Algorithm efficiency of Mergesort and Quicksort

1. Mergesort
2. Quicksort

1. Mergesort

Pseudocode from another lecture: Mergesort: a divide and conquer algorithm

```
1   MERGE(A, L, M, R)
2   input number array A, L is index of leftmost element to be
3   handled, R is index of rightmost element to be handled and M
4   is some index inbetween L and R, L ≤ M ≤ R
5   for i from L to R
6       Temp[i] = A[i]
7   end
8   iL = L, iR = M + 1, iA = L
9   while iL ≤ M and iR ≤ R
10      if Temp[iL] ≤ Temp[iR] then
11          A[iA] = Temp[iL], iL = iL + 1
12      else
13          A[iA] = Temp[iR], iR = iR + 1
14      end
15      iA = iA + 1
16  end
17  if iL > M then
18      ▷ copy Temp[iR..R] to A[iA..R]
19  else
20      ▷ copy Temp[iL..M] to A[iA..R]
21  end
```

2^x 2^x

```
1   MERGESORT(A, L, R)
2   input number array A, L is index of leftmost element to be
3   handled, R is index of rightmost element to be handled
4   if L < R then
5     M = ⌊(L + R)/2⌋
6     MERGESORT(A, L, M)
7     MERGESORT(A, M + 1, R)
8     MERGE(A, L, M, R)
9   end
```

Assumption: input array to MERGE $A[1..n]$, where $n = 2^x$, for integer x, $x \geq 2$.

Q: What is runtime efficiency of MERGE, for input $A[1..2^x]$, $L = 1$, $M = 2^{(x-1)}$, $R = 2^x$?

Observations:

- simple operations in **for**-loop in lines 5-7: $\Theta(2^x)$

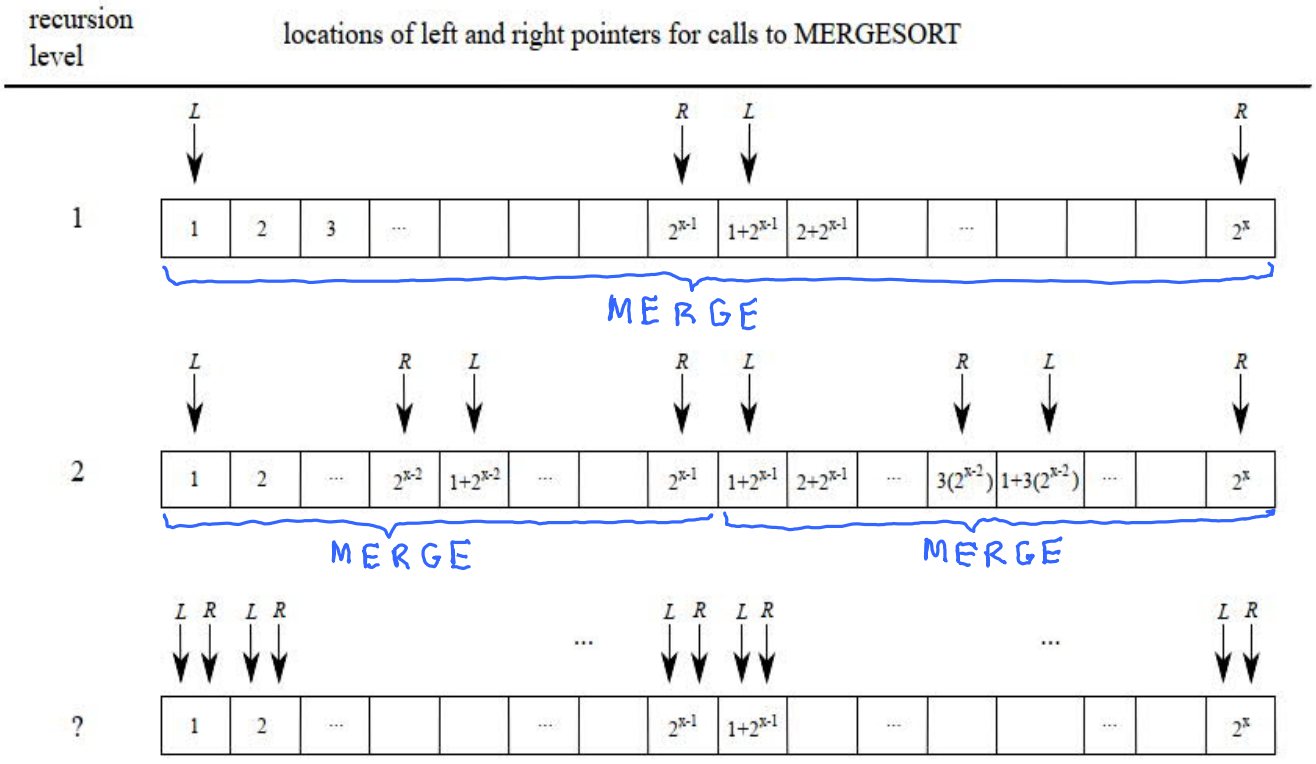
- simple operations in lines 8-21: $\Theta(2^x)$

$2^x + 2^x = 2(2^x)$

A: Runtime efficiency of MERGE: $\Theta(2^x)$.

Assumption: input array to MERGESORT: $A[1..2^x]$, $L = 1$, $R = 2^x$.

Q: How often is MERGE executed in MERGESORT and with what arguments?



recursion level	MERGE executed	elements being mereged	sum of MERGE operations
1	1 time	$A[1..2^{x-1}]$ and $A[(1+2^{x-1})..2^x]$	$2^{x-1}+2^{x-1}$
2	2 times	$A[1..2^{x-2}]$ and $A[(1+2^{x-2})..2(2^{x-2})]$, $A[1+2(2^{x-2})..3(2^{x-2})]$ and $A[(1+3(2^{x-2})..4(2^{x-2})]$	$4(2^{x-2})$
3	4 times	$A[1..2^{x-3}]$ and $A[(1+2^{x-3})..2(2^{x-3})]$, $A[1+2(2^{x-3})..3(2^{x-3})]$ and $A[(1+3(2^{x-3})..4(2^{x-3})]$ $A[1+4(2^{x-3})..5(2^{x-3})]$ and $A[(1+5(2^{x-3})..6(2^{x-3})]$ $A[1+6(2^{x-3})..7(2^{x-3})]$ and $A[(1+7(2^{x-3})..8(2^{x-3})]$	$8(2^{x-3})$
i	2^{i-1} times	$A[1..2^{x-i}]$ and $A[(1+2^{x-i})..2(2^{x-i})]$, $A[1+2(2^{x-i})..3(2^{x-i})]$ and $A[(1+3(2^{x-i})..4(2^{x-i})]$ ...	
?	2^{x-1} times	$A[1..1]$ and $A[2..2]$, $A[3..3]$ and $A[4..4]$, ... $A[(2^x-1)..(2^x-1)]$ and $A[2^x..2^x]$	

On each recurssion level all elements of $A[1..2^x]$ are merged.

Q: How many recursion levels in MERGESORT?

A: x.

Sum all operations in MERGE:

$$\underbrace{2^x + 2^x + \ldots + 2^x}_{x \text{ times}} = x \cdot 2^x$$

Using $n = 2^x$ or $\log_2 n = x$:

$$\text{runtime efficiency of MERGESORT: } \Theta(n \log_2 n)$$

Q: How about when $n \neq 2^x$?

A: Same efficiency.

2. Quicksort

Pseudocode from another lecture: Quicksort: a divide and conquer algorithm

```
1   PARTITION(A, L, R)
2   input number array A, L is index of leftmost element to be
3   handled, R is index of rightmost element to be handled
4   α = A[R], cut = L − 1
5   for j = L to R − 1
6     if A[j] ≤ α then
7       cut = cut + 1, swap elements A[cut] and A[j]
8     end
9   end
10  k = cut + 1, swap elements A[k] and A[R]
11  return k
```

$2^x - 2$

```
1   QUICKSORT(A, L, R)
2   input number array A, L is index of leftmost element to be
3   handled, R is index of rightmost element to be handled
4   if L < R then
5     k = PARTITION(A, L, R)
6     QUICKSORT(A, L, k − 1)
7     QUICKSORT(A, k + 1, R)
8   end
```

Assumption: input array is $A[1..n]$, where $n = 2^x - 1$, for integer x, $x \geq 2$.

Q: What is runtime efficiency of PARTITION when input is $A[1..(2^x - 1)]$, $L = 1$, $R = (2^x - 1)$.

A: Runtime efficiency is $\Theta(2^x - 1) = \Theta(2^x)$

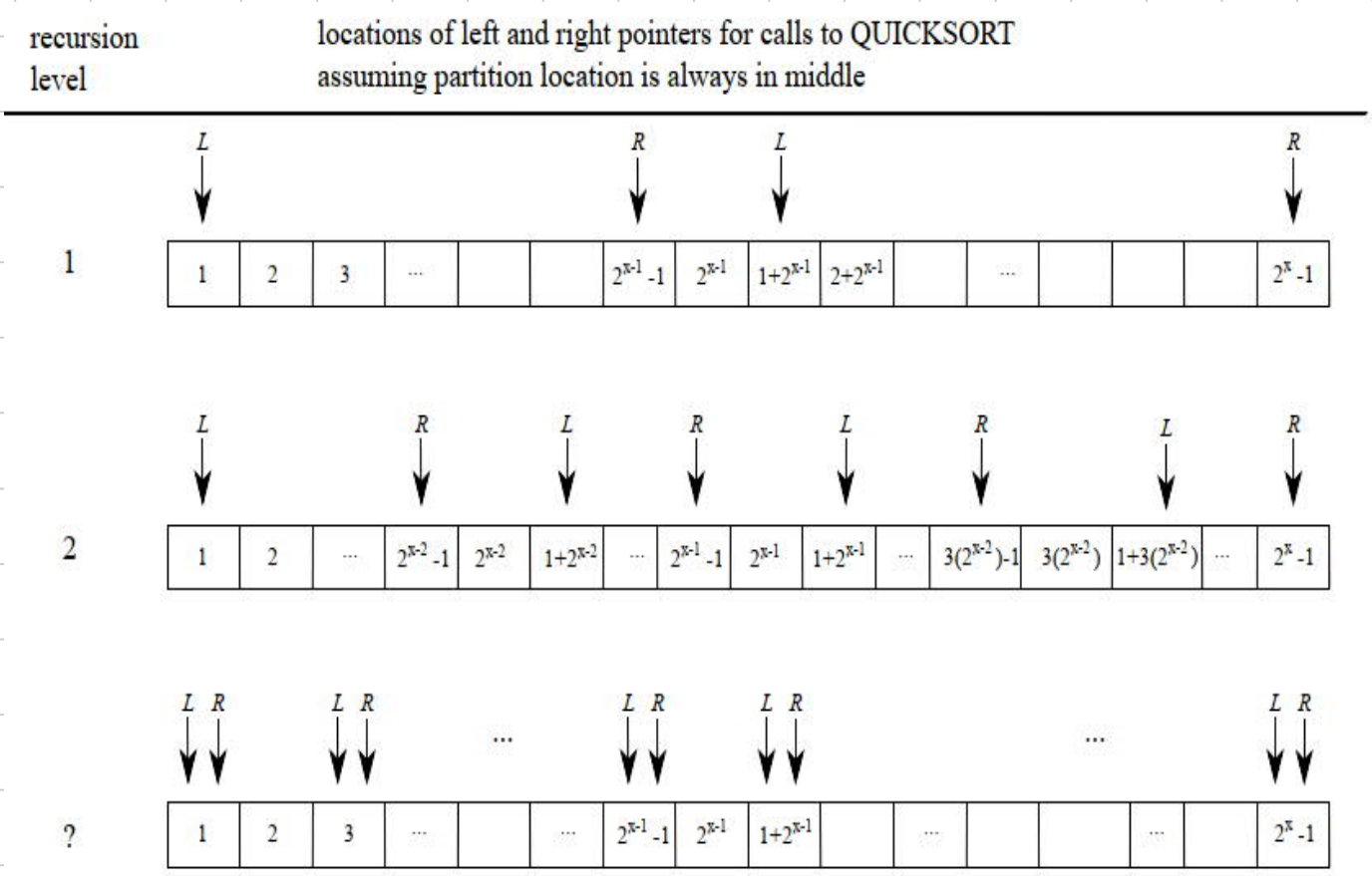
Assumption: input to QUICKSORT is $A[1..(2^x - 1)]$, $L = 1$, $R = (2^x - 1)$.

Q: How often is PARTITION executed in QUICKSORT and with what arguments?

A: Depends on pivot location (value of *k*).

Assumption: pivot is always in middle $k = \lfloor (L + R)/2 \rfloor$.

This assumption corresponds to best case (lower bound on runtime efficiency).



recursion level	PARTITIONs executed	arrays in calls to ~~QUICKSORT~~ PARTITION assuming partition always in middle
1	1 time	$A[1..(2^x - 1)]$
2	2 times	$A[1..(2^{x-1} - 1)]$, $A[(1 + 2^{x-1})..(2^x - 1)]$
3	4 times	$A[1..(2^{x-2} - 1)]$, $A[(1 + 2^{x-2})..(2(\cdot 2^{x-2}) - 1)]$, $A[(2(\cdot 2^{x-2}) + 1)..(3(\cdot 2^{x-2}) - 1)]$ and $A[(3(\cdot 2^{x-2}) + 1)..(2^x - 1)]$
i	2^{i-1} times	$A[1..(2^{x-i+1} - 1)]$, $A[(2^{x-i+1} + 1)..(2(2^{x-i+1}) - 1)]$, $A[(2(2^{x-i+1}) + 1)..(3(2^{x-i+1}) - 1)]$ ...
?	2^{x-1} times	$A[1..1]$, $A[3..3]$, ... $A[(2^x - 1)..(2^x - 1)]$

$2^x - 1$

$2(2^{x-1} - 1)$

NOTE: arrays in last row are actually not partitioned, but they are used in a call to QUICKSORT.

Q: How many recursion levels when pivot is always in middle?

A: *x*

Sums of all array sizes in previous table:

$$2^x - 1 + 2(2^{x-1} - 1) + 4(2^{x-2} - 1) + 8(2^{x-3} - 1) + \ldots + 2^{x-1}(2^{x-(x-1)} - 1)$$
$$= 2^x - 1 + 2^x - 2 + 2^x - 4 + \ldots + 2^x - 2^{x-1}$$
$$= x2^x - \sum_{j=0}^{x-1} 2^j = x2^x - 2^x - 1 = (x-1)2^x - 1$$

When $n = 2^x - 1$, $(x-1) \leq \log_2 n \leq x$ and it can be shown that for $n \geq 4$:

$$(1/2)n \log_2 n \leq (x-1)2^x - 1 \leq 2n \log_2 n$$

Runtime efficiency for best case of QUICKSORT is $\Theta(n \log_2 n)$.

Worst case for Quicksort in lecture: Algorithm design techniques: randomization

Efficiency result for both MERGESORT and QUICKSORT: $n \log_2 n$

Are MERGESORT and QUICKSORT equally efficient?

- yes (theoretically) when considering best case for QUICKSORT
- in practice QUICKSORT usually better, but it requires 'tuning'