CS 180A : Discussion

Week 3. Hashing and Heaps

Zexi Huang

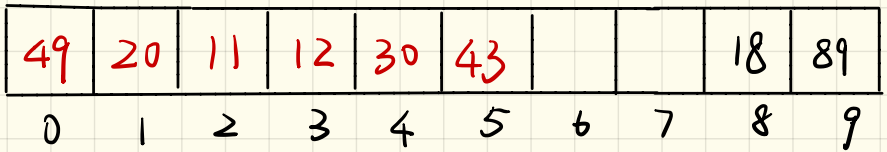
Outlines:

1. Hashing
   - Probing and clustering
2. Heaps
   - Bottom-up Construction
3. Algorithm Design
   - Two Sum

Hashing : Probing and clustering

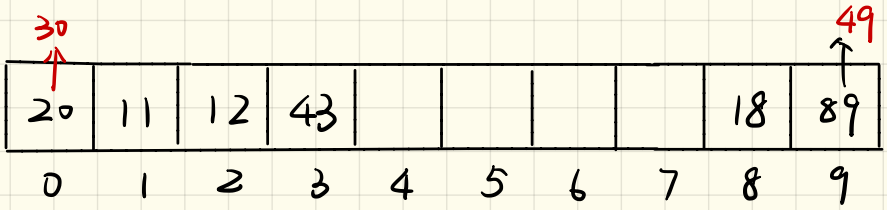
elements : [89, 18, 49, 20, 11, 12, 30, 43] hash function : $f(x) = x \bmod 10$

Linear Probing



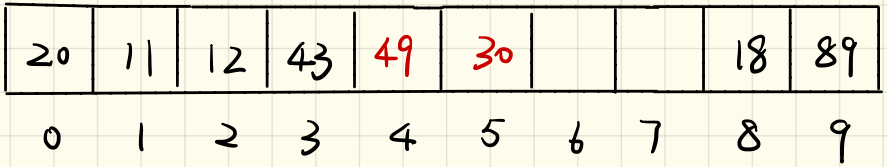
Primary Clustering

Separate Chaining



[5, 1, 2, 8, 7, 3, 4, 6, 9, 10] $f(x)+5 \rightarrow f(x)+1 \rightarrow f(x)+2$

Random Probing

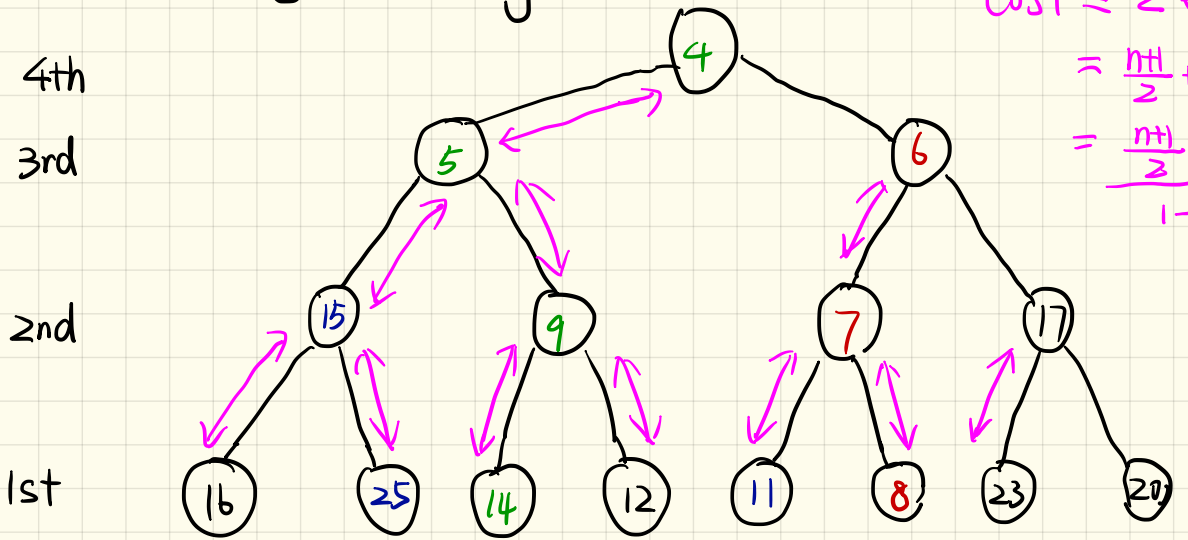


Heaps: Bottom-up Construction

Heap: Binary tree where parent nodes are smaller than their children.

Elements: [16, 15, 4, 12, 6, 7, 23, 20, 25, 9, 11, 17, 5, 8, 14]

Insert one by one: $O(n \log n) \Longrightarrow O(n)$



Cost $\leq \sum$ edges

$= \frac{n+1}{2} + \frac{n+1}{4} + \cdots + 2$

$= \frac{\frac{n+1}{2} \div 1}{1 - \frac{1}{2}} = n - 2$

$= O(n)$

$\frac{n+1}{4}$

$\frac{n+1}{2}$

Algorithm Design : Two Sum

Given an array A, a target value s. find two numbers in A that add up to S.

① Brute Force Algorithm: $O(n^2)$

```
for all pairs (A[i], A[j])          O(n^2)
    return A[i], A[j] if A[i]+A[j]=S  O(1)
```

② Sorting and Binary Search $O(n \log n)$

```
sort A                              O(nlogn)
for all A[i]=                       O(n)
    binary search for S-A[i]        O(logn)
    return if found
```

③ Hashing $O(n)$

```
for all A[i]:                       O(n)
    put A[i] in a hash table        O(1)
for all A[i]                        O(n)
    find S-A[i] in the hash table   O(1)
    return if found
```

$A = [3, 2, 7, 5, 11, 16]$, $S = 9$

3 → find 6 ; 2 → find 7

For A[i]. find S-A[i]

$A' = [2, 3, 5, 7, 11, 16]$, $S = 9$

Cost of finding:

Binary search: $O(\log n)$

Hashing : $O(1)$