rock ★ 5276 Last Edit: August 25, 2019 6:09 PM 3.1K VIEWS

26

Earlier sticks will be counted again. Therefore, alwasy use current shortest two sticks till only one remains.

**Java**

```java
public int connectSticks(int[] sticks) {
    PriorityQueue<Integer> pq = new PriorityQueue<>();
    for (int s : sticks) {
        pq.offer(s);
    }
    int sum = 0;
    while (pq.size() > 1) {
        int two = pq.poll() + pq.poll();
        sum += two;
        pq.offer(two);
    }
    return sum;
}
```

**Python 3 heapq**

```python
class Solution:
    def connectSticks(self, sticks: List[int]) -> int:
        h = []
        for s in sticks:
            heapq.heappush(h, s)
        sum = 0
        while len(h) > 1:
            two = heapq.heappop(h) + heapq.heappop(h)
            sum += two
            heapq.heappush(h, two)
        return sum
```

Note: if the input list `sticks` is allowed to modify, we can just use `heapify(sticks)` to make the code shorter.

**Analysis:**

Time: O(nlogn), space: O(n), where n = sticks.length.



Type comment here... (Markdown is supported)

Post

kushuyanhan ★ 5 December 8, 2019 5:30 AM

why does greedy guarantee the minimum?

1 Show 2 replies Reply

jborgaonkar ★ 21 January 13, 2020 8:25 AM

what is the time complexity for this solution

0 Show 1 reply Reply

maverick009 ★ 174 August 31, 2019 3:06 AM

Amazing solution

0 Reply