

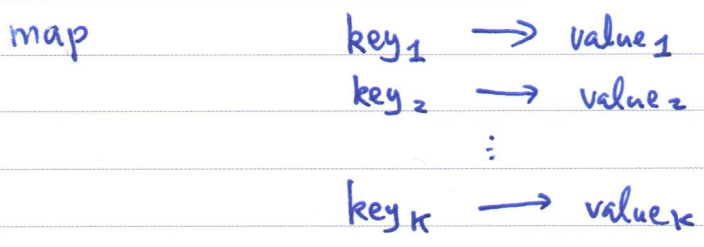
Subject:

* priority queue

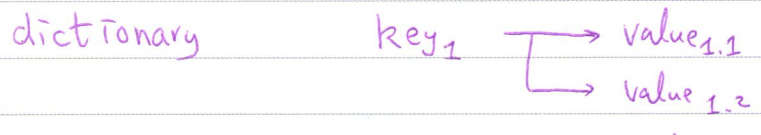
want A. insert (key, value)
A. retrieve Max()
fast

* what if

want A. insert (key, value)
A. retrieve (key)
fast?



unordered/
ordered

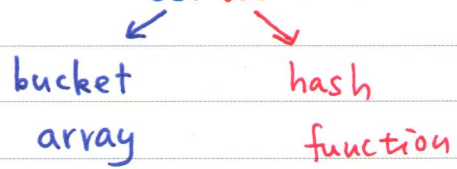


		search	insert	retrieve
* sparse array	w/ key		O(1)	O(n)
	(on dense)		O(n)	O(log n)
linked list	w/ key		O(1)	O(n)
	ordered key		O(n)	O(n)

* if keys are integers
O(1) insert O(1) retrieve
by dense array (that wastes space)

* how about using a "simple" (almost O(1)) function
h: key → integer

A.insert(key, value) \Rightarrow $A[h(\text{key})] = \text{value}$
 $v = A.\text{retrieve}(\text{key}) \Rightarrow v = A[h(\text{key})]$



* if {keys} \longleftrightarrow {0, 1, 2, ..., k-1}

one-to-one

"perfect hashing"

* what if not?
 e.g. #keys > k



\Rightarrow two keys of the same $h(\cdot)$
 \Rightarrow collision!

* e.g. $h(\text{str}) = \text{str}[0] - 'a'$; 26 possibilities
 $h(\text{"act"}) = h(\text{"apple"})$

$h(\text{str}) = \sum_i (\text{str}[i] - 'a')$ 2^{32} ? possibilities
 not uniform

$h(\text{str}) = \sum_i (\text{str}[i] - 'a') \% k$ k possibilities
 "stop" "pots" "spot"
 \Rightarrow easy collision

* how to hash

$h(\text{key}) = \text{compress}(\text{Hashcode}(\text{key}))$

