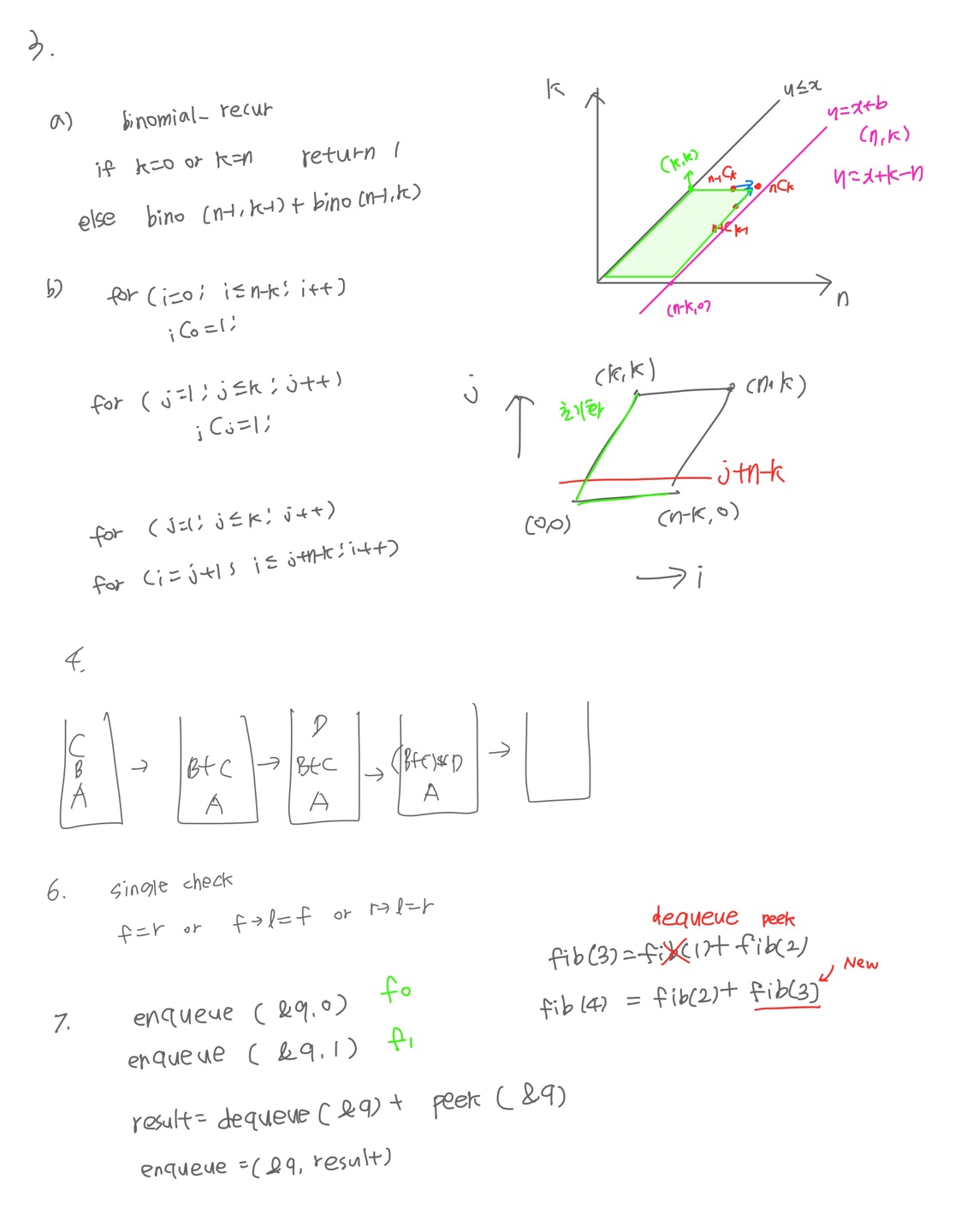
**[Data Structure]**

**Midterm Exam Review**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**Lecture 8: Priority Queue**

**Priority queue**

– A queue that stores items with priority

– The data with the higher priority is output first, not the FIFO order.

**Operation**

– Create () :: = Creates a priority queue.

– Init (q) :: = Initializes the priority queue q.

– is\_empty (q) :: = Checks if the priority queue q is empty.

– is\_full (q) :: = Checks if the priority queue q is full.

– insert (q, x) :: = Add an element x to the priority queue q.

– delete (q) :: = Removes the highest priority element from the priority queue and returns this element.

– find (q) :: = Returns the highest priority element without deleting it.

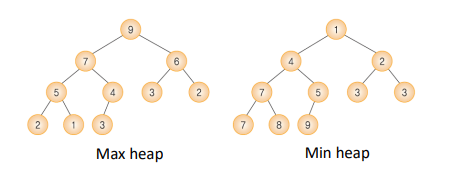
**Types**

– Minimum priority queue : Min heap

key (parent node) ≤key (child node)

– Maximum priority queue : Max heap

key (parent node) ≥key (child node)

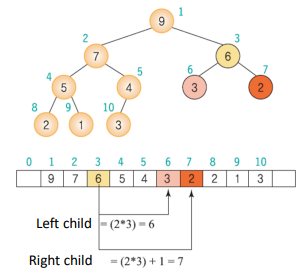


**TimeComplexity**

**테이블이(가) 표시된 사진

자동 생성된 설명**

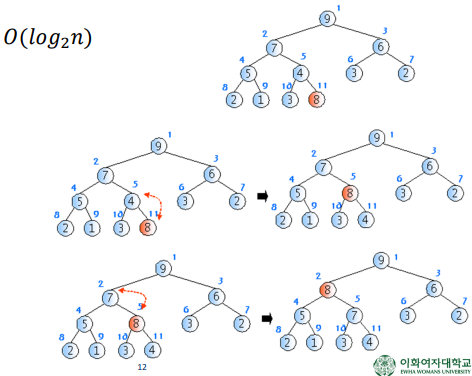
**Heap Implementation using arrays**



– Parent node of node i: i/2

– Left child node of node i: 2i

– Right child node of node i: 2݅ + 1

**Insertion in Max Heap**

1. The new node is inserted next to the last node in the heap

2. The nodes in the path from the inserted node to the root are compared and exchanged to satisfy the heap property.

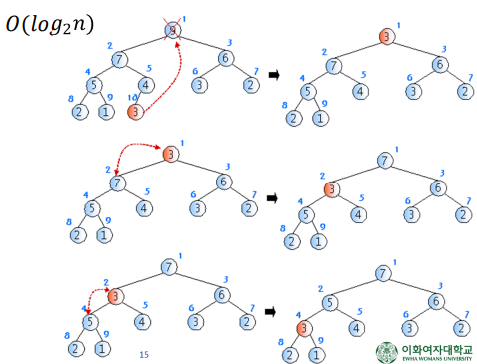
**Deletion in Max Heap**

**:** Deleting the node with the largest key value

1. Delete the root node (top most)

2. Move the last node to the root node.

3. The nodes in the path from the root to the leaf nodes are compared and exchanged to satisfy the heap property.

****

**Building Max Heap**

• Build-Max-Heap()

– When an array is given, put all elements into the heap first.

– Then, we can build a heap in a bottom-up manner by moving the element to meet the heap property.

– For the array of length n, all elements in n/2 + 1 … n already meet the heap property!

– Thus,

• Walk backwards through the array from n/2 to 1, moving the element on each node until it meets the heap property

. • The order of processing guarantees that the children of node i are heaps when i is processed

**Heap Application: Heap Sort**

**:** the sorting algorithm using the heap

• Procedure

1. Insert ݊ elements to be sorted in the max heap

2. Delete a root from the max heap and save it in an array ݊ times

• Time complexity: O(nlogn) , n :number of elements

– Step 1: O(nlogn)+ Step 2: O(nlogn)

Heap sort is useful, when you need a few of the largest values, not sorting the entire data.