CS 180A : Review 1

Week 5 : Heaps and Algorithms

Zexi Huang

Outlines:

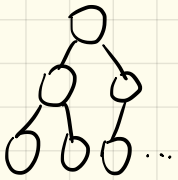
1. Heaps Review
   - Binary Heap, Leftist Heap, Skew Heap and Binomial Queues
   - HW2: P5 ~8
2. Algorithm Design
   - Top K Frequent Elements

Heaps

Motivation: Fast DeleteMin and Insert methods ⇒ Priority Queue

Heap-order Property: Parent nodes are smaller than their children.

① Binary Heap

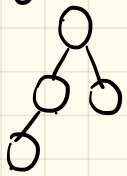


DeleteMin O(logn)

Insert O(logn) / O(1)

BuildHeap O(n)

② Leftist Heap



Null Path Length: left ≥ right

DeleteMin → Merge O(logn)

Insert →

③ Skew Heap:

Always swap children

Amortized $O(\log n) \Rightarrow \frac{O(m \log n)}{m}$

④ Binomial Queue:

Forest of heaps

Merge O(logn)

Insert O(logn) / O(1)

DeleteMin O(logn)

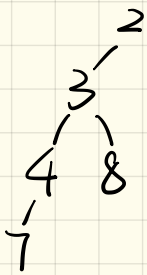
Heaps: Examples

P5 (Binary): 89%; P6 (Leftist): 85%; P7 (Skew): 82%; P8 (Binomial): 95%

H_1: [3, 4, 8, 7, 2]; H_2: [9, 1, 5]

(1)

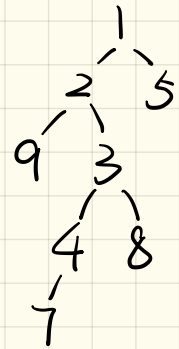
H_1:



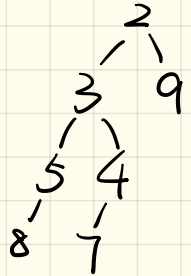
H_2:

(2)

H_1+H_2:



(3)



⇒1

Algorithm Design: Top K Frequent Elements

Given an array A, an integer K, find K most frequent elements.

Ex: A = [0, 3, 2, 1, 3, 2, 5], K = 2

① Brute-Force Algorithm $O(Kn + n\log n)$

```
sort A                                   O(nlogn)
for k = 1 : K :                          O(K)
    for i = 1 : n :                      O(n)
        find most frequent element a
    for i = 1 : n :                      O(n)
        remove A[i] = a.
```

② Store the frequencies: $O(n\log n)$

```
freq : element → frequency
for i = 1 : n :                          O(n)
    freq[A[i]] += 1                      O(1)
sort freq based on frequency             O(nlogn)
return top K                             O(K)
```

③ Heap $O(n + K\log n)$

```
freq : element → frequency
for i = 1 : n :                          O(n)
    freq[A[i]] += 1                      O(1)
build max heap with freq.                O(n)
deleteMax K times                        O(Klogn)
```