Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

**Лабораторна робота *№5***

із дисципліни ***«Теорія алгоритмів»***

Тема: ***«Деревовидні структури даних»***

**Виконали:**

Студенти групи ІА-34

Янович Марія,

Ковальчук Станіслав,

Ястремський Богдан,

Сухоручкін Гліб

**Перевірив:**

Степанов Андрій Сергійович

Київ — 2024

Теорія алгоритмів

Практичне завдання №6

“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 = [x1, ..., xN]. Припустимо, що елементи масиву поступають на вхід програми послідовно: в кожний момент часу розглядається новий елемент xi. Необхідно для кожного i (від 1 до N) визначити медіану підмасиву A' = [x1, ..., xi], 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 Hlow 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 Hhigh 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 xi. На поточний момент сумарна кількість елементів, які зберігаються в обох пірамідах, становить (i-1). Наступні кроки, які ми повинні виконати:

1. Визначимо, в яку піраміду (Hlow або Hhigh) потрібно додати новий елемент. Якщо xi менше ніж найбільший елемент з Hlow (тобто новий елемент буде розташовуватись в меншій поточній половині), то додаємо його у цю піраміду. В іншому випадку додаємо елемент в піраміду Hhigh.

2. В кожний момент часу, тобто на кожній ітерації роботи алгоритму, повинен зберігатись наступний інваріант: кількість елементів в піраміді Hlow не повинна відрізнятись від кількості елементів в Hhigh не більше ніж на одиницю. Під час виконання попереднього етапу цей інваріант може порушитись. Тому тепер необхідно відновити даний інваріант: якщо у піраміді Hlow елементів більше на 2 за Hhigh, то визначаємо найбільший елемент з Hlow і вставляємо його у Hhigh; якщо кількість елементів у Hhigh більше на 2 за Hlow, то визначаємо найменший елемент з Hhigh і вставляємо його у Hlow. Зрозуміло, що після кожної вставки нового елементу в піраміду необхідно перевіряти властивість піраміди: для Hlow - властивість незростаючої піраміди, для Hhigh - властивість неспадної піраміди.

3. Визначити медіану для поточного масиву A' = [x1, ..., xi]:

• Якщо кількість елементів у A' парна, то після збереження інваріанту у пункті 2, кількість елементів у пірамідах Hlow та Hhigh буде рівною. Тому одна медіана буде найбільшим елементом Hlow, а інша медіана — найменшим елементом Hhigh.

• 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 Hlow більше за Hhigh, то медіана — це найбільший елемент з Hlow. Інакше медіана — найменший елемент з Hhigh.

Наведений алгоритм використовує процедури extract\_max незростаючої піраміди Hlow та

extract\_min неспадної піраміди Hhigh, які виконуються за час O(logN), де N — розмір піраміди. Тому на кожній ітерації №i для поточного масиву A' = [x1, ..., xi] 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 [x1, ..., xi]. 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 |

**Псевдокод:**

Function median\_from\_stream(nums):

low = [] # Min heap for elements less than median

high = [] # Max heap for elements greater than median

medians = [] # List to store medians as they are computed

For each num in nums:

If low is empty or num <= -low[0]:

Push -num onto low (negate num for max heap behavior)

Else:

Push num onto high

# Balance the heaps

If len(low) > len(high) + 1:

Push -heapq.heappop(low) onto high

Else if len(high) > len(low):

Push -heapq.heappop(high) onto low

# Compute median based on heap sizes

If len(low) > len(high):

Append -low[0] to medians

Else if len(low) == len(high):

Append "-low[0] high[0]" to medians

Else:

Append high[0] to medians

Return medians

**Код:**

def median\_from\_stream(nums):  
 low, high = [], []  
 medians = []  
  
 for num in nums:  
 if len(low) == 0 or num <= -low[0]:  
 heapq.heappush(low, -num)  
 else:  
 heapq.heappush(high, num)  
  
 if len(low) > len(high) + 1:  
 heapq.heappush(high, -heapq.heappop(low))  
 elif len(high) > len(low):  
 heapq.heappush(low, -heapq.heappop(high))  
  
 if len(low) > len(high):  
 medians.append(-low[0])  
 elif len(low) == len(high):  
 medians.append(f"{-low[0]} {high[0]}")  
 else:  
 medians.append(high[0])  
  
 return medians

**Висновки:** на даній лабораторній роботі ми реалізували задачу визначення послідовності медіан для заданого вхідного масиву, складність якої — O(log(i)). Також ми розглянули і навчились використовувати структуру даних — піраміди.