НИУ ВШЭ - Нижний Новгород

November 22, 2024

Алгоритмы и структуры данных

Лекция 3. Рекурсия, сортировки (продолжение)

Илья Сергеевич Бычков
ibychkov@hse.ru

Лекция 3.
Рекурсия, сортировки
(продолжение)

0. План лекции
1. Введение в рекурсию
2. Сортировка слиянием
3. Быстрая сортировка
4. Дополнения

Рекурсия — алгоритмическией механизм, представляющий один из способов организации повторяющихся действий. Механизм решения задачи через сведение ёё к самой себе, но для более простого случая.

Отличительные особенности рекурсии:

- Выполняет повторяющиеся действия для разнымых данных
- Каждое новое выполнение/вызов решает более простую и меньшую по размерам задачу
- Необходим четко определённый базовый случай, для которого ответ тривиален и известен
- Необходим контроль за количеством выполняемых подзадач

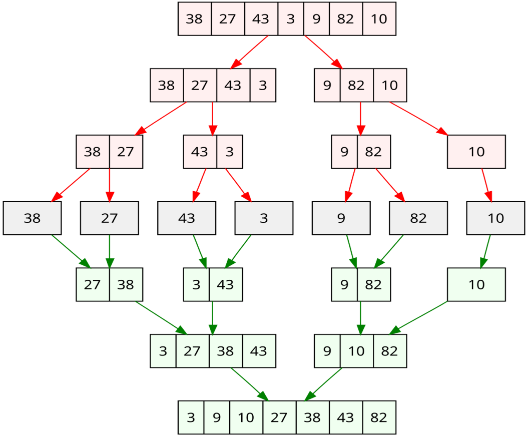
Сортировка слиянием (Merge sort)

В алгоритме используется распространенная алгоритмическая парадигма, известная как “разделяй и властвуй” (divide and conquer).

- Разделение. Определяется правило, по которому задача разбивается на несколько подзадач, которые представляют собой меньшие экземпляры той же самой задачи.
- Властвование. Рекурсивно решаются подзадачи. Если они достаточно малы, они решаются как базовый случай.
- Объединение.Решения подзадач объединяются в решение исходной задачи

Сортировка слиянием
Общая информация

Сортировка слиянием (Merge sort)



(1) Сортировка слиянием, Источник - Wiki

Сортировка слиянием (Merge sort)

```
function mergeSortRecursive(a : int[n]; left, right : int):
    if left + 1 >= right
        return
    mid = (left + right) / 2
    mergeSortRecursive(a, left, mid)
    mergeSortRecursive(a, mid, right)
    merge(a, left, mid, right)
```

(2) Сортировка слиянием - основная функция, Источник - Neerc IFMO

Сортировка слиянием (Merge sort)

```
function merge(a : int[n]; left, mid, right : int):
    it1 = 0
    it2 = 0
    result : int[right - left]

    while left + it1 < mid and mid + it2 < right
        if a[left + it1] < a[mid + it2]
            result[it1 + it2] = a[left + it1]
            it1 += 1
        else
            result[it1 + it2] = a[mid + it2]
            it2 += 1
```

(3) Сортировка слиянием - процедура слияния, Источник - Neerc IFMO

Сортировка слиянием (Merge sort)

```
while left + it1 < mid
    result[it1 + it2] = a[left + it1]
    it1 += 1

while mid + it2 < right
    result[it1 + it2] = a[mid + it2]
    it2 += 1

for i = 0 to it1 + it2
    a[left + i] = result[i]
```

(4) Сортировка слиянием - процедура слияния, Источник - Neerc IFMO

Сортировка слиянием (Merge sort)

Итеративная версия позволяет съэкономить примерно $\mathcal{O}(log_2 n)$ памяти

```
function mergeSortIterative(a : int[n]):
    for i = 1 to n, i *= 2
        for j = 0 to n - i, j += 2 * i
            merge(a, j, j + i, min(j + 2 * i, n))
```

(5) Сортировка слиянием - итеративная версия, Источник - Neerc IFMO

Сортировка слиянием
Сравнения

Proposition. Mergesort uses $\le N \lg N$ compares to sort an array of length N.

Pf sketch. The number of compares $C(N)$ to mergesort an array of length N satisfies the recurrence:

$$C(N) \le \underbrace{C(\lceil N/2 \rceil)}_{\text{left half}} + \underbrace{C(\lfloor N/2 \rfloor)}_{\text{right half}} + \underbrace{N}_{\text{merge}} \quad \text{for } N > 1, \text{ with } C(1) = 0.$$

We solve the recurrence when N is a power of 2: ← result holds for all N (analysis cleaner in this case)

$$D(N) = 2\,D(N/2) + N, \text{ for } N > 1, \text{ with } D(1) = 0.$$

(6) Сортировка слиянием - сравнения, Источник - Alg4cs Princeton - Sedgewick & Wayne

Сортировка слиянием

Операции чтения/записи

Proposition. Mergesort uses $\leq 6N \lg N$ array accesses to sort an array of length N.

Pf sketch. The number of array accesses $A(N)$ satisfies the recurrence:

$$A(N) \leq A(\lceil N/2 \rceil) + A(\lfloor N/2 \rfloor) + 6N \text{ for } N > 1, \text{ with } A(1) = 0.$$

Key point. Any algorithm with the following structure takes $N \log N$ time:

```
public static void linearithmic(int N)
{
    if (N == 0) return;
    linearithmic(N/2);    ←——  solve two problems
    linearithmic(N/2);    ←——  of half the size
    linear(N);        ←————  do a linear amount of work
}
```

(7) Сортировка слиянием - операции чтения, Источник - Alg4cs Princeton - Sedgewick & Wayne

Сортировка слиянием
Улучшения

Use insertion sort for small subarrays.

- Mergesort has too much overhead for tiny subarrays.
- Cutoff to insertion sort for ≈ 10 items.

```
private static void sort(Comparable[] a, Comparable[] aux, int lo, int hi)
{
   if (hi <= lo + CUTOFF - 1)
   {
      Insertion.sort(a, lo, hi);
      return;
   }
   int mid = lo + (hi - lo) / 2;
   sort (a, aux, lo, mid);
   sort (a, aux, mid+1, hi);
   merge(a, aux, lo, mid, hi);
}
```

(8) Сортировка слиянием - улучшения, Источник - Alg4cs Princeton - Sedgewick & Wayne

Сортировка слиянием
Улучшения

Stop if already sorted.

- Is largest item in first half ≤ smallest item in second half?
- Helps for partially-ordered arrays.

A	B	C	D	E	F	G	H	I	J	M	N	O	P	Q	R	S	T	U	V
A	B	C	D	E	F	G	H	I	J	M	N	O	P	Q	R	S	T	U	V

```
private static void sort(Comparable[] a, Comparable[] aux, int lo, int hi)
{
   if (hi <= lo) return;
   int mid = lo + (hi - lo) / 2;
   sort (a, aux, lo, mid);
   sort (a, aux, mid+1, hi);
   if (!less(a[mid+1], a[mid])) return;
   merge(a, aux, lo, mid, hi);
}
```

(9) Сортировка слиянием - улучшения, Источник - Alg4cs Princeton - Sedgewick & Wayne

0. План лекции
1. Введение в рекурсию
2. Сортировка слиянием
3. Быстрая сортировка
4. Дополнения

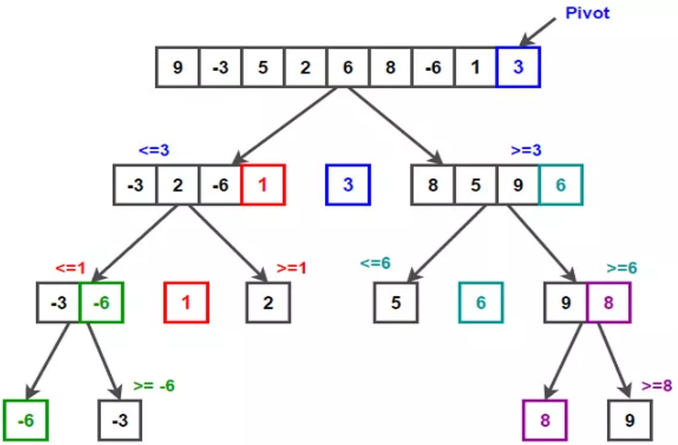
Быстрая сортировка (Quick sort)

В алгоритме также используется парадигма “разделяй и властвуй”.

Общая схема:

- Из массива выбирается некоторый опорный элемент (pivot) a[i]
- Запускается процедура разделения массива, которая перемещает все значения, меньшие, либо равные a[i], влево от него, а все значения, большие, либо равные a[i] – вправо
- Рекурсивно запускаем процедуру для левой и правой частей

Быстрая сортировка (Quick Sort)



(10) Быстрая сортировка, Источник - Неизвестен

Разбиение Ломуто (Lomuto Partition)

```
partition(arr[], lo, hi)
    pivot = arr[hi]
    i = lo - 1       // place for swapping
    for j := lo to hi - 1 do
        if arr[j] <= pivot then
            i = i + 1
            swap arr[i] with arr[j]
    swap arr[i+1] with arr[hi]
    return i+1
```

(11) Разбиение Ломуто, Источник - Geeks4Geeks

Быстрая сортировка

Разбиение Хоара

```
partition(arr[], lo, hi)
   pivot = arr[lo]
   i = lo - 1  // Initialize left index
   j = hi + 1  // Initialize right index

   // Find a value in left side greater
   // than pivot
   do
      i++;
   while (arr[i] < pivot)

   // Find a value in right side smaller
   // than pivot
   do
      j--;
   while (arr[j] > pivot);

   if i >= j then
      return j

   swap arr[i] with arr[j]
```

(12) Разбиение Хоара, Источник - Geeks4Geeks

Быстрая сортировка
Сложность

Quicksort is a (Las Vegas) randomized algorithm.

- Guaranteed to be correct.
- Running time depends on random shuffle.

Average case. Expected number of compares is $\sim 1.39\, n \lg n$.

- 39% more compares than mergesort.
- Faster than mergesort in practice because of less data movement.

Best case. Number of compares is $\sim n \lg n$.

Worst case. Number of compares is $\sim \frac{1}{2} n^2$.

[but more likely that lightning bolt strikes computer during execution]

(13) Быстрая сортировка - Сложность, Источник - Alg4cs Princeton - Sedgewick & Wayne

Goal. Given an array of n items, find the k^{th} smallest item.

Ex. Min $(k = 0)$, max $(k = n - 1)$, median $(k = n / 2)$.

Applications.

- Order statistics.
- Find the "top k."

Use theory as a guide.

- Easy $n \log n$ upper bound. How?
- Easy n upper bound for $k = 1, 2, 3$. How?
- Easy n lower bound. Why?

(14) k-я порядковая статистика, Источник - Alg4cs Princeton - Sedgewick & Wayne

Порядковые статистики

Quick-select

Partition array so that:

- Entry a[j] is in place.
- No larger entry to the left of j.
- No smaller entry to the right of j.

Repeat in one subarray, depending on j; finished when j equals k.

(15) Разбиение, Источник - Alg4cs Princeton - Sedgewick & Wayne

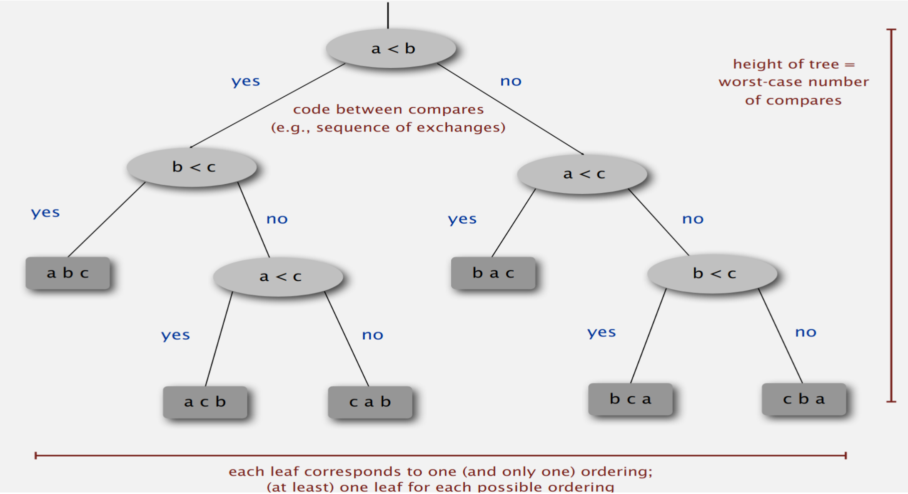
```
public static Comparable select(Comparable[] a, int k)
{
    StdRandom.shuffle(a);
    int lo = 0, hi = a.length - 1;
    while (hi > lo)
    {
       int j = partition(a, lo, hi);
       if      (j < k) lo = j + 1;
       else if (j > k) hi = j - 1;
       else            return a[k];
    }
    return a[k];
}
```

(16) Quick-select, Источник - Alg4cs Princeton - Sedgewick & Wayne

0. План лекции
1. Введение в рекурсию
2. Сортировка слиянием
3. Быстрая сортировка
4. Дополнения

Дополнения

Сортировки на сравнениях



(17) Сортировки на сравнениях, Источник - Alg4cs Princeton - Sedgewick & Wayne

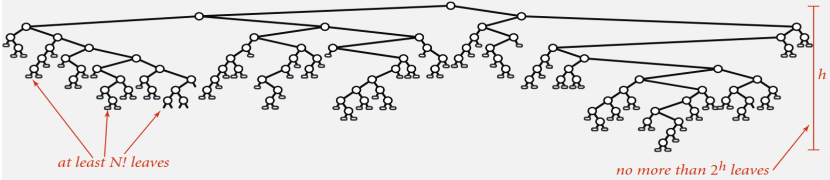
Дополнения

Сортировки на сравнениях

Proposition. Any compare-based sorting algorithm must use at least $\lg(N!) \sim N \lg N$ compares in the worst-case.

Pf.

- Assume array consists of N distinct values a_1 through a_N.
- Worst case dictated by height h of decision tree.
- Binary tree of height h has at most 2^h leaves.
- $N!$ different orderings $\Rightarrow$ at least $N!$ leaves.



(18) Сортировки на сравнениях, Источник - Alg4cs Princeton - Sedgewick & Wayne

Proposition. Any compare-based sorting algorithm must use at least $\lg(N!) \sim N \lg N$ compares in the worst-case.

Pf.

- Assume array consists of N distinct values a_1 through a_N.
- Worst case dictated by height h of decision tree.
- Binary tree of height h has at most 2^h leaves.
- $N!$ different orderings $\Rightarrow$ at least $N!$ leaves.

$$2^h \geq \#\,\text{leaves} \geq N!$$
$$\Rightarrow\ h \geq \lg(N!) \sim N \lg N$$

Stirling's formula

(19) Идея доказательства, Источник - Alg4cs Princeton - Sedgewick & Wayne

Дополнения
Стабильность

A typical application. First, sort by name; then sort by section.

`Selection.sort(a, new Student.ByName());`

Andrews	3	A	664-480-0023	097 Little
Battle	4	C	874-088-1212	121 Whitman
Chen	3	A	991-878-4944	308 Blair
Fox	3	A	884-232-5341	11 Dickinson
Furia	1	A	766-093-9873	101 Brown
Gazsi	4	B	766-093-9873	101 Brown
Kanaga	3	B	898-122-9643	22 Brown
Rohde	2	A	232-343-5555	343 Forbes

`Selection.sort(a, new Student.BySection());`

Furia	1	A	766-093-9873	101 Brown
Rohde	2	A	232-343-5555	343 Forbes
Chen	3	A	991-878-4944	308 Blair
Fox	3	A	884-232-5341	11 Dickinson
Andrews	3	A	664-480-0023	097 Little
Kanaga	3	B	898-122-9643	22 Brown
Gazsi	4	B	766-093-9873	101 Brown
Battle	4	C	874-088-1212	121 Whitman

@#%&@! Students in section 3 no longer sorted by name.

A stable sort preserves the relative order of items with equal keys.

(20) Стабильность сортировки - Пример, Источник - Alg4cs Princeton - Sedgewick &

Дополнения

Стабильность - сортировка вставками

Proposition. Insertion sort is stable.

```
public class Insertion
{
   public static void sort(Comparable[] a)
   {
      int N = a.length;
      for (int i = 0; i < N; i++)
         for (int j = i; j > 0 && less(a[j], a[j-1]); j--)
            exch(a, j, j-1);
   }
}
```

i	j	0	1	2	3	4
0	0	B_1	A_1	A_2	A_3	B_2
1	0	A_1	B_1	A_2	A_3	B_2
2	1	A_1	A_2	B_1	A_3	B_2
3	2	A_1	A_2	A_3	B_1	B_2
4	4	A_1	A_2	A_3	B_1	B_2
		A_1	A_2	A_3	B_1	B_2

Pf. Equal items never move past each other.

(21) Стабильность, вставки - Пример, Источник - Alg4cs Princeton - Sedgewick

Дополнения

Стабильность - сортировка выбором

Proposition. Selection sort is not stable.

```
public class Selection
{
   public static void sort(Comparable[] a)
   {
      int N = a.length;
      for (int i = 0; i < N; i++)
      {
         int min = i;
         for (int j = i+1; j < N; j++)
            if (less(a[j], a[min]))
               min = j;
         exch(a, i, min);
      }
   }
}
```

i	min	0	1	2
0	2	B_1	B_2	A
1	1	A	B_2	B_1
2	2	A	B_2	B_1
		A	B_2	B_1

Pf by counterexample. Long-distance exchange can move one equal item past another one.

(22) Стабильность, выбор - Пример, Источник - Alg4cs Princeton - Sedgewick &

Дополнения

Стабильность - слияние

Proposition. Merge operation is stable.

```
private static void merge(...)
{
   for (int k = lo; k <= hi; k++)
      aux[k] = a[k];

   int i = lo, j = mid+1;
   for (int k = lo; k <= hi; k++)
   {
      if      (i > mid)              a[k] = aux[j++];
      else if (j > hi)               a[k] = aux[i++];
      else if (less(aux[j], aux[i])) a[k] = aux[j++];
      else                           a[k] = aux[i++];
   }
}
```

0	1	2	3	4		5	6	7	8	9	10
A_1	A_2	A_3	B	D		A_4	A_5	C	E	F	G

Pf. Takes from left subarray if equal keys.

(23) Стабильность, слияние - Пример, Источник - Alg4cs Princeton - Sedgewick

Дополнения

Стабильность - быстрая сортировка

Proposition. Quicksort is not stable.

Pf. [by counterexample]

i	j	0	1	2	3
		B_1	C_1	C_2	A_1
1	3	B_1	C_1	C_2	A_1
1	3	B_1	A_1	C_2	C_1
0	1	A_1	B_1	C_2	C_1

(24) Стабильность, быстрая - Пример, Источник - Alg4cs Princeton - Sedgewick & Wayne

Дополнения

Сложность сортировок

	inplace?	stable?	best	average	worst	remarks
selection	✔		$\frac{1}{2} N^2$	$\frac{1}{2} N^2$	$\frac{1}{2} N^2$	N exchanges
insertion	✔	✔	N	$\frac{1}{4} N^2$	$\frac{1}{2} N^2$	use for small N or partially ordered
shell	✔		$N \log_3 N$	?	$c N^{3/2}$	tight code; subquadratic
merge		✔	$\frac{1}{2} N \lg N$	$N \lg N$	$N \lg N$	$N \log N$ guarantee; stable
timsort		✔	N	$N \lg N$	$N \lg N$	improves mergesort when preexisting order
?	✔	✔	N	$N \lg N$	$N \lg N$	holy sorting grail

(25) Сортировки, Источник - Alg4cs Princeton - Sedgewick & Wayne

Дополнения

Сложность сортировок

Array Sorting Algorithms

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Quicksort	$\Omega(n \log(n))$	$\Theta(n \log(n))$	$O(n^2)$	$O(\log(n))$
Mergesort	$\Omega(n \log(n))$	$\Theta(n \log(n))$	$O(n \log(n))$	$O(n)$
Timsort	$\Omega(n)$	$\Theta(n \log(n))$	$O(n \log(n))$	$O(n)$
Bubble Sort	$\Omega(n)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$
Insertion Sort	$\Omega(n)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$
Selection Sort	$\Omega(n^2)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$
Bucket Sort	$\Omega(n+k)$	$\Theta(n+k)$	$O(n^2)$	$O(n)$
Radix Sort	$\Omega(nk)$	$\Theta(nk)$	$O(nk)$	$O(n+k)$
Counting Sort	$\Omega(n+k)$	$\Theta(n+k)$	$O(n+k)$	$O(k)$