярні системи загального призначення: BALSA, TANGO, ZEUS, Animated Algorithms.

Резюме (що встигли за лекцію)

- Загальний план математичного аналізу ітеративних та рекурсивних алгоритмів.
- Розглянули у чому відмінність математичного аналізу від емпіричного.
- Розглянули на прикладах метод розв’язання рекурентних рівнянь: метод зворотної підстановки.

Завдання до наступної лекції

- Навчитися спрощувати вирази з сумами.
- Навчитися розв’язувати рекурентні рівняння.