



You

Help me create a class that can store a list of integers. The data structure should be such that I can iterate from the data structure with the smallest total sum easily, as well as add and sort . Think of it as a min heap where it is minimise the total sum of the list



ChatGPT

You can create a class that behaves like a min heap where the key for comparison is the total sum of the list. Here's how you can implement such a class in Python:

```
class MinSumListHeap:

    def __init__(self):
        self.heap = []

    def add_list(self, new_list):
        self.heap.append(new_list)
        self._heapify_up(len(self.heap) - 1)

    def pop_smallest(self):
        if not self.heap:
            return None

        smallest_list = self.heap[0]
        if len(smallest_list) > 1:
            self.heap[0] = self.heap.pop()
            self._heapify_down(0)
        else:
            self.heap.pop(0)

        return smallest_list

    def _heapify_up(self, index):
```

```
while index > 0:

    parent_index = (index - 1) // 2

    if self._get_sum(self.heap[index]) < self._get_sum(self.heap[parent_index]):

        self.heap[index], self.heap[parent_index] = self.heap[parent_index], self.heap[index]

        index = parent_index

    else:

        break
```

```
def _heapify_down(self, index):

    while True:

        left_child_index = 2 * index + 1

        right_child_index = 2 * index + 2

        smallest = index

        if (left_child_index < len(self.heap) and

            self._get_sum(self.heap[left_child_index]) < self._get_sum(self.heap[smallest])):

            smallest = left_child_index

        if (right_child_index < len(self.heap) and

            self._get_sum(self.heap[right_child_index]) < self._get_sum(self.heap[smallest])):

            smallest = right_child_index

        if smallest != index:

            self.heap[index], self.heap[smallest] = self.heap[smallest], self.heap[index]

            index = smallest

        else:

            break
```

```
def _get_sum(self, lst):
```

```
return sum(lst)
```

Example usage:

```
min_heap_list = MinSumListHeap()
```

```
min_heap_list.add_list([1, 3, 5])
```

```
min_heap_list.add_list([2, 4, 6])
```

```
min_heap_list.add_list([0, 7, 8])
```

```
print("Pop smallest:")
```

```
print(min_heap_list.pop_smallest()) # [0, 7, 8]
```

```
print(min_heap_list.pop_smallest()) # [1, 3, 5]
```

```
print(min_heap_list.pop_smallest()) # [2, 4, 6]
```

```
print(min_heap_list.pop_smallest()) # None, as the heap is empty
```

This class `MinSumListHeap` allows you to add lists and pop the one with the smallest total sum iteratively. It implements the necessary methods to maintain the min heap property where the key for comparison is the total sum of the list.



You

I dont want a pop method, hepl me generate a iterate that lets me iterate through from min heap downwards



ChatGPT

To create a data structure that behaves like a min-heap where the iteration goes from the smallest total sum downwards, you can implement a custom class that internally maintains the list of integers and sorts it based on the sum each time a new element is added. You can implement the class as follows:

```
class MinSumHeap:
```

```
    def __init__(self):
```

```
self.data = []
```

```
def add(self, num_list):  
    self.data.extend(num_list)  
    self.data.sort()
```


```
def __iter__(self):  
    return iter(self.data)
```

In this implementation:

- The `MinSumHeap` class has an internal list `self.data` to store the integers.
- The `add` method adds a list of integers to the internal data and sorts the data afterwards.
- The `__iter__` method returns an iterator over the internal data, which is sorted in ascending order.

Here's how you can use this class:

python

 Copy code

```
# Create an instance of MinSumHeap  
heap = MinSumHeap()  
  
# Add lists of integers  
heap.add([3, 1, 4])  
heap.add([2, 7, 5])  
  
# Iterate through the heap from smallest total sum downwards  
for num in heap:  
    print(num)
```