HW02 for ECE 9343

Tongda XU, N18100977

October 3, 2018

1 Question 1: 3-divide maximum subarray

MAXFROMLEFT(A, p, r)

$max = -\infty$
for $i = p$ **to** r
    $max = Sum(A, p, i) > max?Sum(A, p, i) : max$
return $max > 0?max : 0$

MAXFROMRIGHT(A, p, r)

$max = -\infty$
for $i = r$ **downto** p
    $max = Sum(A, i, r) > max?Sum(A, i, r) : max$
return $max > 0?max : 0$

3-CROSS(A, p, s, t, r)

return $max(maxFromRight(A, p, s-1) + Sum(A, s, t-1) + maxFromLeft(A, t, r)$

3-MAXSUB(A, p, r)

$s = \lfloor (p+r)/3 \rfloor$
$t = \lfloor (p+r)2/3 \rfloor$
return $max($3-CROSS(A, p, s, t, r), 3-MAXSUB$(A, p, t-1)$, 3-MAXSUB$(A, s-1, r))$

The time complexity for 3-CROSS(A, p, s, t, r) is $\Theta(n)$, since $maxFromLeft, maxFromRight, Sum$ all take $\Theta(n)$ time, but all of them are $\frac{1}{3}n$ size. We have a iteration tree like:

$$T(h) = \begin{cases} \Theta(1) & h = 0 \\ 2T(\frac{2}{3}n) + \Theta(n) & h > 0 \end{cases}$$

Note that for leaf, the complexity is : $\Theta(n^{\frac{lg2}{lg3-lg2}})$
for branch, the complexity is : $\Theta(n^{\frac{lg4-lg3}{lg3-lg2}+1})$
These two are equal, so the overall complexity is: $\Theta(n^{log_{\frac{3}{2}}2})$

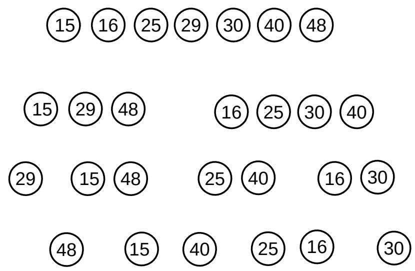

Figure 1: Merge Sort

2 Question 2: Intermediate Sequence

BUBBLE SORT(A)

$A = [11, 8, 7, 5, 3, 1]$
$\rightarrow [8, 11, 7, 5, 3, 1] \rightarrow [8, 7, 11, 5, 3, 1] \rightarrow [8, 7, 5, 11, 3, 1] \rightarrow [8, 7, 5, 3, 11, 1] \rightarrow [8, 7, 5, 3, 1, 11]$
$\rightarrow [7, 8, 5, 3, 1, 11] \rightarrow [7, 5, 8, 3, 1, 11] \rightarrow [7, 5, 3, 8, 1, 11] \rightarrow [7, 5, 3, 1, 8, 11]$
$\rightarrow [5, 7, 3, 1, 8, 11] \rightarrow [5, 3, 7, 1, 8, 11] \rightarrow [5, 3, 1, 7, 8, 11]$
$\rightarrow [3, 5, 1, 7, 8, 11] \rightarrow [3, 1, 5, 7, 8, 11]$
$\rightarrow [1, 3, 5, 7, 8, 11]$

INSERTION SORT(A)

$A = [11, 8, 7, 5, 3, 1]$
$\rightarrow [8, 11, 7, 5, 3, 1]$
$\rightarrow [8, 7, 11, 5, 3, 1] \rightarrow [7, 8, 11, 5, 3, 1]$
$\rightarrow [7, 8, 5, 11, 3, 1] \rightarrow [7, 5, 8, 11, 3, 1] \rightarrow [5, 7, 8, 11, 3, 1]$
$\rightarrow [5, 7, 8, 3, 11, 1] \rightarrow [5, 7, 3, 8, 11, 1] \rightarrow [5, 3, 7, 8, 11, 1] \rightarrow [3, 5, 7, 8, 11, 1]$
$\rightarrow [3, 5, 7, 8, 1, 11] \rightarrow [3, 5, 7, 1, 8, 11] \rightarrow [3, 5, 1, 7, 8, 11] \rightarrow [3, 1, 5, 7, 8, 11] \rightarrow [1, 3, 5, 7, 8, 11]$

3 Question 3: Illustrate Merge Sort

See Figure 1

MERGE SORT(A)

$15, 16, 25, 29, 30, 40, 48$
$15, 29, 48||16, 25, 30, 40$
$29||15, 48||25, 40||16, 30$
$-||48||15||40||25||16||30$

4 Question 4: CLRS Problem 2-1

4.1 a. show time complexity

$\Theta(T) = \frac{n}{k}\Theta(k^2) = \Theta(nk)$

4.2 b. show merge, c. show whole, max k

There should not be anything special about Merge function, just use the original interface and implement of Merge in CLRS pp 31.

$$T(n) = \begin{cases} n & n \le k \\ 2T(\frac{1}{2}n) + n & n > k \end{cases}$$

Regarding the iterative tree, it is easy to notice that: For branch (Merge), the complexity: $\Theta(nlg\frac{n}{k})$, For leaf (Insertion): $\Theta(nk)$, The sum is: $\Theta(nlg\frac{n}{k} + nk)$

Merge-sort-insertion(A, p, r, k)

```
if r − p + 1 ≤ k
      Insertion-Sort(A, p, r)
      return
elseif p < r
      q = ⌊(p + r)/2⌋
      Merge-Sort(A, p, q)
      Merge-Sort(A, q + 1, r)
      Merge (A, p, q, r)
      return
else return
```

Consider $\Theta(nlg\frac{n}{k} + nk) = \Theta(nlgn - nlgk + nk) = \Omega(nlgn)$, When $k = \Theta(lgn)$, it is OK.
But when $k = \omega(lgn), sum = \Theta(nk) = \omega(nlgn)$, so $k_{max} = \Theta(lgn)$

4.3 d. how to choose k

Note that in practice, we could have:
$T(n, k) = c_2(c_1nk + nlg(\frac{n}{k}))$
$\frac{\partial T(n,k)}{\partial k} = c_2(c_1n - \frac{n}{k})$
$c_1 = \frac{constant-of-insertion-sort}{constant-of-merge-sort}$, obviously<1 according to the question
$k \in [0, \infty], k = \frac{1}{c_1} = \frac{constant-of-merge-sort}{constant-of-insertion-sort}$ could minimize T(n,k)

5 Question 5: CLRS Problem 6.1-3

1. Since $x.Parent.key \ge x.key$, we have:
When $root.child.child \ne null, root.child.key \ge root.child.child.key$
When $root.child.child = null$, the conclusion naturally correct
2. Combined with $x.key \ge x.child.key$, using deduction, it is easy to conclude that $\forall h, root.child.key \ge root.(child)^h.key$

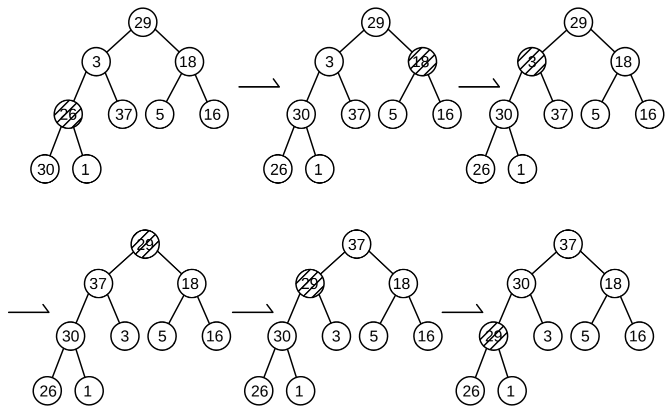

Figure 2: Build Heap

6 Question 6: CLRS Problem 6.2-6

1. Note that the height of a Heap is no more than $lg(n + \frac{1}{2}n - 1)$ in worst condition
2. Note that each round of $MAX - HEAPIFY$ takes constant time
4. Each time $MAX - HEAPIFY$ happen, the height of pointer $\leftarrow$ pointer- 1
5. We have:

$$T(h) = \begin{cases} c & h = 0 \\ T(h-1) + c & n > 0 \end{cases}$$

Solves: $T(h) = \Theta(h) = \Omega(lg\frac{3}{2}n - 1) = \Omega(lgn)$

7 Question 7: Draw Heap Sort Procedure

Build max heap, See Figure 2
heap sort, See Figure 3

8 Question 8: CLRS Problem 6-2

8.1 a. how to present

Within a part of array A[1, n]
get parent, Parent[i] = $\lfloor (i + d - 2)/d \rfloor$
get (k+1)th child, $k \in [0, d-1]$ Child[i,k] $= di + k$

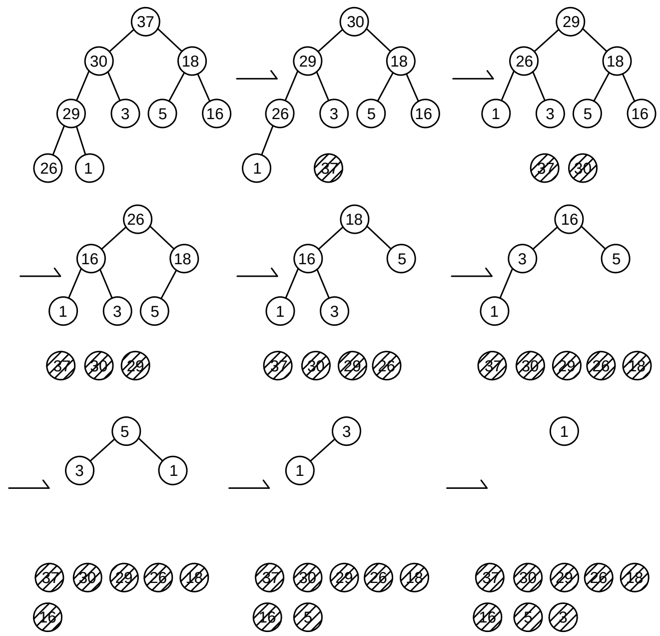

Figure 3: Heap Sort

8.2 b. height

$h = \lfloor log_d n \rfloor$

8.3 c. extract max

implement of max child value and index of i in $\Theta(d)$:

$\textsc{maxChild}(A, i, d)$

$max = -\infty$
$maxIndex = -1$
for $k = 0$ **to** $d - 1$
    if $di + k \leq n = A.size()$
        $max = A[di + k] > max?A[di + k] : max$
        $maxIndex = A[di + k] > max?[di + k] : maxIndex$
return $max, maxIndex$

implement of d-maxHeapify:

MAXHEAPIFY(A, i, d)

while $i \leq n = A.size()$
    if $A[i] \leq maxChild(A, i, d)[0]$
        $swap(A[i], maxChild(A, i, d)[1])$
        $i = maxChild(A, i, d)[1]$
return

$$T(h) = \begin{cases} \Theta(d) & h = 0 \\ T(h-1) + \Theta(d) & h > 0 \end{cases}$$

From iteration tree, it is easy to find that MaxHeapify from root for d-dimension heap cost $\Theta(dlog_d n)$

EXTRACTMAX(A, d)

$max = A[1]$
$swap(A[1], A[n])$
$erase(A[n])$
$maxHeapify(A, 1, d)$
return max

Extract is simple, also cost $\Theta(dlog_d n + Constant)$

9 Question 9: Visualize CLRS Problem 7-1

See Figure 4

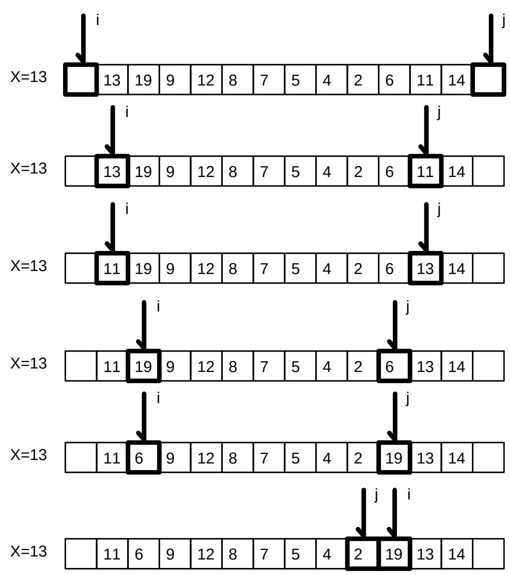

Figure 4: Hoare partition