Python Competitive Programming Cheatsheet

Soumitra Das

December 4, 2022

Modules

heapq

A data structure to maintain smallest element first. Note that it do not keep things sorted.

```
import heapq
x = [] #should contain data structure of same type
heapq.heappush(x, [4, 'd']) #complexity log(n)
heapq.heappush(x, [4, 'b'])
heapq.heappush(x, [7, 'c'])
print(x) ## [[4, 'b'], [4, 'd'], [7, 'c']]
heapq.heappop(x) #complexity log(n)
print(x) ## [[4, 'd'], [7, 'c']]
heapq.heappushpop(x, [1, 'd']) #first push  then pop
print(x) ## [[4, 'd'], [7, 'c']]
heapq.heapreplace(x, [1, 'e']) #first pop then push
print(x) ## [[1, 'e'], [7, 'c']]
y = [2, 5, 1]
heapq.heapify(y) # creates heap, just brings the
    smallest element first, O(n) complexity
print(y) ## [1, 5, 2]
heapq.heappush(y, 4)
print(y) ## [1, 4, 2, 5]
heapq.heappop(y)
print(y) ## [2, 4, 5]
print(heapq.nlargest(2, x)) ## [[7, 'c'], [1, 'e']] ##
     returns n largest element list
    ### if returns t element, complexity O(t*log(n))
print(heapq.nsmallest(2, y)) ## [2, 4] ## returns n
    smallest element list
```

Algorithms