Алгоритми та структури даних
Модуль 2: Методи розробки алгоритмів
Лекція 9. Метод декомпозиції.

Кудін Олексій Володимирович
avk256@gmail.com

План

1. Визначення
2. Бінарний пошук.
3. Сортування злиттям.
4. Швидке сортування.
5. Обхід бінарного дерева.
6. Множення більших цілих чисел і алгоритм множення матриць Штрассена.

Питання по попередній лекції

- *Яка основна ідея метода грубої сили?*
- *Яка часова складність базових алгоритмів сортування?*
- *Що таке метод вичерпного перебору?*
- *Яка часова складність розв'язання задачі коммівояжера методом вичерпного перебору?*

Етапи метода декомпозиції

1. Екземпляр задачі розбивається на кілька менших екземплярів тої ж задачі, в ідеалі - однакового розміру.
2. Вирішуються менші екземпляри задачі (частіше рекурсивно, хоча іноді для невеликих екземплярів застосовується який-небудь інший алгоритм).
3. При необхідності рішення вихідної задачі знаходиться шляхом комбінації рішень менших екземплярів.

Етапи метода декомпозиції

Часова складність декомпозиції

У загальному випадку екземпляр задачі розміру *n* може бути розділений на кілька екземплярів розміром *n/b*, *a* з яких потрібно вирішити (тут *a* й *b* — константи; *a* > *1 і b* > *1*). Полагаючи для спрощення аналізу, що розмір *n* дорівнює степені *b*, одержуємо наступне рекурентне співвідношення для часу роботи алгоритму *T(n)*:

$$T(n) = aT(n/b) + f(n),$$

де *f (n)* — функція, що враховує витрати часу на поділ задачі на менші підзадачи й комбінування рішень підзадач.

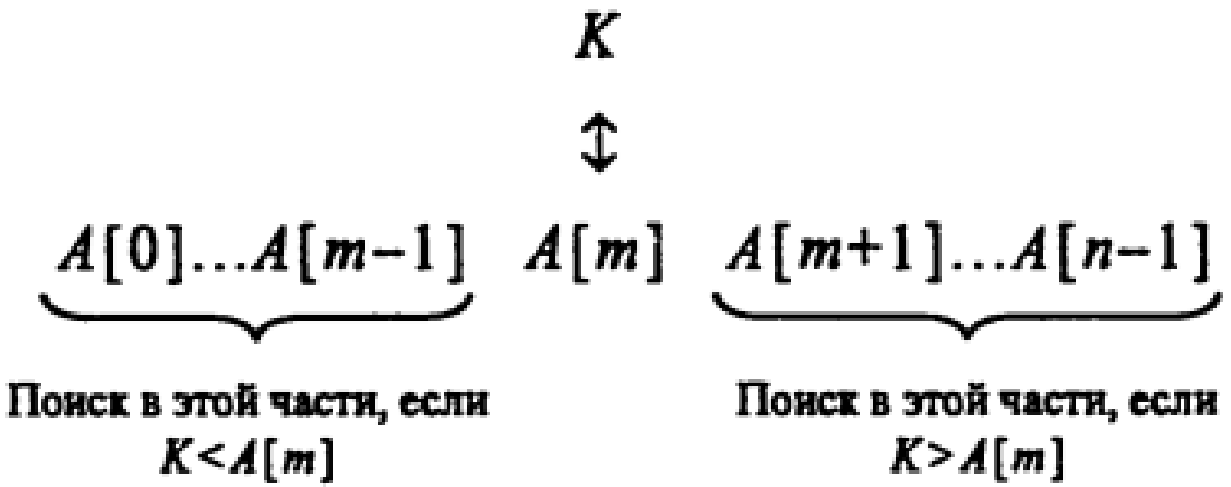
ОСНОВНА ТЕОРЕМА. Якщо в рекурентному рівнянні декомпозиції $f(n) \in \Theta(n^d)$, де d>=0, то

$$T(n) \in \begin{cases} \Theta\left(n^d\right) & \text{если } a < b^d \\ \Theta\left(n^d \log n\right) & \text{если } a = b^d \\ \Theta\left(n^{\log_b a}\right) & \text{если } a > b^d \end{cases}$$

Бінарний пошук

Бінарний пошук – це ефективний алгоритм для пошуку у відсортованому масиві.

Він працює шляхом порівняння шуканого ключа *K* із середнім елементом масиву A [*m*]



Бінарний пошук. Приклад.

Як приклад знайдемо ключ $K = 70$, застосовуючи алгоритм бінарного пошуку до масиву

3	14	27	31	39	42	55	70	74	81	85	93	98

Ітерації алгоритму показані в наступній таблиці:

Индекс	0	1	2	3	4	5	6	7	8	9	10	11	12
Значение	3	14	27	31	39	42	55	70	74	81	85	93	98
Итерация 1	*l*						*m*						*r*
Итерация 2								*l*		*m*			*r*
Итерация 3								*l,m*	*r*				

Бінарний пошук. Код.

АЛГОРИТМ $BinarySearch\,(A\,[0..n-1]\,,K)$

// Реализует нерекурсивный бинарный поиск
// **Входные данные:** Массив $A[0..n-1]$, отсортированный
// в возрастающем порядке, и искомый ключ K
// **Выходные данные:** Индекс элемента массива, равного K,
// или -1, если такого элемента нет

$l \leftarrow 0;\ r \leftarrow n-1;$
while $l \leqslant r$ **do**
  $m \leftarrow \lfloor (l+r)/2 \rfloor$
  if $K = A[m]$
    return m
  else if $K < A[m]$
    $r \leftarrow m-1$
  else
    $l \leftarrow m+1$
return -1

Яка часова складність алгоритму бінарного пошуку?

Сортування злиттям

Алгоритм сортує заданий масив A [0..*n-1*] шляхом поділу його на дві половини, A [0..$\lfloor n/2 \rfloor$-*1*] [$\lfloor n/2 \rfloor$..*n-1*], рекурсивного сортування кожної половини й злиття двох відсортованих половин в один відсортований масив.

Сортування злиттям

Алгоритм $Mergesort\,(A\,[0..n-1])$

// Рекурсивно сортирует массив $A[0..n-1]$
// **Входные данные:** Массив $A[0..n-1]$ упорядочиваемых
// элементов
// **Выходные данные:** Отсортированный в неубывающем порядке
// массив $A[0..n-1]$
if $n > 1$
Копировать $A[0..\lfloor n/2 \rfloor - 1]$ в $B[0..\lfloor n/2 \rfloor - 1]$
Копировать $A[\lfloor n/2 \rfloor..n-1]$ в $C[0..\lfloor n/2 \rfloor - 1]$
$Mergesort(B[0..\lfloor n/2 \rfloor - 1])$
$Mergesort(C[0..\lfloor n/2 \rfloor - 1])$
$Merge(B, C, A)$

Алгоритм злиття

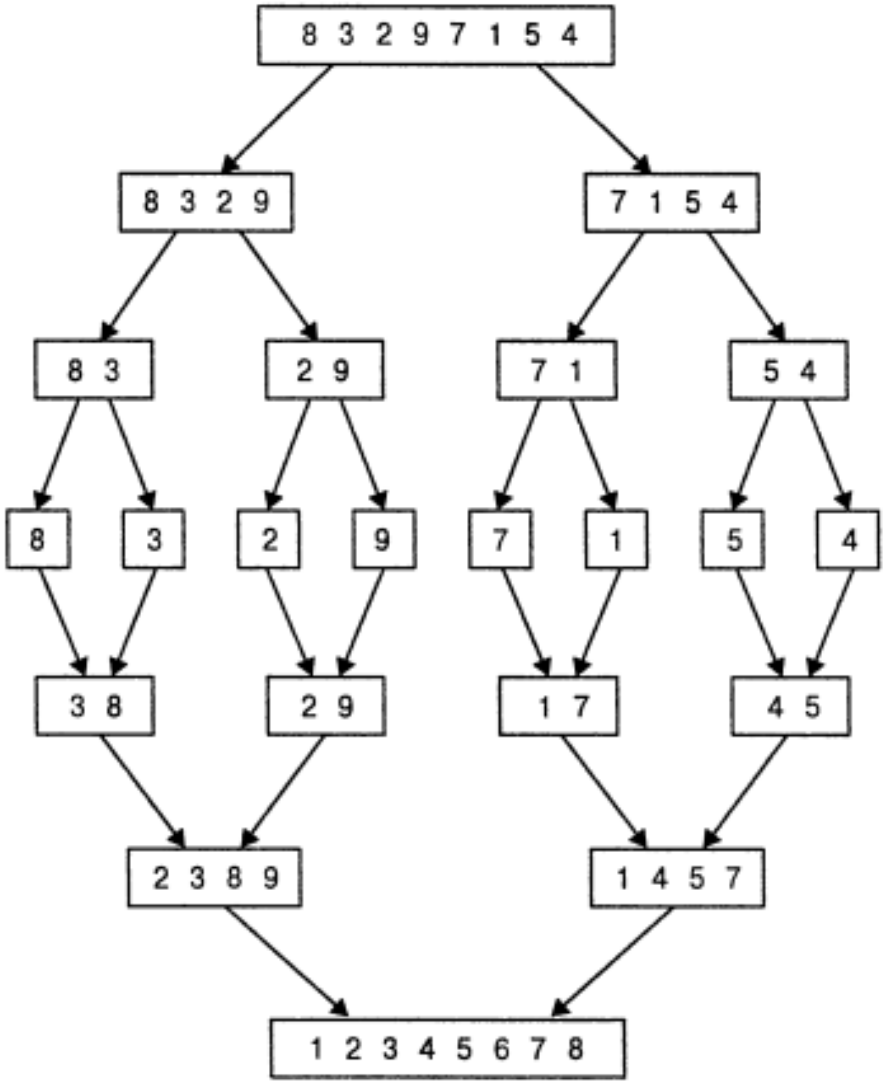
Алгоритм $Merge\,(B\,[0..p-1]\,.C\,[0..q-1]\,,A\,[0..p+q-1])$

// Слияние двух отсортированных массивов в один
// **Входные данные:** Сортированные массивы $B[0..p-1]$
// и $C[0..q-1]$
// **Выходные данные:** Сортированный массив $A[0..p+q-1]$,
// состоящий из элементов B и C

$i \leftarrow 0;\ j \leftarrow 0;\ k \leftarrow 0$
while $i < p$ **and** $j < q$ **do**
  if $B[i] \leqslant C[j]$
    $A[k] \leftarrow B[i];\ i \leftarrow i+1$
  else
    $A[k] \leftarrow C[j];\ j \leftarrow j+1$
  $k \leftarrow k+1$
if $i = p$
  Копировать $C[j..q-1]$ в $A[k..p+q-1]$
else
  Копировать $B[i..p-1]$ в $A[k..p+q-1]$

Сортування злиттям

Приклад. Робота алгоритму зі списком 8,3,2,9,7,1,5,4.



Часова ефективність сортування злиттям

Наскільки ефективне сортування злиттям? Покладемо для простоти, що *n* є ступенем 2. Рекурентне співвідношення для кількості виконуваних порівнянь ключів *C (n)* виглядає в такий спосіб:

$$C(n) = 2C(n/2) + C_{merge}(n) \quad \text{для } n > 1, C(1) = 0.$$

$$C_{worst}(n) = 2C_{worst}(n/2) + n - 1 \quad \text{для } n > 1, C_{worst}(1) = 0.$$

Відповідно до основної теореми?

Швидке сортування

Швидке сортування виконує перестановку елементів даного масиву *A[0..n-1]* для одержання **розбивки** (partition), коли всі елементи до деякої позиції s не перевищують елемента *A[s]*, а елементи після позиції s не менше *A[s]*:

$$\underbrace{A[0] \ldots A[s-1]}_{\text{Все элементы } \leqslant A[s]} \quad A[s] \quad \underbrace{A[s+1] \ldots A[n-1]}_{\text{Все элементы } \geqslant A[s]}.$$

Так може виглядати рекурсивна реалізація алгоритму:

Алгоритм $Quicksort\,(A\,[l..r])$

```
// Сортирует массив методом быстрой сортировки
// Входные данные:  Подмассив A[l..r] массива A[0..n − 1],
//                  определяемый начальным и конечным
//                  индексами l и r
// Выходные данные: Подмассив A[l..r], отсортированный
//                  в неубывающем порядке
if l < r
    s ← Partition(A[l..r])   // s — позиция разбиения
    Quicksort(A[l..s − 1])
    Quicksort(A[s + 1..r])
```

Процедура розбивки

Алгоритм $Partition\,(A\,[l..r])$

// Разбивает подмассив с использованием первого элемента
// в качестве опорного
// **Входные данные:** подмассив $A[l..r]$ массива $A[0..n-1]$,
// определяемый левым и правым
// индексами l и r $(l < r)$
// **Выходные данные:** разбиение $A[l..r]$; при этом позиция
// разбиения возвращается как значение
// функции
$p \leftarrow A[l]$
$i \leftarrow l;\ j \leftarrow r+1$
repeat
  repeat
    $i \leftarrow i+1$
  until $A[i] \geqslant p$
  repeat
    $j \leftarrow j-1$
  until $A[j] \leqslant p$
  $swap(A[i], A[j])$
until $i \geqslant j$
$swap(A[i], A[j])$ // Отмена последнего обмена при $i \geqslant j$
$swap(A[l], A[j])$
return j

Швидкість роботи алгоритму

$$C_{best}(n) = 2C_{best}(n/2) + n \quad \text{при } n > 1, C_{best}(1) = 0.$$

$$C_{worst}(n) = (n+1) + n + \cdots + 3 = \frac{(n+1)(n+2)}{2} - 3 \in \Theta\left(n^2\right).$$

$$C_{avg}(n) \approx 2n \ln n \approx 1.38 n \log_2 n.$$

Як приклад розглянемо рекурсивний *алгоритм визначення висоти бінарного дерева*:

АЛГОРИТМ $Height\,(T)$

// Рекурсивное вычисление высоты бинарного дерева
// **Входные данные:** Бинарное дерево T
// **Выходные данные:** Высота бинарного дерева T
if $T = \varnothing$
    return -1
else
    return $\max\{Height(T_L), Height(T_R)\} + 1$

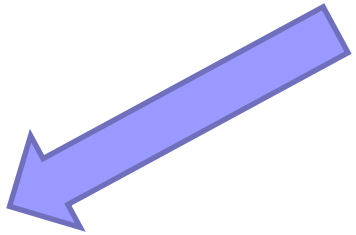
Множення двох цілих чисел

Розглянемо алгоритм грубої сили множення двох цілих чисел:

$$\begin{array}{r} 23 \\ \times 14 \\ \hline 92 \\ 23 \\ \hline 322 \end{array}$$

$23 = 2 \cdot 10^1 + 3 \cdot 10^0$ и $14 = 1 \cdot 10^1 + 4 \cdot 10^0$.

$$23 \cdot 14 = (2 \cdot 10^1 + 3 \cdot 10^0) \cdot (1 \cdot 10^1 + 4 \cdot 10^0) =$$
$$= (2 \cdot 1) \cdot 10^2 + (3 \cdot 1 + 2 \cdot 4) \cdot 10^1 + (3 \cdot 4) \cdot 10^0.$$



$$3 \cdot 1 + 2 \cdot 4 = (2 + 3) \cdot (1 + 4) - (2 \cdot 1) - (3 \cdot 4).$$

Узагальнення множення двох цілих чисел

Для будь-якої пари двозначних чисел a = a_1a_0 і b = b_1b_0 їхній добуток з можна обчислити по формулі:

$$c = a \cdot b = c_2 \cdot 10^2 + c_1 \cdot 10^1 + c_0,$$

де

$c_2=a_1b_1$

$c_0=a_0b_0$

$c_1=(a_1+a_0)(b_1+b_0)-(c_2+c_0)$

$$M(n) = 3M(n/2) \quad \text{при } n > 1, M(1) = 1.$$

$$M(n) = 3^{\log_2 n} = n^{\log_2 3} \approx n^{1.585}.$$

Узагальнення множення двох цілих чисел

При добутку двох *n*-значних чисел *a* й *b*, де *n* — позитивне парне число, необхідно розділити числа навпіл.

$$c = a \cdot b = c_2 \cdot 10^2 + c_1 \cdot 10^1 + c_0,$$

$$a = a_1 \cdot 10^{n/2} + a_0$$

$$b = b_1 \cdot 10^{n/2} + b_0.$$

$$\begin{aligned} c = a \cdot b &= \left(a_1 \cdot 10^{n/2} + a_0\right) \cdot \left(b_1 \cdot 10^{n/2} + b_0\right) = \\ &= (a_1 \cdot b_1) \cdot 10^n + (a_1 \cdot b_0 + a_0 \cdot b_1) \cdot 10^{n/2} + (a_0 \cdot b_0) = \\ &= c_2 \cdot 10^n + c_1 10^{n/2} + c_0, \end{aligned}$$

де

$c_2 = a_1 b_1$ - добуток перших цифр.

$c_0 = a_0 b_0$ - добуток других цифр.

$c_1 = (a_1 + a_0)(b_1 + b_0) - (c_2 + c_0)$

$$M(n) = 3^{\log_2 n} = n^{\log_2 3} \approx n^{1.585}.$$

Алгоритм Штрассена множення матриць

Розглянемо множення двох матриць методом грубої сили:

$$\mathbf{A} = \begin{pmatrix} \mathbf{a}_{0,0} & \mathbf{a}_{0,1} \\ \mathbf{a}_{1,0} & \mathbf{a}_{1,1} \end{pmatrix}$$

$$\mathbf{B} = \begin{pmatrix} b_{0,0} & b_{0,1} \\ b_{1,0} & b_{1,1} \end{pmatrix}$$

$$\mathbf{C} = \mathbf{A} \cdot \mathbf{B} = \begin{pmatrix} \mathbf{a}_{0,0} & \mathbf{a}_{0,1} \\ \mathbf{a}_{1,0} & \mathbf{a}_{1,1} \end{pmatrix} \cdot \begin{pmatrix} b_{0,0} & b_{0,1} \\ b_{1,0} & b_{1,1} \end{pmatrix} = \begin{pmatrix} b_{0,0}\mathbf{a}_{0,0} + b_{0,1}\mathbf{a}_{1,0} & b_{0,0}\mathbf{a}_{0,1} + b_{0,1}\mathbf{a}_{1,1} \\ b_{1,0}\mathbf{a}_{0,0} + b_{1,1}\mathbf{a}_{1,0} & b_{1,0}\mathbf{a}_{0,1} + b_{1,1}\mathbf{a}_{1,1} \end{pmatrix}$$

Алгоритм Штрассена множення матриць

Розглянемо множення двох матриць алгоритмом Штрассена:

$$\begin{bmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} \cdot \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} =$$

$$= \begin{bmatrix} m_1 + m_4 - m_5 + m_7 & m_3 + m_5 \\ m_2 + m_4 & m_1 + m_3 - m_2 + m_6 \end{bmatrix}$$

де

$$m_1 = (a_{00} + a_{11}) \cdot (b_{00} + b_{11})$$
$$m_2 = (a_{10} + a_{11}) \cdot b_{00}$$
$$m_3 = a_{00} \cdot (b_{01} - b_{11})$$
$$m_4 = a_{11} \cdot (b_{10} - b_{00})$$
$$m_5 = (a_{00} + a_{01}) \cdot b_{11}$$
$$m_6 = (a_{10} - a_{00}) \cdot (b_{00} + b_{01})$$
$$m_7 = (a_{01} - a_{11}) \cdot (b_{10} + b_{11}).$$

$$\left[\begin{array}{c|c} C_{00} & C_{01} \\ \hline C_{10} & C_{11} \end{array}\right] = \left[\begin{array}{c|c} A_{00} & A_{01} \\ \hline A_{10} & A_{11} \end{array}\right] \cdot \left[\begin{array}{c|c} B_{00} & B_{01} \\ \hline B_{10} & B_{11} \end{array}\right].$$

$$M(n) = 7^{\log_2 n} = n^{\log_2 7} \approx n^{2.807},$$

Резюме

Резюме (що встигли за лекцію)

- Дали загальне визначення методу декомпозиції.
- Розглянули алгоритм бінарного пошуку.
- Розглянули алгоритм сортування злиттям та швидкого сортування.
- Розглянули алгоритми швидкого множення великих чисел та матриць.