6.006 Homework	**Problem set 3**	# 1 – Recurrences
Xola Ntumy		March 7, 2011
Collaborators: Kwadwo Nyarko		

(a) T(n) = O(n log n)

(b) T(n) = O (n to the power log base 4 of 3)

(c) T(n) = O(n to the power 2)

(d) The recurrence relation would be: T(n) = c * T(n / c) + log n
Solving this relation would give us:
T(n) = O (n log n)

(e) The running time ceases to be constant. It would now change based on the function n.

6.006 Homework	**Problem set 3**	# 2 – d-ary Heaps
Xola Ntumy		March 7, 2011
Collaborators: Kwadwo Nyarko		

(a) Parent(i):

return ((i - 1) / d) + 1

(b) Child(i, k):

return (i * d) + k + 1

(c) The height of the d-array heap would be O(log base d of n)

(d) The asymptotic running time of Heapify is O(d * log base d of n) since for every level it does d comparisons and it is of height log base d of n.

while the running time of Increase-key will be O(log base d of n) since it does one comparison per each of the log base d of n levels in the heap.

(e) Since the height is O(log base d of n):

If d = O(1), the resulting running time is O(log n)

If d = O(log n), the resulting running time becomes O(log base log n of n) which simplifies to O(log n / (log (log n)))

If d = O(n), the resulting running time becomes O(log base n of n) which simplifies to O(1)

(f) The running times for Heapify:

a) when d = O(1), would be O(log n)

b) when d = O(log n), would be O(log n * log n / (log (log n)))

c) when d = O(n), would be O(n)

The running times for Increase-key:

a) when d = O(1), would be O(log n)

b) when d = O(log n), would be O(log n / (log (log n)))

c) when d = O(n), would be O(1)

When d is increased the running times increase for Heapify but decrease for Increase-Key