DATA STRUCTURES AND PYTHON

Priority Queue, Heap and Python heapq module

Abstract

A quick guide to understand abstract data structures – Priority Queue and Heap
Example of how python's heapq module can be a handy tool to implement Priority Queues

Hiral Amodia
amodia.hiral@gmail.com

heapq module

Article #23 – I was recently working on a coding problem statement from HackerRank. To complete this problem statement, it took me 3 attempts along with some help from the others who had submitted a solution to this problem statement. Due to my insufficient knowledge of three important concepts – **1). Priority queues**, **2). Heaps** and **3). Python's heapq module**, it took me much longer to implement an efficient solution to the problem statement.

I then worked towards a proper understanding these three important concepts through the help available on the Internet and by some practice problems. In this article I am sharing all the details with hope that it would further help others.

There are many experts who have done extensive work in technology. I benefitted from their work and the resources shared by them. I express my gratitude to them and I am sharing their links in the references section.

The Jesse and Cookies problem on HackerRank:

Link to this problem statement is available in references section

HackerRank PRACTICE CERTIFICATION COMPETE JOBS LEADERBOARD Search

Practice > Data Structures > Heap > Jesse and Cookies

Jesse and Cookies ★

Problem | Submissions | Leaderboard | Discussions | Editorial

Jesse loves cookies and wants the sweetness of some cookies to be greater than value k. To do this, two cookies with the least sweetness are repeatedly mixed. This creates a special combined cookie with:

sweetness $= (1\times$ Least sweet cookie $+ 2\times$ 2nd least sweet cookie).

This occurs until all the cookies have a sweetness $\geq k$.

Given the sweetness of a number of cookies, determine the minimum number of operations required. If it is not possible, return -1.

Example

$k = 9$

$A = [2, 7, 3, 6, 4, 6]$

The smallest values are $2, 3$.

Remove them then return $2 + 2 \times 3 = 8$ to the array. Now $A = [8, 7, 6, 4, 6]$.

Remove $4, 6$ and return $4 + 6 \times 2 = 16$ to the array. Now $A = [16, 8, 7, 6]$.

Remove $6, 7$, return $6 + 2 \times 7 = 20$ and $A = [20, 16, 8, 7]$.

Finally, remove $8, 7$ and return $7 + 2 \times 8 = 23$ to A. Now $A = [23, 20, 16]$.

All values are $\geq k = 9$ so the process stops after 4 iterations. Return 4.

Function Description

Complete the cookies function in the editor below.

cookies has the following parameters:

- int k: the threshold value
- int A[n]: an array of sweetness values

Returns

- int: the number of iterations required or -1

My first attempt – Correct but inefficient

The first thought that came to my mind was to solve this using recursion. So, I wrote the following solution:

```
def recur(k, A, ctr):
  A. sort()
  if A[0] >= k:
    return ctr
  else:
    num_1 = A[0]
    num_2 = A[1]
    temp_arr = []
    temp_arr.insert(0,(num_1 * 1) + (num_2 * 2))

    new_arr = temp_arr + A[2:]
    ctr += 1
    return(recur(k, new_arr, ctr))

def cookies(k, A):
    A.sort()
    if sum(A) < k:
      return -1
    else:
      x = recur(k, A, 0)
    return x
```

With the above solution, I hit the '*RecursionError*' exception in test case:

RecursionError: maximum recursion depth exceeded in comparison

My second attempt – Resolves the previous exception, but still inefficient

In the second attempt I removed the recursion and wrote a solution based on function iteration so that I can circumvent the recursion limit

```
def cookies(k, A):
  len_A = len(A)
  ctr = 0
  A.sort()
  if sum(A) < k:
    return -1
  else:
    while((A[0]< k) and (ctr <= len_A)):
      A.sort()
      num_1 = A[0]
      num_2 = A[1]
      temp_arr = []
      temp_arr.insert(0,(num_1 * 1) + (num_2 * 2))
      new_arr = temp_arr + A[2:]
      ctr += 1
      A = new_arr
      A.sort()
    else:
      if (ctr > len_A):
        return -1
      else:
        return ctr
```

With this solution, the recursion limit test case passed, however I got a lot of test case failures for time limit exceeding the expected time limit. Thus, my solution did become better but was still inefficient.

I spent some more time and tried fixing in a few ways, but I could not overcome the inefficiency. That is, when I took to the hints from the solutions submitted by others. This opened a great opportunity for me to learn 1). Priority Queues, 2). Heaps and 3). Python heapq module

My third attempt – using Python's heapq module

```
from heapq import heapify, heappush, heappop

def cookies(k, A):
    thecookies = list(A)
    minimum_value = k
    heapify(thecookies)
    count = 0
    while thecookies[0] < minimum_value:
        if len(thecookies) < 2:
            return -1
        heappush(thecookies, heappop(thecookies) + 2 * heappop(thecookies))
        count += 1
    return count
```

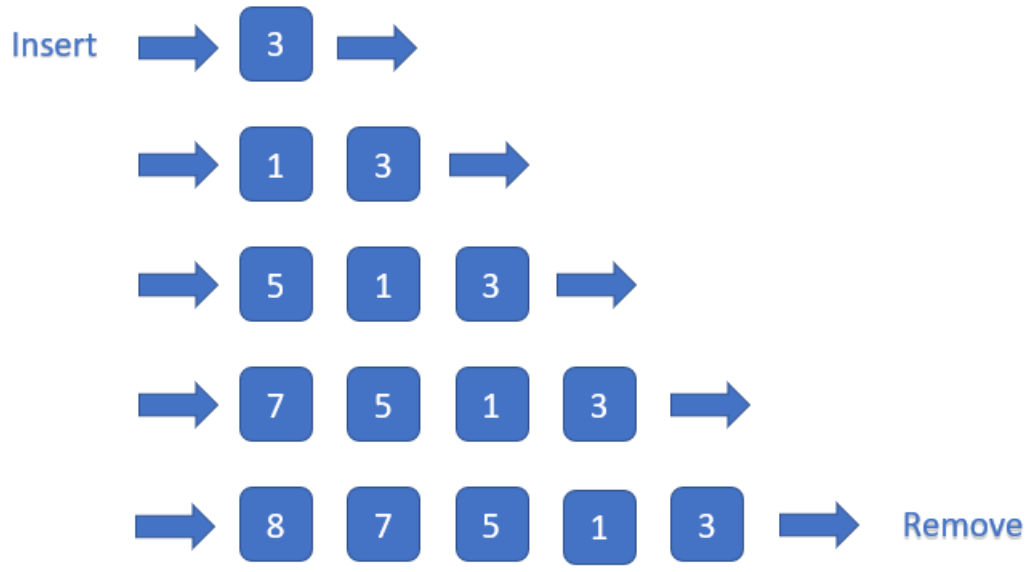
Note: This solution is inspired by solution provided here: https://programs.programmingoneonone.com/2021/05/hackerrank-jesse-and-cookies-solution.html

This problem statement requires us to remove the least two values from the array in every iteration. This indicates that this can be best implemented as by 'Priority Queue' data structure. Python has a module named 'heapq' which is an efficient implementation of the heap queue algorithm also known as priority queue algorithm.

So, lets spend some time in understanding what are Priority Queues, Heaps and then the Python heapq package.

What are priority queues:

A Priority Queue is an abstract data type like a queue with one difference that every element in a priority queue has an associated priority and the elements are removed as per the associated priority.



Queue Data structure

Priority queues can either be 'Ascending Priority Queue' where the smallest value holds the highest priority or 'Descending Priority Queue' where the largest value holds the highest priority.

Ascending Priority Queue

Descending Priority Queue

Note: More information about Priority Queue can be read through the links shared in References section.

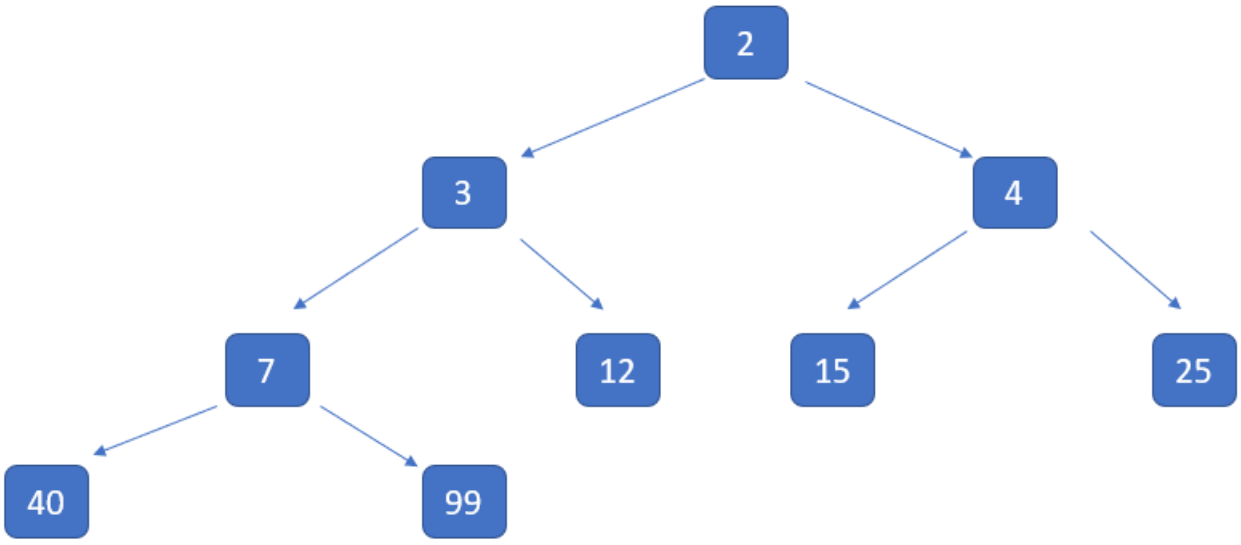
Priority queues can be implemented in various ways, one of it being 'heap'. Let's understand what are heaps.

What is a heap

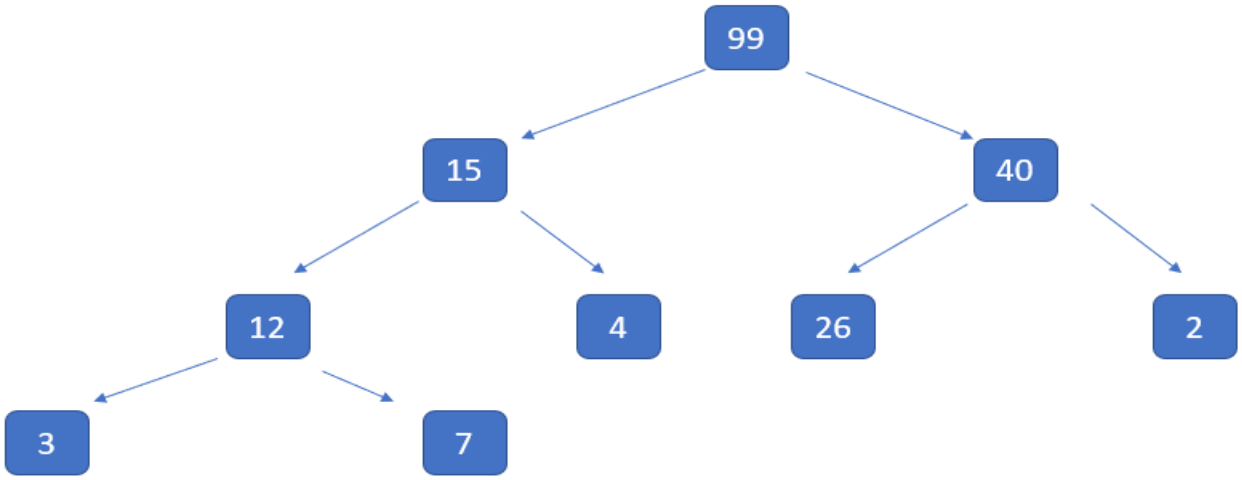
Priority queues can be implemented very efficiently using another abstract data type called 'heap'.

Heap is a specialized tree data structure that always satisfies 'heap property':

- In an ascending heap (min heap), for any given node C, if P is a parent node of C, then the value of P is lesser than or equal to the value of C
- In an descending heap (max heap), for any given node C, if P is a parent node of C, then the value of P is greater than or equal to the value of C



Ascending Heap



Descending Heap

Note: More information about Heap can be read through the links shared in References section.

What is heapq

Heapq is a python module that provides the implementation of Heaps or Priority Queues and is a very handy tool for all the problem statements that follow Priority Queue characteristic like the Jesse and Cookes problem. Link to the documentation of this package are in the references section

A PDF Version of this article can also be downloaded from my GitHub: Link

References

Priority Queues:
https://en.wikipedia.org/wiki/Priority_queue
https://www.programiz.com/dsa/priority-queue
https://www.javatpoint.com/ds-priority-queue
https://www.geeksforgeeks.org/priority-queue-set-1-introduction/

Heap
https://www.geeksforgeeks.org/heap-data-structure/
https://en.wikipedia.org/wiki/Heap_(data_structure)
https://www.tutorialspoint.com/data_structures_algorithms/heap_data_structure.htm

Python heapq package
https://docs.python.org/3/library/heapq.html

The Jesse and Cookies problem on HackerRank:
Link: https://www.hackerrank.com/challenges/jesse-and-cookies/problem

About Me:

My Name is **Hiral Amodia.** I am based in Bangalore, India. I am a Software Engineer working in Indian IT Industry for over 17 years now. Currently, I am employed as a Software Engineering Manager at a leading Indian IT services company. I am passionate about learning new technologies and concepts. I strongly believe that teaching is the best way of learning and that caring is the true way of sharing. With this philosophy in mind, I keep on writing articles on technology, concepts, etc. that I learn.

Feel free to buzz me on my below coordinates if you want to share any feedback or improvement areas that you encounter in my articles.

My Coordinates are as below:

Email: amodia.hiral@gmail.com

LinkedIn: https://www.linkedin.com/in/hiral-amodia/

GitHub: **https://github.com/amodiahs**