

CSE 21: Math for Algorithm & System Analysis Summer II 2006

Lecture 11

Wenjing Rao
wrao@cs.ucsd.edu

Review: Asymptotic Order of Growth

- Upper bounds
 - Big-Oh: $f(n) = O(g(n))$
- Lower bounds
 - Big-Omega: $f(n) = \Omega(g(n))$
- Tight bounds
 - Big-Theta: $f(n) = \Theta(g(n))$

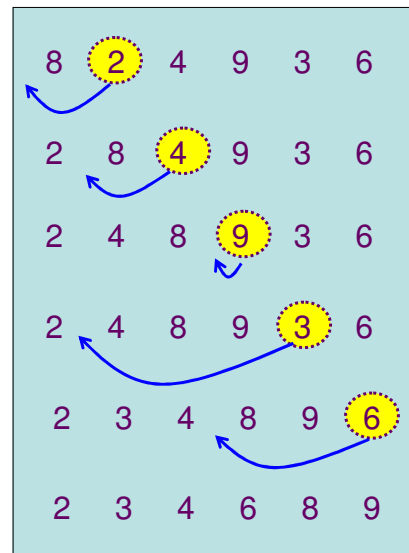
Today's Topic

- Analyze the growth of a function with O , Ω , and Θ notations

Analyze insertion sort

- Insertion sort

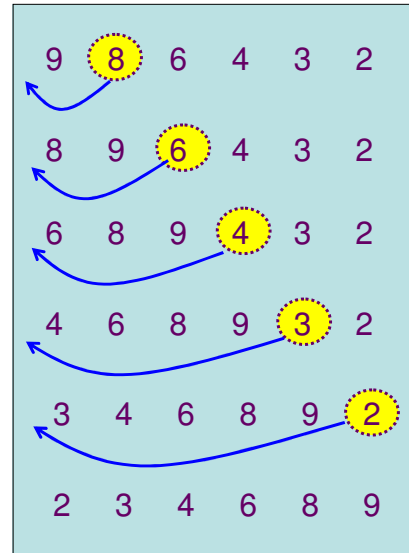
```
for k:=2 to n
  temp:=a[k]
  j:=k-1
  while(j>0 & a[j]>temp)
    a[j+1]:=a[j]
    j:=j-1
  a[j+1]:=temp
next k
```



Insertion sort – worst case

```

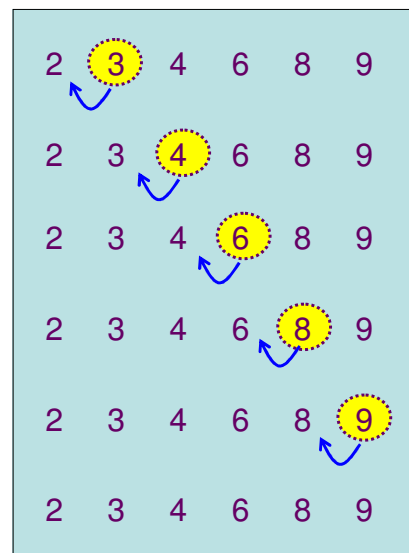
for k:=2 to n
  temp:=a[k]
  j:=k-1
  while(j>0 & a[j]>a[k])
    a[j+1]:=a[j]
    j:=j-1
  a[j+1]:=temp
next k
    
```



Insertion sort – best case

```

for k:=2 to n
  temp:=a[k]
  j:=k-1
  while(j>0 & a[j]>a[k])
    a[j+1]:=a[j]
    j:=j-1
  a[j+1]:=temp
next k
    
```

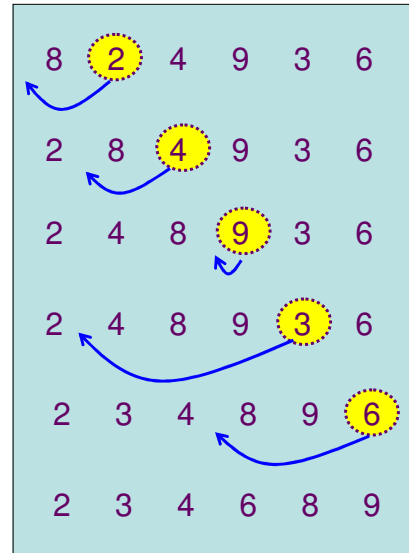


Summary: insertion sort analysis

- Insertion sort

```

for k:=2 to n
  temp:=a[k]
  j:=k-1
  while(j>0 & a[j]>a[k])
    a[j+1]:=a[j]
    j:=j-1
  a[j+1]:=temp
next k
    
```

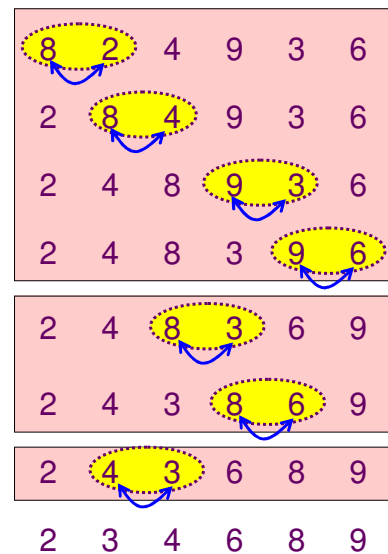


Analyze bubble sort

- Bubble sort

```

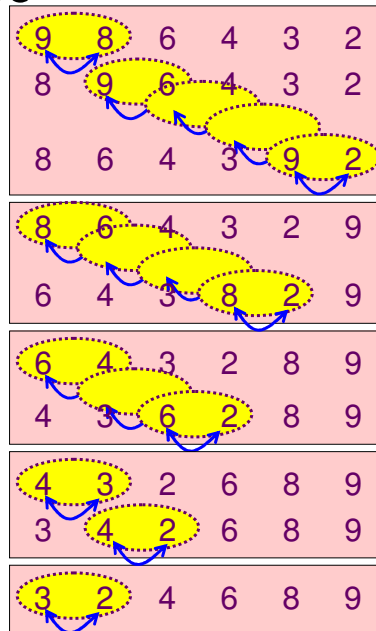
do
  swapped := 0
  for i:=1 to n-1
    if (a[i] > a[i+1])
      swap(a[i], a[i+1])
      swapped:=1
  next i
repeat while(swapped)
    
```



Bubble sort – worst case

```

do
  swapped := 0
  for i:=1 to n-1
    if (a[i] > a[i+1])
      swap(a[i], a[i+1])
      swapped:=1
  next i
repeat while(swapped)
  
```



Bubble sort – best case

```

do
  swapped := 0
  for i:=1 to n-1
    if (a[i] > a[i+1])
      swap(a[i], a[i+1])
      swapped:=1
  next i
repeat while(swapped)
  
```

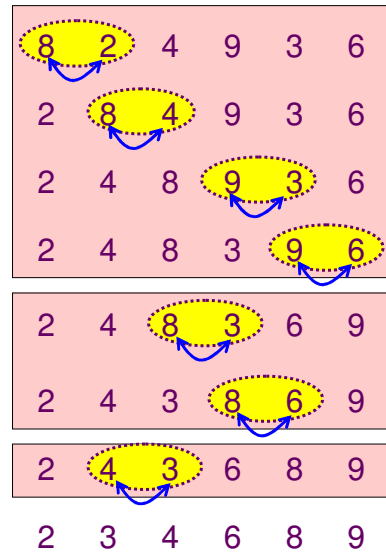


Summary: bubble sort analysis

- Bubble sort

```

do
  swapped := 0
  for i:=1 to n-1
    if (a[i] > a[i+1])
      swap(a[i], a[i+1])
      swapped:=1
  next i
repeat while(swapped)
    
```



Mergesort: analysis

- $T(n) = 2 T(n/2) + \Theta(n)$

Mergesort(L)

If $|L| = 1$,

Return L

Else

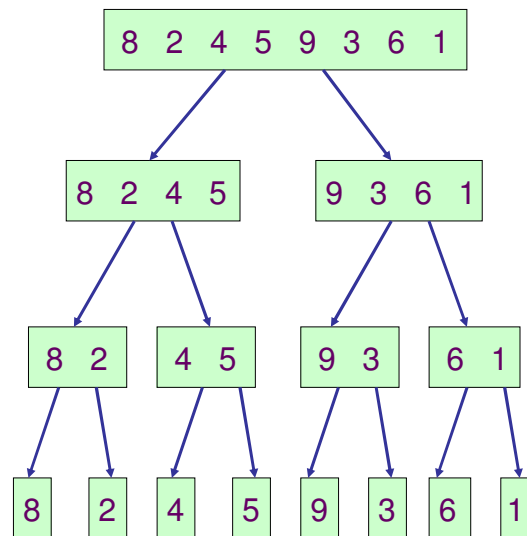
$(L1, L2) = \text{Split}(L)$

Mergesort(L1)

Mergesort(L2)

$L = \text{Merge}(L1, L2)$

Return L



Mergesort: best & worst cases?

- $T(n) = 2 T(n/2) + \Theta(n)$

Mergesort(L)

If $|L| = 1$,

Return L

Else

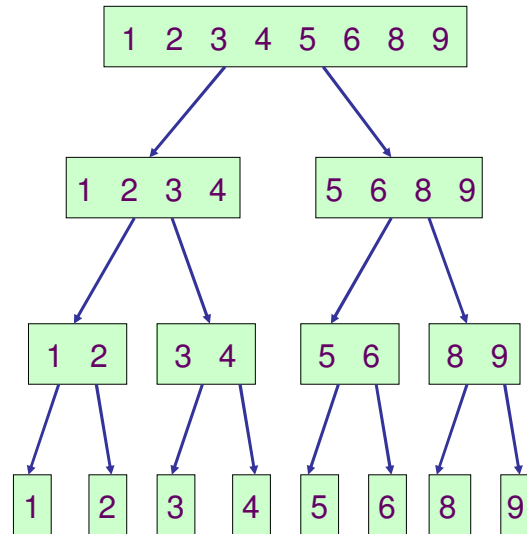
$(L1, L2) = \text{Split}(L)$

Mergesort(L1)

Mergesort(L2)

$L = \text{Merge}(L1, L2)$

Return L



Bubble/Insertion sort vs Mergesort

- Bubble/insertion sort
- Merge sort

The worst ever sorting algorithm

- n elements: $a(1), a(2), \dots, a(n)$
 - # of possible permutation?
 - Check each to see (exhaustive search)

Various growths of functions

n	10	20	50	100
$\lg(n)$				
$n \lg(n)$				
n^2				
n^3				
2^n				
$n!$				

Loop analysis

- Constant loops

```
for i:=1 to n
  for j:=1 to n
    a[i,j]=0
  next j
next i
```

Loop analysis

- Loops with floor function

```
for k:=1 to  $\lfloor n/2 \rfloor$ 
  a[k]= a[k]+1
next k
```

Loop analysis

- Loops with dependent variables

```
for i:=1 to n
  for j:=1 to i
    a[i,j]=0
  next j
next i
```

Loop analysis

- General loops with dependent variables

```
for i:=1 to n
  for j:=1 to i
    for k:=1 to j
      a[i,j,k]:=i+j+k
    next k
  next j
next i
```

Search algorithm

- Sequential search through an unsorted array

Seq_search(x)

 find:=0

 i:=1

 while (find ≠ 1)

 if (a[i]=x)

 find:=1

 end while

 return find

2 6 7 3 8 9

Best case?

Worst case?

Search algorithm

- Sequential search through an unsorted array

Seq_search(x)

 find:=0

 i:=1

 while (find ≠ 1)

 if (a[i]=x)

 find:=1

 end while

 return find

2 6 7 3 8 9

Average case:

Search algorithm

- Binary search through a sorted array

Bin_search(x)

find:=0

bot:=1

top:=n

while (top≥bot & find=0)

mid:= $\lfloor (bot+top)/2 \rfloor$

if (a[mid]=x)

find:=mid

else if (a[mid]>x)

top:=mid-1

else (a[mid]<x)

bot:=mid+1

end while

return find

2 3 5 7 8 9 11

Best case?

Worst case?

Search algorithm

- Recursive version of Binary search

Bin_search(x, a, bot, top)

if(bot>top)

return not found

else if(x=a[mid])

return mid

else if(x>a[mid])

return **Bin_search(x, a, mid+1, top)**

else (x<a[mid])

return **Bin_search(x, a, bot, mid-1)**

2 3 5 7 8 9 11