Priority Queues

Chapter 6

Priority Queues

- Priority Queues are an example of an application of heaps
- A priority queue is an Abstract Data Type for maintaining a set of elements, each with an associated key

- What's the difference between an ADT and a data structure?

Priority Queues

- Max-priority queue supports dynamic set operations:
  - INSERT(S, x): inserts element x into set S.
  - MAXIMUM(S): returns element of S with largest key.
  - EXTRACT-MAX(S): removes and returns element S with largest key.
  - INCREASE-KEY(S, x, k): increases value of element x's key to k. Assume k >= x's current key value.

HEAP-MAXIMUM(A)

HEAP-MAXIMUM(A)

return $A[1]$

Time: $\Theta(1)$.

HEAP-EXTRACT-MAX

Heap_Extract_Max(A) // A: Array

1	`if A.heap_size < 1`
2	`  error "underflow"`
3	`max = A[1]`
4	`A[1] = A[A.heap_size]`
5	`A.heap_size = A.heap_size -1`
6	`Max_Heapify(A,1)`
7	`return max`

Example

- 15 6 4 8 5 3 1 2 7

Heap_Increase_Key, p 164

Heap_Increase_Key(A, i, key) // A: Array; i,key: ints

1	`if key < A[i]`
2	`  error "new key is smaller than current key"`
3	`A[i] = key`
4	`while i > 1 and A[Parent(i)] < A[i]`
5	`  swap (A[i], A[Parent(i)] )`
6	`  i = Parent(i)`

Why?

Example

- Increase key of node 6 in previous example to 20

MAX-HEAP-INSERT

- Given a key k to insert into the heap:
  - Insert a new node in the very last position in the tree with the key -infinity.
  - Increase the -infinity key to k using the HEAP-INCREASE-KEY procedure.

Max_Heap_Insert(A, key) // A:array; key:int

1	**A.heap_size = A.heap_size + 1**
2	**A[A.heap_size] = - infinity**
3	**Heap_Increase_Key(A, A.heap_size, key)**

Example

- Insert 12 into the above heap.