Теорія алгоритмів

Практичне завдання №4

“Піраміди”

Завдання

В даній роботі необхідно розв'язати наступну задачу визначення послідовності медіан для заданого вхідного масиву. Нагадаємо, що медіаною для масиву називається елемент, який займає середнє положення у відсортованому масиві. Так, якщо кількість елементів у масиві непарна, то медіана одна та індекс її у відсортованому масиві визначається як $[n/2]$ (де n — розмір вхідного масиву). Якщо кількість елементів у масиві парна, то медіан буде дві та їх індекси визначаються за формулами $[n/2]$ та $[n/2] + 1$.

Задача формулюється наступним чином. Нехай заданий вхідний масив $A = [x_1, ..., x_N]$. Припустимо, що елементи масиву поступають на вхід програми послідовно: в кожний момент часу розглядається новий елемент x_i. Необхідно для кожного i (від 1 до N) визначити медіану підмасиву $A' = [x_1, ..., x_i]$, тобто медіану для масиву елементів, які були отримані програмою на даний момент часу. Необхідно розв’язати цю задачу, використовуючі структури даних пірамід і так, щоб кожна медіана визначалась за час $O(\log(i))$.

Цю задачу можна розв'язати, використовуючи дві піраміди (heap) наступним чином.

- Позначимо через H_{low} незростаючу піраміду (max-heap), яка буде містити елементи меншої половини масиву (тобто такі елементи, які у відсортованому поточному елементі A' будуть розташовуватись у першій, меншій половині масиву).
- Позначимо через H_{high} неспадну піраміду (min-heap), яка буде містити елементи більшої половини масиву (тобто такі елементи, які у відсортованому поточному елементі A' будуть розташовуватись у другій, більшій половині масиву).

Тепер розглянемо роботу процедури, яка розв'язує поставлену задачу із використанням двох наведених пірамід. Нехай додається черговий елемент x_i. На поточний момент сумарна кількість елементів, які зберігаються в обох пірамідах, становить $(i\text{-}1)$. Наступні кроки, які ми повинні виконати:

1. Визначимо в яку піраміду (H_{low} або H_{high}) потрібно додати новий елемент. Якщо x_i менше ніж найбільший елемент з H_{low} (тобто новий елемент буде розташовуватись в меншій поточній половині), то додаємо його у цю піраміду. В іншому випадку додаємо елемент в піраміду H_{high}.
2. В кожний момент часу, тобто на кожній ітерації роботи алгоритму, повинен зберігатись наступний інваріант: **кількість елементів в піраміді H_{low} не повинна відрізнятись від кількості елементів в H_{high} не більше ніж на одиницю**. Під час виконання попереднього етапу цей інваріант може порушитись. Тому тепер необхідно відновити даний інваріант: якщо у піраміді H_{low} елементів більше на 2 за H_{high}, то визначаємо найбільший елемент з H_{low} і вставляємо його у H_{high}; якщо кількість елементів у H_{high} більше на 2 за H_{low}, то визначаємо найменший елемент з H_{high} і вставляємо його у H_{low}. Зрозуміло, що після кожної вставки нового елементу в піраміду необхідно перевіряти властивість піраміди: для H_{low} - властивість незростаючої піраміди, для H_{high} - властивість неспадної піраміди.
3. Визначити медіану для поточного масиву $A' = [x_1, ..., x_i]$:
    - Якщо кількість елементів у A' парна, то після збереження інваріанту у пункті 2, кількість елементів у пірамідах H_{low} та H_{high} буде рівною. Тому одна медіана буде найбільшим елементом H_{low}, а інша медіана — найменшим елементом H_{high}.
    - Якщо кількість елементів у A' непарна, то єдина медіана буде знаходитись у тій піраміді, в якій кількість елементів буде більше (на одиницю) за кількість в іншій. Тому, якщо кількість елементів у H_{low} більше за H_{high}, то медіана — це найбільший елемент з H_{low}. Інакше медіана — найменший елемент з H_{high}.

Наведений алгоритм використовує процедури extract_max незростаючої піраміди H_{low} та extract_min неспадної піраміди H_{high}, які виконуються за час $O(\log N)$, де N — розмір піраміди. Тому на кожній ітерації №i для поточного масиву $A' = [x_1, ..., x_i]$ час роботи наведеної процедури становитиме $O(\log(i))$.

Формат вхідних/вихідних даних

Розроблена програма повинна зчитувати вхідні дані з файлу заданого формату та записувати дані у файл заданого формату.

Вхідний файл представляє собою текстовий файл із N+1 рядків, де N — це розмірність вхідного масиву A. Першим записом є число — кількість елементів в масиві; наступні N записів містять елементи вхідного масиву.

Вихідний файл представляє також текстовий файл із N рядків, де кожен рядок i містить медіани

для вхідного підмасиву [x_1, ..., x_i]. Якщо медіана одна, то в рядку буде одне число; якщо медіани дві, то вони записується через пробіл.

До документу завдання також додаються приклади вхідних і вихідних файлів різної розмірності. Нижче наведені приклади вхідного та вихідного файлу для N = 10.

Вхідний файл	Вихідний файл
10 6 10 7 1 4 8 3 9 5 2	6 6 10 7 6 7 6 6 7 6 6 7 6 5 6

Вимоги до програмного забезпечення

- Розробляти програму можна на одній з наступних мов програмування: **C/C++** (версія C++11)**, C#** (версія C# 5.0)**, Java** (версія Java SE 8)**, Python** (версія 2.7).
- Програма повинна розміщуватись в окремому висхідному файлі, без використання додаткових нестандартних зовнішніх модулів.
- Не дозволяється використовувати будь-які нестандартні бібліотеки та розширення. Програма не повинна залежати від операційної системи. Якщо ви не впевнені відносно того, які бібліотеки відносяться до стандартних, зверніться до викладача.
- Не реалізуйте жодного інтерфейсу користувача (окрім командного рядку). Програма не повинна запитувати через пристрій вводу в користувача жодної додаткової інформації. Вашу програму будуть використовувати виключно у вигляді “чорного ящику”.
- **Назва висхідного файлу** вашої програми повинна задовольняти наступному формату: *НомерГрупи_ПрізвищеСтудента_НомерЗавдання.Розширення*, де *НомерГрупи* — це один з рядків is31, is32, is33; *ПрізвищеСтудента* — прізвище студента записане латинськими літерами; *НомерЗавдання* — двозначний номер завдання (01, 02, ...); *Розширення* — розширення файлу, відповідно до мови програмування (.c, .cpp, .cs, .java, .py). Приклад назви висхідного файлу: *is31_ivanenko_03.cs*.
- Розроблена програма повинна зчитувати з командного рядку назву вхідного файлу та записувати результат у вихідний файл. При запуску першим і **єдиним аргументом командного рядку** повинна бути назва вхідного файлу (наприклад, *input_10.txt*). Назва

вихідного файлу повинна складатись із назви файлу самої програми разом із суфіксом “*_output*” і мати розширення .txt. Приклад назви вихідного файлу: *is31_ivanenko_03_output.txt*.

Нараховані бали

За успішне виконання даного завдання нараховується **5 балів**.

Також за ранню здачу нараховується один додатковий бал. Але, якщо на момент ранньої здачі програма не працює коректно, то додаткові бали не нараховуються.

Можна отримати ще один додатковий бал за ефективне алгоритмічне рішення: коли алгоритм є компактним та елегантним. Цей бал враховується тільки у випадку вчасної здачі програми (коли ще не нараховуються штрафні бали).

Термін здачі

Практичне завдання №3 необхідно здати до **23:59 22 березня 2015 року**.

Якщо ви завантажили робочу версію програми до 23:59 15 березня, то це вважається ранньою здачею, за яку нараховуються додаткові бали.

Якщо ви не встигли надати вашу програму до вказаного часу, то маєте додатковий час до 23:59 5 квітня, але в такому разі буде нараховано тільки 50% балів.

Як завантажувати відповідь на завдання

Розроблена програма у вигляді висхідного файлу із відповідною назвою завантажується в систему Moodle (http://moodle.asu.ntu-kpi.kiev.ua/) в курс “**Теорія алгоритмів (КН)**”. Файл програми необхідно завантажити у розділ “**Практичне завдання №4**”. Дозволяється робити декілька спроб завантаження. У випадку кількох завантажень оцінюватись буде тільки останній завантажений варіант.

Рекомендована література

Конспект лекцій з курсу “Теорія алгоритмів”: **Лекція 07 — Піраміди**