Introduction to Algorithms

CELEN086

Seminar 6
(w/c 18/11/2024)

CELEN086 Mid Examination Information

- **Wednesday 20 November 2024, 14:00-15:00**
- **Time Duration : 60 minutes**
- There are 15 Multiple Choice Questions (MCQ) and 5 short answer questions.
- ALL topics up to and including Lecture 6.
- Calculators **ARE NOT ALLOWED!**
- You can bring a hardcopy English-Chinese dictionary
- Bring pen/pencil and your **ID CARD**

Outline

In this seminar, we will study and review on following topics:

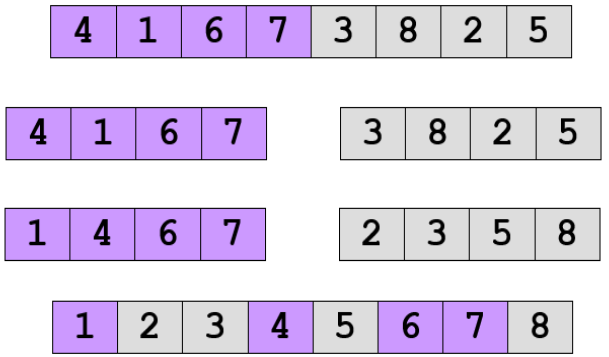
- Split and Merge algorithms in Merge Sort
- Tracing Merge Sort algorithm
- Time complexity of Merge Sort
- Bucket Sort

You will also learn useful Math/CS concepts and vocabularies.

Merge Sort

Merge sort has three components:

- Split the list into two sub-lists
- Sort each sub-list (recursively)
- Merge the sub-lists



How to split one list into two lists?

Do we need helper function?

Split

tests

split_1([a])

n = 1

(L1, L2) = splitHelper(0, Nil, [a])

= (Nil, [a])

= (Nil, [a])

split_1([a, b, c, d, e])

n = 5

(L1, L2) = splitHelper(2, Nil, [a, b, c, d, e])

= splitHelper(1, [a] , [b, c, d, e])

= splitHelper(0, [b, a], [c, d, e])

= ([b, a], [c, d, e])

= (reverse([b, a]), [c, d, e])

= ([a, b], [c, d, e])

Split Algorithm

Algorithm: splitHelper(list1,list2,m)
Requires: two lists and an integer m
Returns: two lists in an ordered pair

1. if m==0
2. return (list1, list2)
3. else
4. return splitHelper(cons(value(list2),list1),tail(list2),m-1)
5. endif

Algorithm: split(list) [main]
Requires: one list
Returns: two lists in an ordered pair, each contains half elements of input list

1. let n=length(list)
2. return splitHelper(Nil ,list, n/2)

Practice: Trace split([5,7,3,2,9])

list=[5,7,3,2,9] Line 1: n=5 Line 2: return splitHelper([ ],[5,7,3,2,9],2)

list1=[ ] list2=[5,7,3,2,9] m=2 Line 4: return splitHelper([5],[7,3,2,9],1)

list1=[5] list2=[7,3,2,9] m=1 Line 4: return splitHelper([7,5],[3,2,9],0)

list1=[7, 5] list2=[3,2,9] m=0 Line 2: return ([7,5],[3,2,9])

Split Algorithm

What is the time complexity of this algorithm? $O(n)$

How can we fix the small issue here?

(L1,L2)=split([5,7,3,2,9])

(L1,L2)=([5,7],[3,2,9])

Algorithm: split(list) [main]
Requires: one list
Returns: two lists in an ordered pair, each contains half elements of input list

1. let n=length(list)
2. (R1,L2)=splitHelper([ ],list, n/2)
3. let L1=reverse(R1)
4. return (L1,L2)

Merge Algorithm

Algorithm: merge(list1, list2)
Requires: two sorted lists
Returns: a merged (and sorted) list

```
1.  if  isEmpty(list1)
2.      return  list2  // base case
3.  elseif  isEmpty(list2)
4.      return  list1  // base case
5.  elseif  value(list1)<value(list2)
6.      return  cons(value(list1),merge(tail(list1), list2) )
7.  else
8.      return  cons(value(list2),merge(list1, tail(list2) ))
9.  endif
```

[Homework exercise 1]

Trace merge([5,7],[2,3,9])

What is the time complexity of this algorithm? $O(n)$

Merge Sort Algorithm

Algorithm: mergeSort(list)
Requires: one list
Returns: a sorted list (with same elements)

```
1.  if isEmpty(list) || isEmpty(tail(list))
2.      return list // list is already sorted (base case)
3.  else
4.      let (L1,L2) = split(list)  // splitting list
5.      let S1 = mergeSort(L1) // sorting (recursive call)
6.      let S2 = mergeSort(L2) // sorting (recursive call)
7.      return merge(S1, S2) // merging
8.  endif
```

Trace
mergeSort([5,8,4,9,2,3,1]) = [1,2,3,4,5,8,9]

(only key statements are shown here)

(L1,L2)=([5,8,4],[9,2,3,1])

S1=mergeSort([5,8,4])

=[4,5,8]

S2=mergeSort([9,2,3,1])

=[1,2,3,9]

return [1,2,3,4,5,8,9]

(L1,L2)=([5],[8,4])

S1=mergeSort([5])=[5]

S2=mergeSort([8,4])=[4,8]

return [4,5,8]

.....

[Homework exercise 2]

(L1,L2)=([8],[4])

S1=mergeSort([8])=[8]

S2=mergeSort([4])=[4]

return [4,8]

.....

Time complexity of Merge Sort

How many "levels" of splitting?

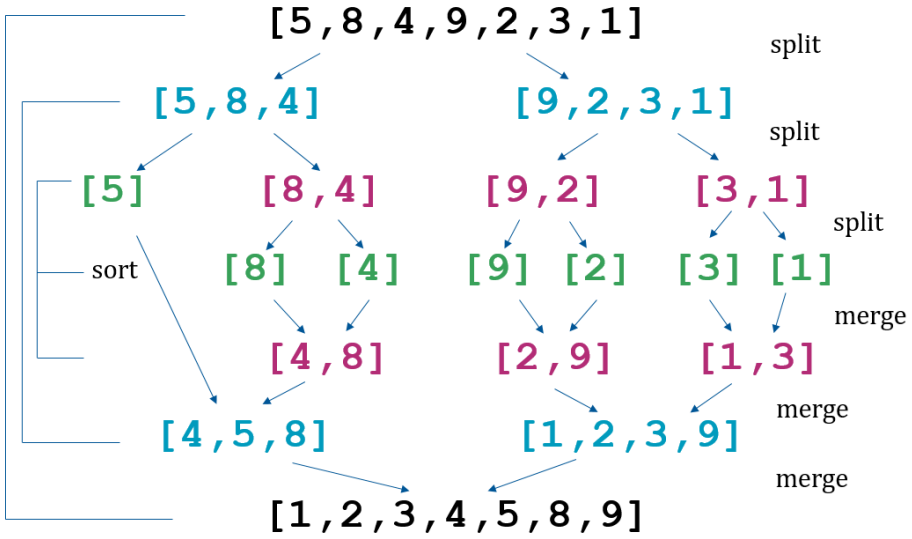
$$\lceil \log_2 7 \rceil = 3$$

$ceil(x) = \lceil x \rceil$ e.g. $\lceil 2.5 \rceil = 3$

Round x up to the nearest integer.

$floor(x) = \lfloor x \rfloor$ e.g. $\lfloor 2.5 \rfloor = 2$

Round x down to the nearest integer.



If the length of list is n, there are about $\log_2 n$ levels of splitting/merging.

At each level, the splitting and merging take $O(n)$ operations.

Therefore, merge sort has a time complexity of $O(n \log_2 n)$

Practice

Show the process of sorting each of the following lists using merge sort:

$$L1=[9, 3, 5, 8, 2, 4, 7, 1]$$

$$L2=[12, 5, 17, 9, 15, 28, 4\ ]$$

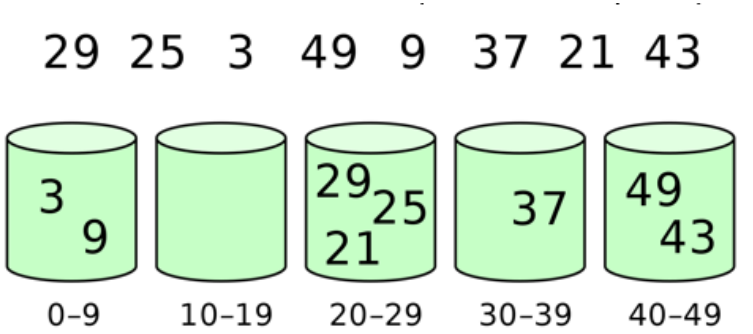
Note: different question styles in exam

- Trace the algorithm — You must follow the step-wise instructions in the algorithm and include key statements in all recursive calls.
- Show the process of computing/finding/searching/sorting… using the algorithm — You may follow the idea of such algorithms and solve the problem with table/diagram for demonstration

Bucket Sort

Process of bucket sort:

- Initializing k empty bucket
- Scattering
- Sorting within each bucket
- Gathering



[3,9] [21,25,29] [37] [43,49]

[3,9,21,25,29,37,43,49]

Which one of following lists is ideal for bucket sort?

L1=[28, 33, 21, 15, 19, 43, 17, 25, 36, 22]

L2=[28, 23, 21, 25, 29, 43, 27, 25, 36, 22]

Practice

Show the process of sorting the list using bucket sort:

[28, 36, 20, 15, 19, 43, 17, 25, 33, 22]

	Bucket 1: $10 \leq x < 20$	Bucket 2: $20 \leq x < 30$	Bucket 3: $30 \leq x < 40$	Bucket 4: $40 \leq x < 50$
(Initializing)				
(Scattering)	[15, 19, 17]	[28, 20, 25, 22]	[36, 33]	[43]
(Sorting)	[15, 17, 19]	[20, 22, 25, 28]	[33, 36]	[43]

(Gathering) [15, 17, 19, 20, 22, 25, 28, 33, 36, 43]

Best/average case time complexity: $O(n + k)$