SET 2 Алгоритмы «Разделяй-и-властвуй». Методы решения рекуррентных соотношений. Основная теорема

Алгоритмы и структуры данных-1
2023/2024 учебный год

Дедлайн: 23.10.2023 22:00

Инструкция

Задания в рамках домашней работы подразделяются на два блока:

1. Блок А «*Аналитические задачи*» – решения задач A1-A4 оформляются в письменном в виде в любом удобном формате (TeX, скан, фото и др.).
2. Блок Р «*Задачи на разработку*» – решения задач P1-P3 загружаются в систему Яндекс.Контест и проходят автоматизированное тестирование.

Блок А				**Блок Р**			**Итого**
A1	**A2**	**A3**	**A4**	**P1**	**P2**	**P3***	
7	5	5	6	10	15	20	48 (68*)

Задачи, помеченные *, не являются обязательными для решения (относятся к *бонусным*). Подтверждение представленных решений бонусных заданий обязательно сопровождается *устной защитой*.

Плагиат влечет за собой обнуление результатов у всех вовлеченных лиц.

Удачи!

SET 2 Алгоритмы «Разделяй-и-властвуй». Методы решения рекуррентных соотношений. Основная теорема

Алгоритмы и структуры данных-1
2023/2024 учебный год

Дедлайн: 23.10.2023 22:00

Блок A. Проектирование и анализ временной сложности алгоритмов «Разделяй-и-властвуй»

Задание A1 (7 баллов) Временная сложность рекурсивных алгоритмов

Даны два рекурсивных алгоритма обработки целочисленного массива *A* размера *n*:

```
algorithm1(A, n)
1  if n ≤ 20
2      return A[n]
3  x = algorithm1(A, n − 5)
4  for i = 1 to ⌊n/2⌋
5      for j = 1 to ⌊n/2⌋
6          A[i] = A[i] − A[j]
7  x = x + algorithm1(A, n − 8)
8  return x
```

```
algorithm2(A, n)
  if n ≤ 50
      return A[n]
  x = algorithm2(A, ⌊n/4⌋)
  for i = 1 to ⌊n/3⌋
      A[i] = A[n − i] − A[i]
  x = x + algorithm2(A, ⌊n/4⌋)
  return x
```

1. (2 балла) Для каждого из представленных алгоритмов составить *рекуррентное соотношение*, которое выражает их временную сложность $T(n)$. Обратите внимание, что рекуррентное соотношение должно давать полное представление о сложности алгоритма, т.е., охватывать как рекурсивную, так и нерекурсивную ветку вычислений (*краевые условия*). Все арифметические операции выполняются за постоянное время.
2. (5 баллов) Вычислите *асимптотическую точную границу* $\Theta(f(n))$ временной сложности для каждого из представленных алгоритмов. Обоснуйте свой ответ с помощью метода подстановки, дерева рекурсии или индукции.

Задание A2 (5 баллов) Применение основной теоремы о рекуррентных соотношениях

Дан ряд рекуррентных соотношений, которые описывают временную сложность некоторых рекурсивных алгоритмов (при $n = 1$ во всех случаях принимаем $T(1) = 1$):

- $T(n) = 7 \cdot T(n/3) + n^2$,
- $T(n) = 4 \cdot T(n/2) + \log n$,
- $T(n) = 0{,}5 \cdot T(n/2) + 1/n$,
- $T(n) = 3 \cdot T(n/3) + n/2$.

1. (2 балла) Для приведенных рекуррентных соотношений вычислите асимптотическую точную границу $\Theta(f(n))$ временной сложности, применяя основную теорему о рекуррентных соотношениях (master-теорему), если это возможно. Если применение master-теоремы невозможно, поясните почему.
2. (3 балла) Для рекуррентного(-ых) соотношения(-ий), не разрешимых с помощью master-теоремы, вычислите возможную асимптотическую верхнюю границу $O(f(n))$, используя метод подстановки.

Дедлайн: 23.10.2023 22:00

Задание А3 (5 баллов) Быстрее алгоритма Штрассена

Вы планируете разработать алгоритм ***MULT***, предназначенный для умножения двух квадратных матриц A и B размера $N \times N$ и асимптотически более эффективный, чем алгоритм Штрассена. Разрабатываемый алгоритм будет также использовать стратегию «Разделяй-и-властвуй».

1. Исходные матрицы A и B разделяются на фрагменты размера ${}^{N}/_{4} \times {}^{N}/_{4}$ для дальнейшей рекурсивной обработки.
2. Временные затраты на выполнение шагов DIVIDE и COMBINE вместе составляют $\Theta(N^2)$.

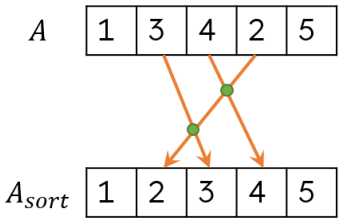
Таким образом, временная сложность алгоритма ***MULT*** будет описываться следующим рекуррентным соотношением: $T(N) = a \cdot T({}^{N}/_{4}) + \Theta(N^2)$, где коэффициент a отвечает за количество решаемых подзадач. Например, для алгоритма Штрассена в соответствии с рекуррентным соотношением $T(N) = 7 \cdot T({}^{N}/_{2}) + \Theta(N^2)$ известно, что для каждой задачи создается 7 подзадач вдвое меньшего размера.

*В каком диапазоне должен находиться параметр a разрабатываемого вами алгоритма **MULT**, для того чтобы он был асимптотически более эффективным по времени в сравнении с алгоритмом Штрассена? Обоснуйте свой ответ.*

Задание А4 (6 баллов) Значительные перестановки

Рассмотрим механизм подсчета «степени упорядоченности» некоторого целочисленного массива $A = [a_1, a_2, a_3, \dots, a_n]$, заполненного *уникальными* числами.

Элементы a_i и a_j массива A назовем *переставленными*, если $i < j$, но $a_i > a_j$. Например, в массиве $A = [1, 3, 4, 2, 5]$ необходимо выполнить две перестановки, а именно $3 \to 2$ и $4 \to 2$ (см. на рисунке ниже), чтобы получить отсортированный массив $A_{sort} = [1, 2, 3, 4, 5]$.



1. (4 балла) Разработайте DaC-алгоритм, сложность которого соответствует $O(n \log n)$, для подсчета степени упорядоченности массива путем вычисления количества необходимых перестановок. Опишите суть шагов DIVIDE, CONQUER и COMBINE, а также представьте рекуррентное соотношение для $T(n)$ и обоснуйте соответствие требуемой сложности.
2. (2 балла) Элементы a_i и a_j массива A назовем *значительно переставленными*, если $i < j$, но $a_i > 2 \cdot a_j$. Какие изменения необходимо внести в алгоритм, разработанный на предыдущем шаге с тем, чтобы в качестве степени упорядоченности велся подсчет количества пар значительно переставленных элементов?
   Например, в массиве $A = [1, 3, 4, 2, 5]$ нет значительно переставленных элементов, а в массиве $B = [5, 3, 2, 4, 1]$ всего 4 пары значительно переставленных элементов: $5 \to 2$, $5 \to 1$, $3 \to 1$ и $4 \to 1$. *Сложность измененного алгоритма должна остаться $O(n \log n)$.*

Дедлайн: 23.10.2023 22:00

Блок P. Реализация алгоритмов «Разделяй-и-властвуй»

Задание P1 (10 баллов) Самое длинное пересечение заданных интервалов

Интервал $[a, b]$ – это упорядоченное множество целых чисел от a до b. Например, интервалу $[16, 23]$ соответствует множество $\{16, 17, 18, 19, 20, 21, 22, 23\}$. *Длиной интервала* назовем количество элементов в соответствующем множестве.

Перекрытием двух интервалов $[a, b]$ и $[c, d]$ назовем другой интервал, содержащий результат пересечения множеств, соответствующих исходным интервалам. Например, перекрытию интервалов $[1, 5]$ и $[3, 6]$ соответствует интервал $[3, 5]$, длина которого равна 3.

Требуется реализовать алгоритм по стратегии «разделяй-и-властвуй», который получает на вход конечный список из n заданных интервалов $[a_1, b_1], [a_2, b_2], \dots, [a_n, b_n]$ и выдает в ответ длину самого длинного перекрытия между парой интервалов среди заданных, а также интервал, соответствующий самому длинному перекрытию. *Асимптотическая сложность алгоритма - $O(n \log n)$, где n – количество интервалов.*

Представление интервала $[a, b]$ должно быть реализовано в виде отдельной структуры данных с методами для вычисления его длины и поиска перекрытия с другим интервалом:

```
struct Interval {
    int left;
    int right;

    size_t length();
    Interval overlap(const Interval& other);
}
```

Использование встроенных методов сортировки, поиска и проч. не допускается.

Примеры работы алгоритма представлены ниже:

Ввод	Вывод
3 1 2 1 5 2 4	3 2 4 *Самое длинное перекрытие у интервалов $[1, 5]$ и $[2, 4]$. Оно соответствует интервалу $[2, 4]$ и имеет длину 3.*
2 -3 5 -5 -4	0 *Входные интервалы не перекрываются*

Дедлайн: 23.10.2023 22:00

Задание P2 (15 баллов) Умножение чисел методом Карацубы

Требуется реализовать «разделяй-и-властвуй» алгоритм умножения двух неотрицательных чисел с большим количеством разрядов, используя алгоритма Карацубы.

$$123456 \cdot 789012 = (123 \cdot 10^3 + 456)(789 \cdot 10^3 + 12)$$

$$A = 123 \cdot 789, B = 456 \cdot 12, C = (123 + 456)(789 + 12), D = C - A - B$$

$$123456 \cdot 789012 = A \cdot 10^6 + D \cdot 10^3 + B = 97408265472$$

В соответствии с алгоритмом, все умножения выполняются рекурсивно до тех пор, пока не будет получен базовый случай «табличного» умножения двух отдельных разрядов.

Реализуйте хранение чисел в виде отдельной структуры `Number` с перегруженным оператором умножения «*»:

```
struct Number {
    std::string digits;

    Number operator* (const Number& other);
}
```

Реализация других методов и полей этой структуры остается на ваше усмотрение.

Примеры работы алгоритма представлены ниже:

Ввод	Вывод
952263744 9359517973	8912729627004270912
4219788824 2743656178	11577649676822954672
2968434375 517784556	1537009474874512500

На вход алгоритма подаются два неотрицательных числа в виде двух строк, количество разрядов в которых не превосходит $5 \cdot 10^4$.

Задание P3* (20 баллов) CLOSEST PAIR 2D

Требуется реализовать «разделяй-и-властвуй» алгоритм для поиска минимального расстояния между парой точек среди некоторого конечного множества точек, заданных своими координатами в двумерном пространстве.

Специальных требований к представлению и хранению точки в виде отдельной структуры данных не предъявляется.

Полная спецификация задания будет доступна в системе тестирования Яндекс.Контест.