CSC 225

addie@uvic.ca

the time

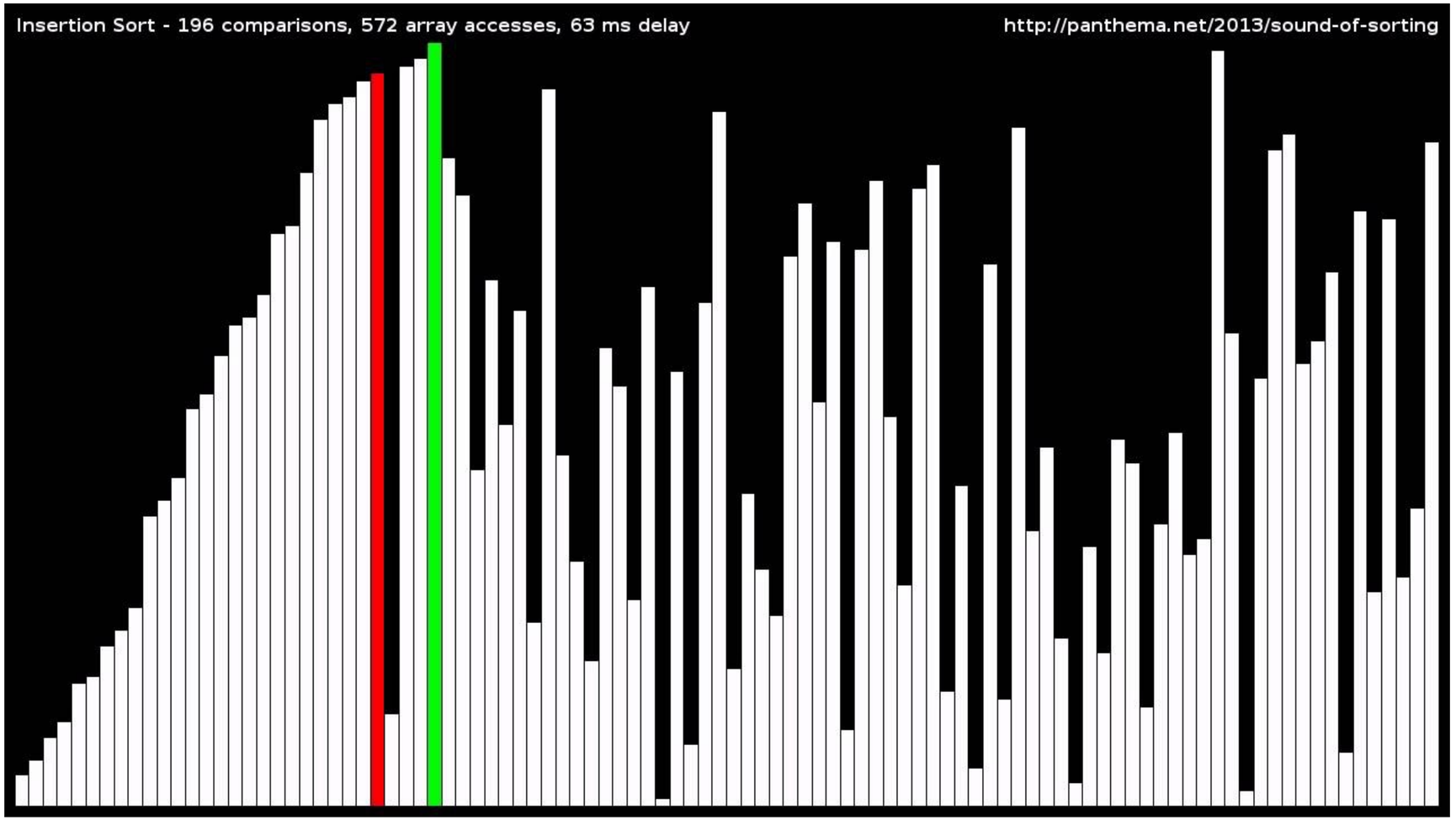
Announcements

- program 1 due on Saturday
  - please be sure to do your own testing!
- quiz 6 out today, due tomorrow

Poll

menti.com

Insertion Sort



Insertion Sort

Idea: like adding new elements to an already sorted list. slot in the element.

Show what the list looks like after each pass

{10, 3, 26, 9, 4, 8, 12}

Insertion Sort

```
Algorithm InsertionSort(A,n):

  for k ← 1 to n-1 do
    val ← A[k]
    j ← k-1
    while j ≥ 0 and A[j] > val do
      A[j+1] ← A[j]
      j ← j-1
    end
    A[j+1] ← val
  end
end
```

What is the worst-case run-time?

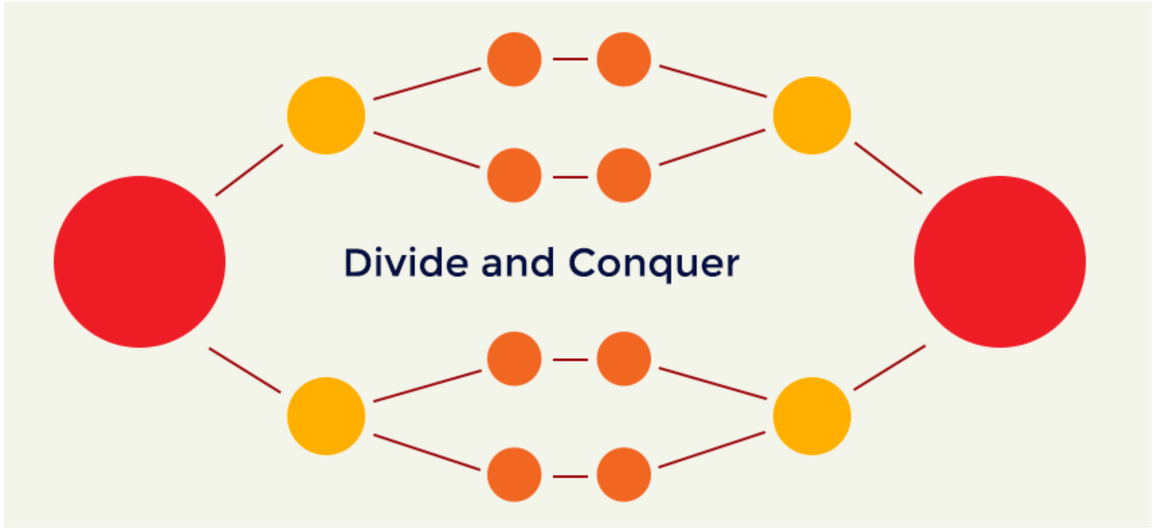
Insertion Sort

```
Algorithm InsertionSort(A,n):

  for k ← 1 to n-1 do
    val ← A[k]
    j ← k-1
    while j ≥ 0 and A[j] > val do
      A[j+1] ← A[j]
      j ← j-1
    end
    A[j+1] ← val
  end
end
```

Best case input?
Worst case input?

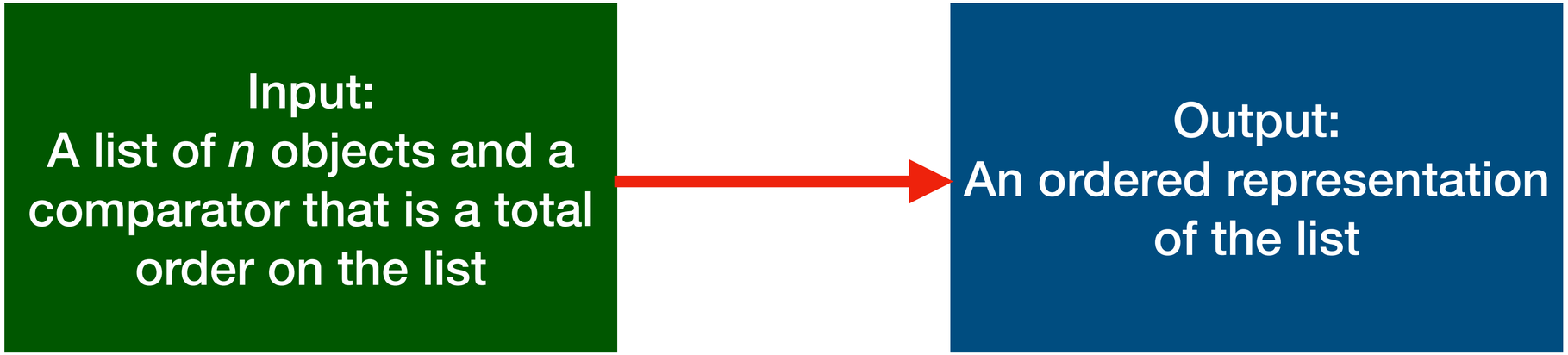
Divide and Conquer

- an efficient and easy-to-understand algorithm technique
    - Binary Search
    - Merge sort
    - Quick sort
- repeatedly divide a problem into smaller pieces
    - solve smaller pieces
    - sometimes may need to combine smaller solutions



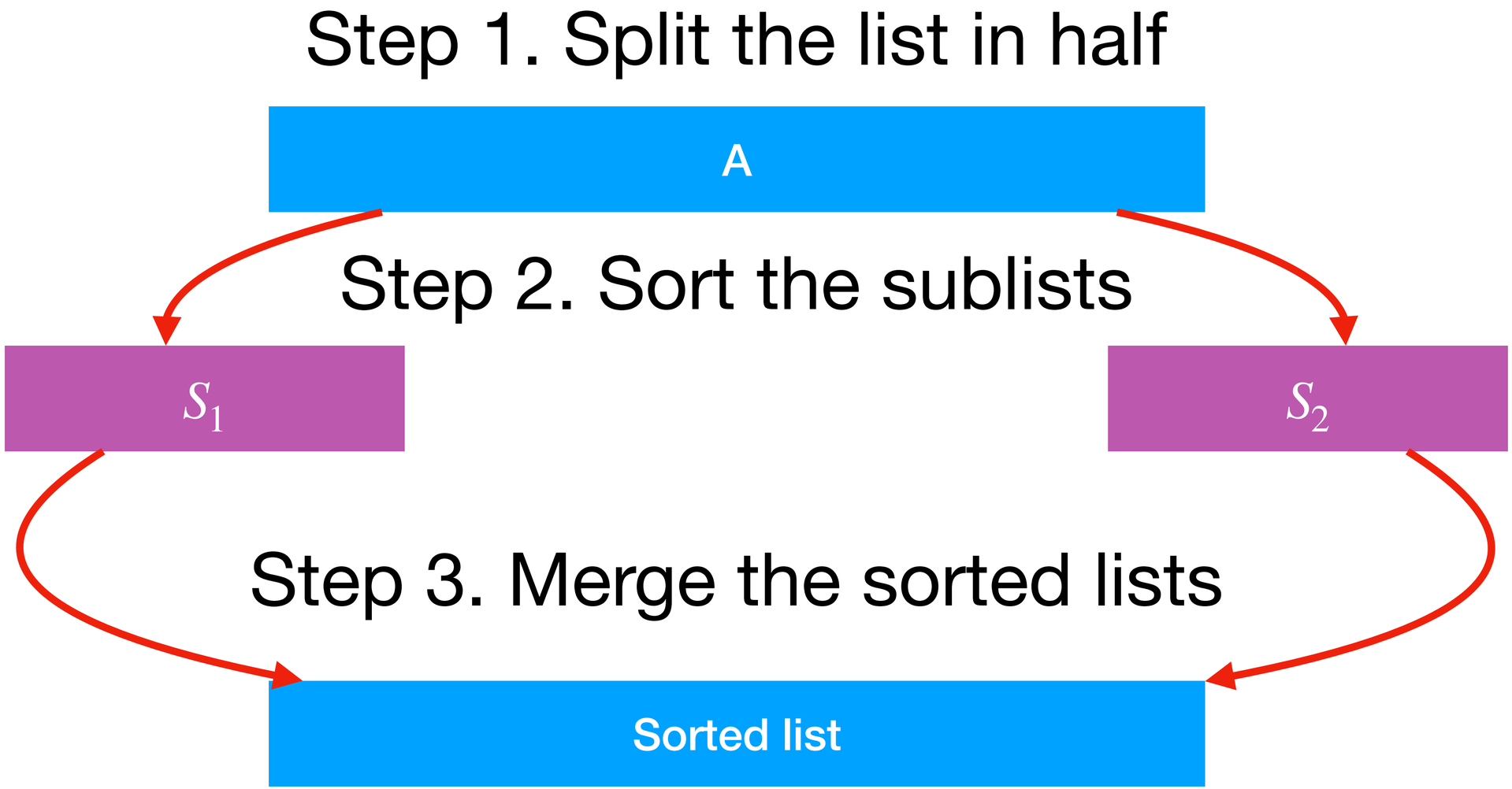
Merge Sort

Definition + requirements



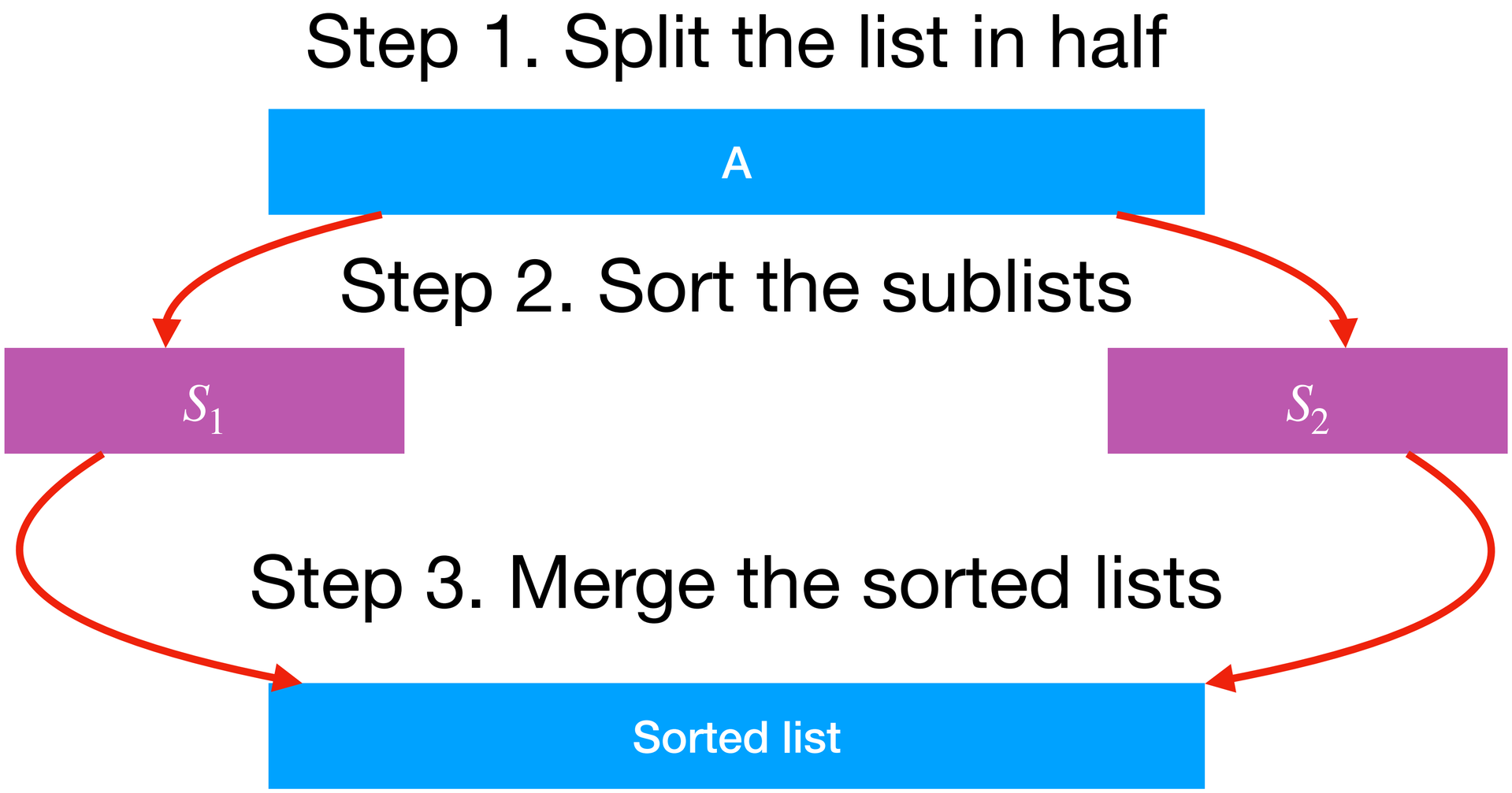
Merge Sort

Intuition

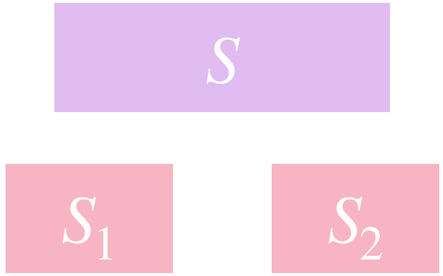


Merge sort: 3 steps

- Divide
- Recur
- Merge



Step 1: Divide

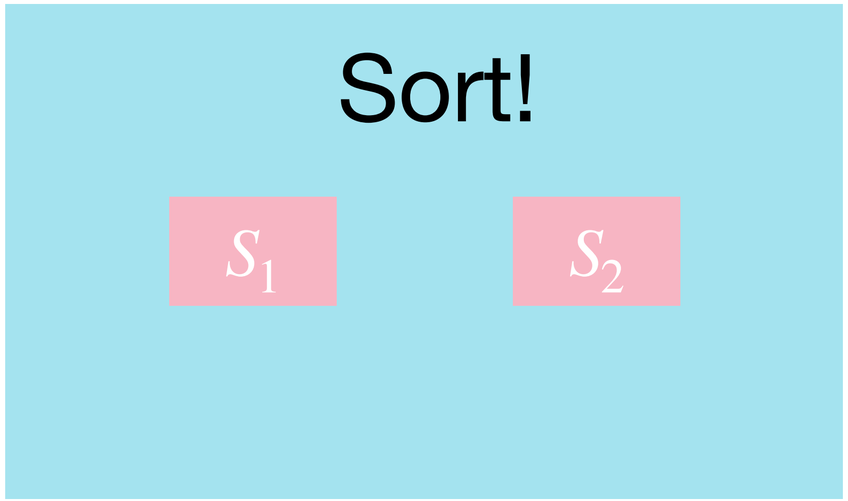


Let S be a sequence of n elements.

- If $|S| = 0$ or $|S| = 1$, return S
- Otherwise, break S into two parts S_1 and S_2 such that they each contain about half of the elements in S

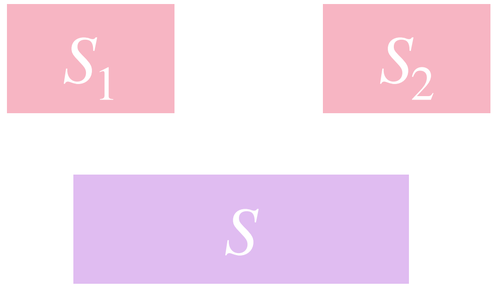
Step 2: Recur

- Recursively sort S_1 and S_2



Step 3: Merge

- Merge S_1 and S_2 to obtain a sorted S



Merging two sorted lists

- Look at the smallest elements in either list and compare
- Add the smaller one to the final sorted list
- Repeat until both sublists are empty

Merging two sorted lists

Example

Let $S = [8,1,11,4,12,3,7,5]$. Draw the Merge sort tree.

Merge sort algorithm

```
Algorithm MergeSort(S):

  if S.size() < 2 then
    return S

  divide(S1, S2, S)
  S1 ← MergeSort(S1)
  S2 ← MergeSort(S2)
  merge(S1, S2, S)

  return S
```

Merge sort algorithm

```
Algorithm MergeSort(S):

  if S.size() < 2 then
    return S

  divide(S1, S2, S)
  S1 ← MergeSort(S1)
  S2 ← MergeSort(S2)
  merge(S1, S2, S)

  return S
```

```
Algorithm divide(S1, S2, S):

  for i←0 to ⌈n/2⌉ − 1 do
    S1[i] ← S[i]
  end

  for i ← ⌈n/2⌉ to n-1 do
    S2[i-⌈n/2⌉] ← S[i]
  end

end
```

Merging two sorted lists

- Look at the smallest elements in either list and compare
- Add the smaller one to the final sorted list
- Repeat until both sublists are empty

Merge sort algorithm

```
Algorithm MergeSort(S):

  if S.size() < 2 then
    return S

  divide(S1, S2, S)
  S1 ← MergeSort(S1)
  S2 ← MergeSort(S2)
  merge(S1, S2, S)

  return S
```

```
Algorithm merge(S1, S2, S):

  i ← 1
  j ← 1

  while i ≤ n1 and j ≤ n2 do
    if S1[i] ≤ S2[j] then
      S[i+j-1] ← S1[i]
      i ← i + 1
    else
      S[i+j-1] ← S2[j]
      j ← j + 1
  while i ≤ n1 do
    S[i+j-1] ← S1[i]
    i ← i + 1
  while j ≤ n2 do
    S[i+j-1] ← S2[j]
    j ← j + 1
```

Merge sort algorithm

```
Algorithm MergeSort(S):

  if S.size() < 2 then
    return S

  divide(S1, S2, S)
  S1 ← MergeSort(S1)
  S2 ← MergeSort(S2)
  merge(S1, S2, S)

  return S
```

Worst case running time?

Merge sort alg

Worst case running time?

```
Algorithm MergeSort(S):

  if S.size() < 2 then
    return S

  divide(S1, S2, S)
  S1 ← MergeSort(S1)
  S2 ← MergeSort(S2)
  merge(S1, S2, S)

  return S
```

```
Algorithm divide(S1, S2, S):

  for i←0 to ⌈n/2⌉ − 1 do
    S1[i] ← S[i]
  end

  for i ← ⌈n/2⌉ to n-1 do
    S2[i-⌈n/2⌉] ← S[i]
  end

end
```

Merge so

Worst case running time?

```
Algorithm MergeSort(S):

  if S.size() < 2 then
    return S

  divide(S1, S2, S)
  S1 ← MergeSort(S1)
  S2 ← MergeSort(S2)
  merge(S1, S2, S)

  return S
```

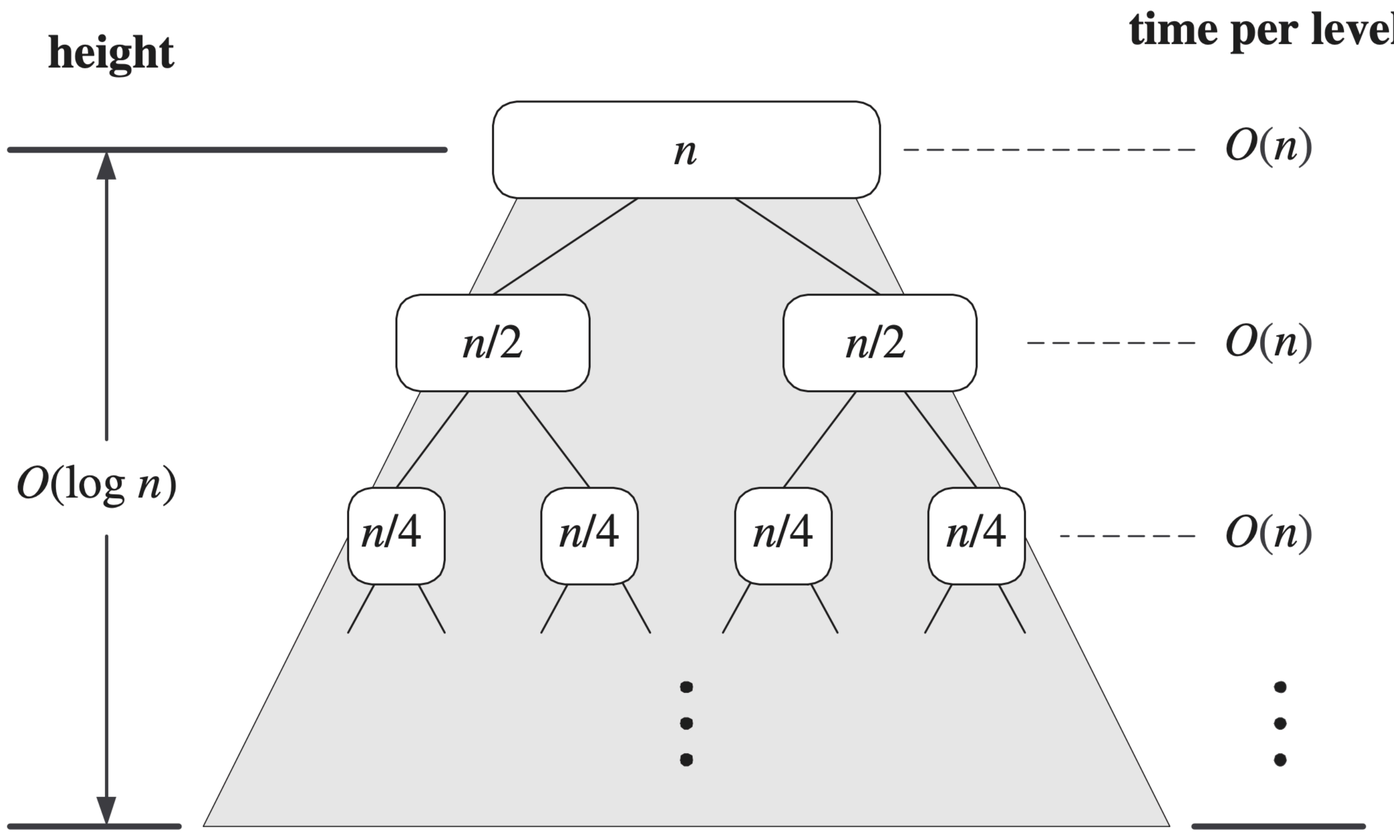
```
Algorithm merge(S1, S2, S):

  i ← 1
  j ← 1

  while i ≤ n1 and j ≤ n2 do
    if S1[i] ≤ S2[j] then
      S[i+j-1] ← S1[i]
      i ← i + 1
    else
      S[i+j-1] ← S2[j]
      j ← j + 1
  while i ≤ n1 do
    S[i+j-1] ← S1[i]
    i ← i + 1
  while j ≤ n2 do
    S[i+j-1] ← S2[j]
    j ← j + 1
```

Merge sort algorithm

```
Algorithm MergeSort(S):

  if S.size() < 2 then
    return S

  divide(S1, S2, S)
  S1 ← MergeSort(S1)
  S2 ← MergeSort(S2)
  merge(S1, S2, S)

  return S
```

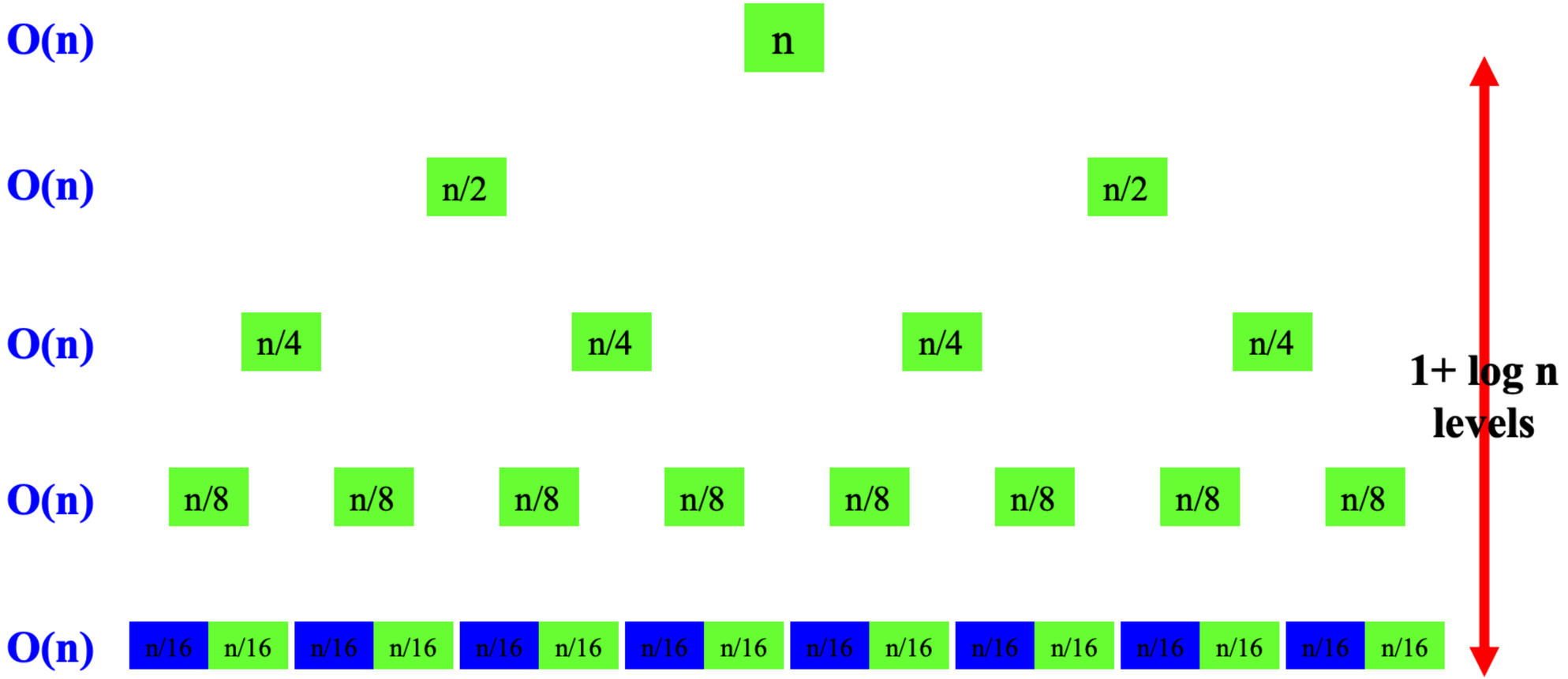
Worst case running time?

Depth recursion of Merge sort



Total time: $O(n \log n)$

Depth recursion of Merge sort



n leaves nodes
n-1 interior nodes

$$T(n) = n \log n \in O(n \log n)$$