

# HW02 for ECE 9343

Tongda XU, N18100977

October 2, 2018

## 1 Question 1: 3-divide maximum subarray

MAXFROMLEFT( $A, p, r$ )

```
1   $max = -\infty$ 
2  for  $i = p$  to  $r$ 
3       $max = Sum(A, p, i) > max ? Sum(A, p, i) : max$ 
4  return  $max$ 
```

MAXFROMRIGHT( $A, p, r$ )

```
1   $max = -\infty$ 
2  for  $i = r$  downto  $p$ 
3       $max = Sum(A, i, r) > max ? Sum(A, i, r) : max$ 
4  return  $max$ 
```

THREE-FOLD-MAXSUB( $A, p, r$ )

```
1   $s = \lfloor (p + r) / 3 \rfloor$ 
2   $t = \lfloor (p + r) 2 / 3 \rfloor$ 
3  if  $Sum(A, s, t - 1) > 0$ 
4      return  $max(maxFromLeft(A, p, s - 1), maxFromRight(A, t, r)) + Sum(A, s, t - 1)$ 
5  else return  $max(maxFromLeft(A, p, s - 1), maxFromRight(A, t, r))$ 
```

The time complexity is  $\Theta(n)$ , since  $maxFromLeft, maxFromRight, Sum$  all take  $\Theta(n)$  time, but all of them are  $\frac{1}{3}n$  size, the overall complexity is  $\Theta(n)$

## 2 Question 2: Intermediate Sequence

BUBBLE SORT( $A$ )

```
1   $A = [11, 8, 7, 5, 3, 1]$ 
2   $\rightarrow [8, 11, 7, 5, 3, 1] \rightarrow [8, 7, 11, 5, 3, 1] \rightarrow [8, 7, 5, 11, 3, 1] \rightarrow [8, 7, 5, 3, 11, 1] \rightarrow [8, 7, 5, 3, 1, 11]$ 
3   $\rightarrow [7, 8, 5, 3, 1, 11] \rightarrow [7, 5, 8, 3, 1, 11] \rightarrow [7, 5, 3, 8, 1, 11] \rightarrow [7, 5, 3, 1, 8, 11]$ 
4   $\rightarrow [5, 7, 3, 1, 8, 11] \rightarrow [5, 3, 7, 1, 8, 11] \rightarrow [5, 3, 1, 7, 8, 11]$ 
5   $\rightarrow [3, 5, 1, 7, 8, 11] \rightarrow [3, 1, 5, 7, 8, 11]$ 
6   $\rightarrow [1, 3, 5, 7, 8, 11]$ 
```

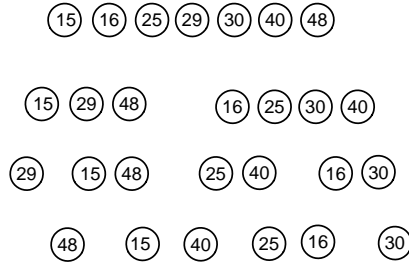


Figure 1: Merge Sort

INSERTION SORT( $A$ )

```

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

```

### 3 Question 3: Illustrate Merge Sort

See Figure 1

MERGE SORT( $A$ )

```

1  15, 16, 25, 29, 30, 40, 48
2  15, 29, 48 || 16, 25, 30, 40
3  29 || 15, 48 || 25, 40 || 16, 30
4  - || 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.

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

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

MERGE-SORT-INSERTION( $A, p, r, k$ )

```

1  if  $r - p + 1 \leq k$ 
2      Insertion-Sort( $A, p, r$ )
3      return
4  elseif  $p < r$ 
5       $q = \lfloor (p + r)/2 \rfloor$ 
6      Merge-Sort( $A, p, q$ )
7      Merge-Sort( $A, q + 1, r$ )
8      Merge ( $A, p, q, r$ )
9      return
10 else return
```

Consider  $\Theta(n \lg \frac{n}{k} + nk) = \Theta(n \lg n - n \lg k + nk) = \Omega(n \lg n)$ , When  $k = \Theta(\lg n)$ , it is OK.

But when  $k = \omega(\lg n)$ ,  $sum = \Theta(nk) = \omega(n \lg n)$ , so  $k_{max} = \Theta(\lg n)$

#### 4.3 d. how to choose k

Note that in practice, we could have:

$$T(n, k) = c_2(c_1nk + n \lg(\frac{n}{k}))$$

$$\frac{\partial T(n, k)}{\partial k} = c_2(c_1n - \frac{n}{k})$$

$$c_1 = \frac{\text{constant-of-insertion-sort}}{\text{constant-of-merge-sort}}, \text{ obviously } < 1 \text{ according to the question}$$

$$k \in [0, \infty], k = \frac{1}{c_1} = \frac{\text{constant-of-merge-sort}}{\text{constant-of-insertion-sort}} \text{ could minimize } T(n, k)$$

## 5 Question 5: CLRS Problem 6.1-3

1. Since  $x.Parent.key \geq x.key$ , we have:

When  $root.child.child \neq null$ ,  $root.child.key \geq root.child.child.key$

When  $root.child.child = null$ , the conclusion naturally correct

2. Combined with  $x.key \geq x.child.key$ , using deduction, it is easy to conclude that  $\forall h, root.child.key \geq root.(child)^h.key$

## 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:

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

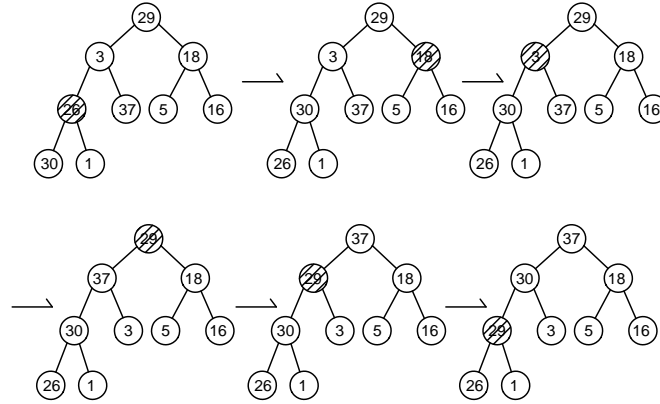


Figure 2: Build Heap

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

## 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,  $\text{Parent}[i] = \lfloor (i + d - 2) / d \rfloor$

get  $(k+1)$ th child,  $k \in [0, d - 1]$   $\text{Child}[i, k] = di + k$

### 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  $\Theta(d)$ :

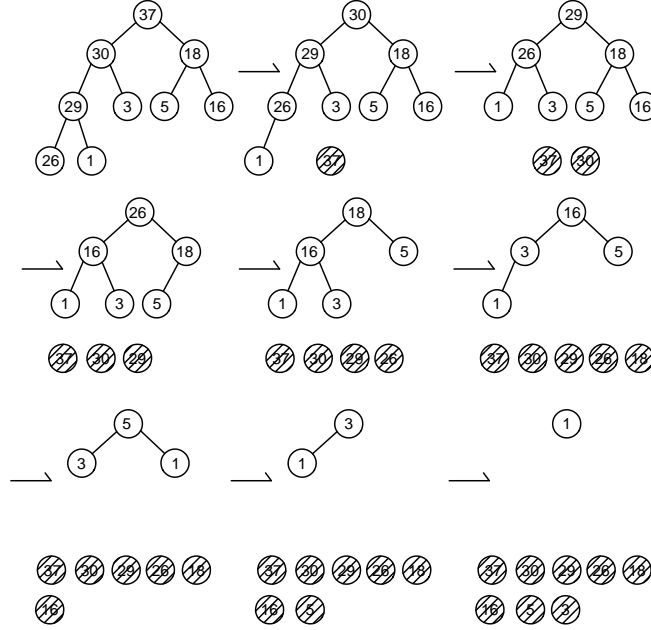


Figure 3: Heap Sort

MAXCHILD( $A, i, d$ )

```

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

```

implement of d-maxHeapify:

MAXHEAPIFY( $A, i, d$ )

```

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

```

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

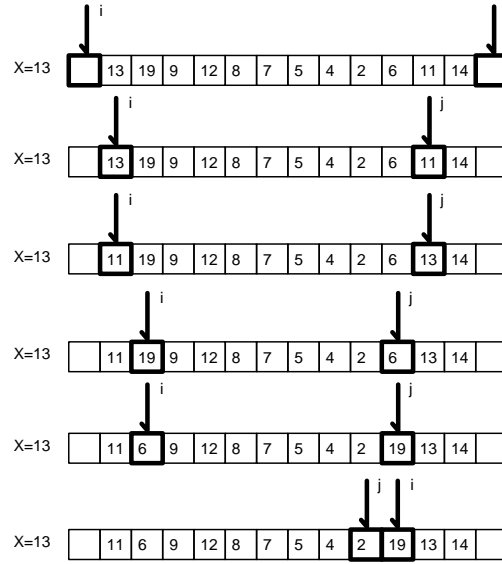


Figure 4: Hoare partition

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

```

EXTRACTMAX( $A, d$ )
1   $max = A[1]$ 
2   $swap(A[1], A[n])$ 
3   $erase(A[n])$ 
4   $maxHeapify(A, 1, d)$ 
5  return  $max$ 

```

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

## 9 Question 9: Visualize CLRS Problem 7-1

See Figure 4