CS1102: Lecture 13

Mix and Match

Final Exam

Date:	24 April 2010 (Sat)
Time:	1:00pm
Time allowed:	2 hours

Final Exam Scopes

- Entire semester with emphasis on the second half
  - Lecture notes
  - Tutorials and labs

Final Exam paper format

- **Closed book** examination
- Section 1 consists of 12 MCQ questions (3 marks each), total 36 marks.
  - Answer the MCQs by shading the OCR form.
- Section 2 consists of 5 short answer questions (10 to 16 marks each, total 64 marks)
  - Each question has two or three parts.
  - Answer each question directly in the space given after each question. If necessary, use the back of the page
- Total 100 marks.

Final Exam paper format …

- Please bring a 2B or darker pencil for filling the OCR form.
- Please write clearly and with reasonable big characters.
- Remember to write your **matriculation number** on the front page of the examination paper.
- You may write in pencil for the this examination if you wish.

CS1102 Objectives

- Give an introduction to data structures and algorithms for constructing **efficient** computer programs.
- Emphasize on **data abstraction** issues (through ADTs) in the code development.
- Emphasize on **efficient implementations** of chosen data structures and algorithms.

CS1102 Objectives

- Include arrays, lists, stacks, queues, trees (including BST and heap), hash tables, and graphs; together with their algorithms (insert, delete, find, tree and graph traversals and updates).
- Simple algorithmic paradigms, such as **sorting** and **search** algorithms, **greedy** algorithms and **divide-and-conquer** algorithms were introduced.
- Elementary **analysis of algorithmic complexities** were taught.

Data Structures with Multiple Organization

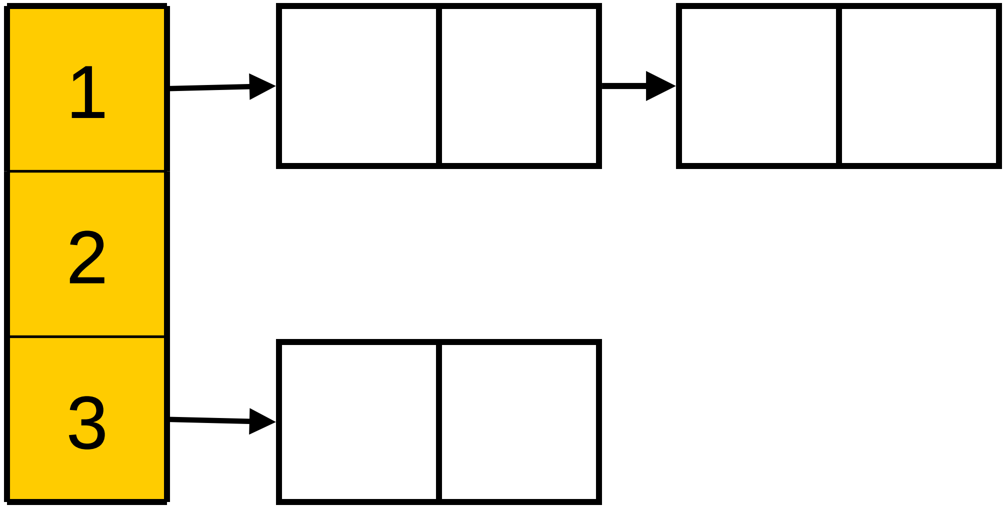
Basic Data Structures

- Arrays
- Linked Lists
- Trees

We can combine them to implement different data structures for different applications.

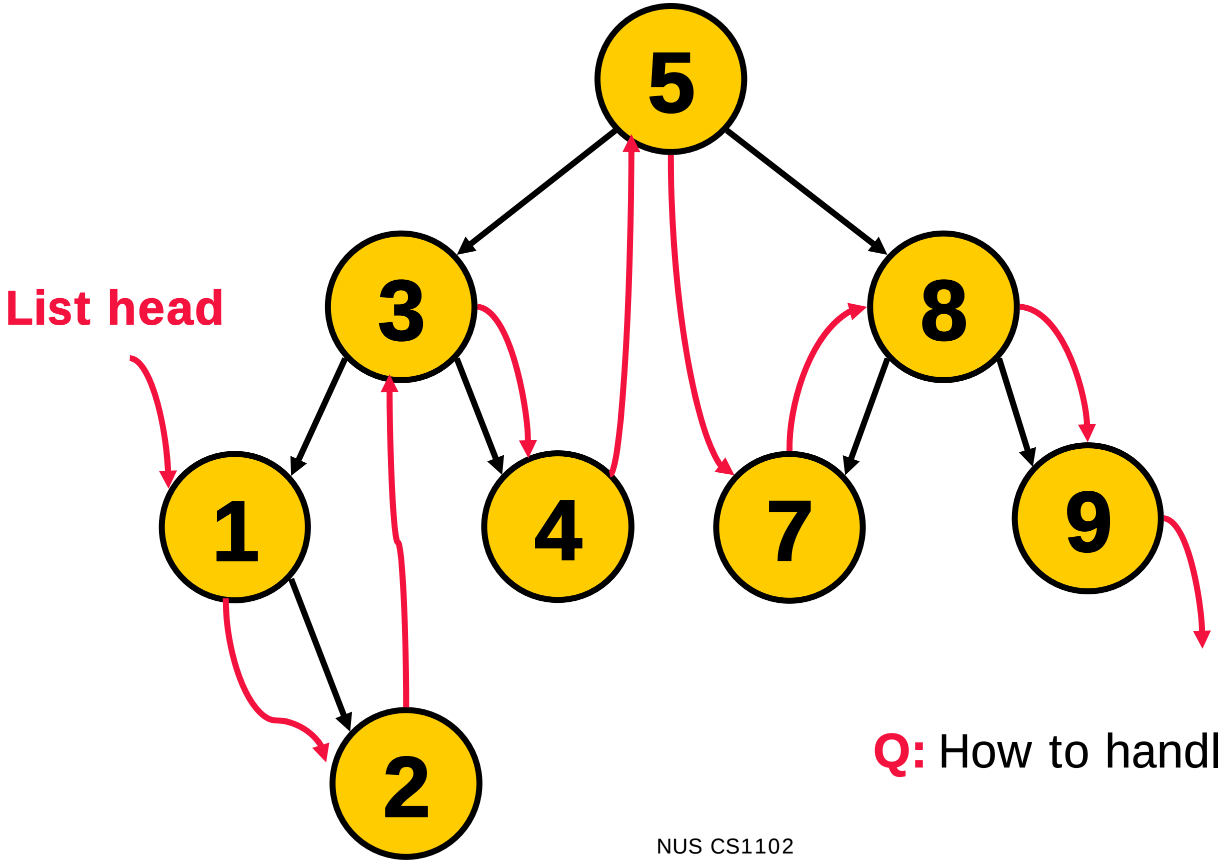
Mix-and-Match 1

- Array of Linked-Lists
  - E.g. Adjacent list for representing graph
  - E.g. Hash table with separate chaining



Mix-and-Match 2

- Binary Search Tree + Linked-List
- Can find the successors easily



Q: How to handle updates?

More Examples

- Suppose we need an ADT that support the following operations
  - enqueue(item)
  - dequeue()
  - peek()
  - printInOrder()

Use a Queue

- If we use a queue, we can support the queue operations efficiently O(1).
- But to print the items in order, we need to first sort the items in the queue, which is O(N log N) time.

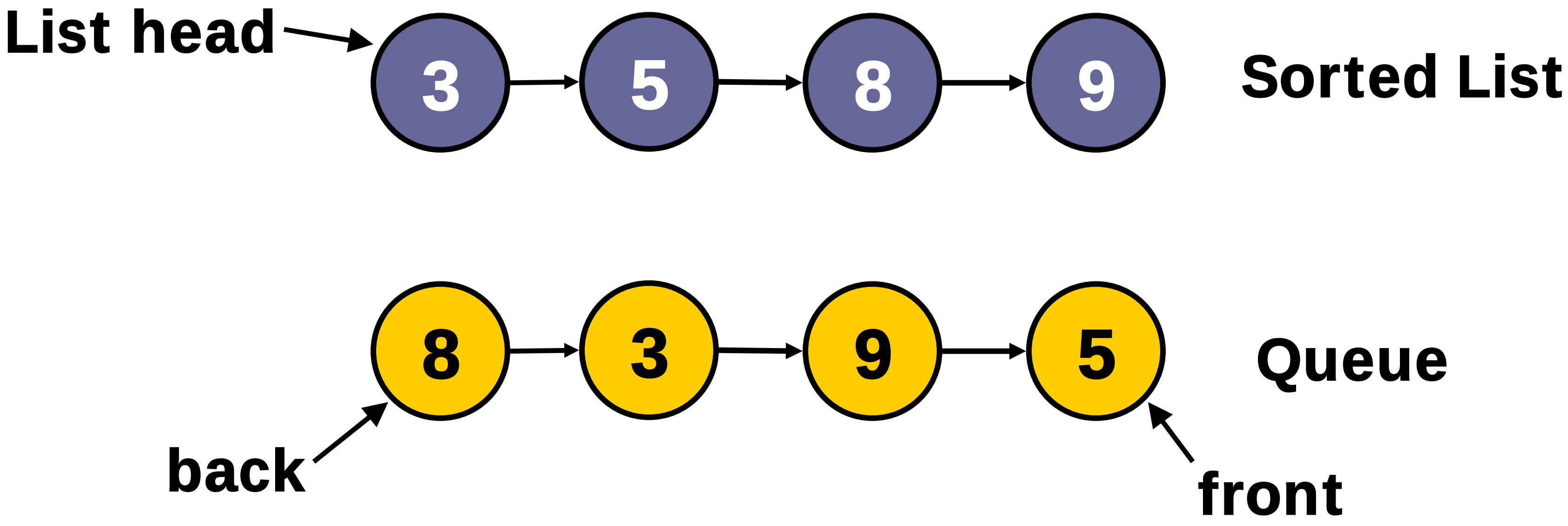
enqueue(item)	O(1)
dequeue()	O(1)
peek()	O(1)
printInOrder()	O(N log N)

Use a Sorted Linked List

- We can reduce printInOrder() to O(N) using a sorted linked list instead.
- But the queue operations are not supported

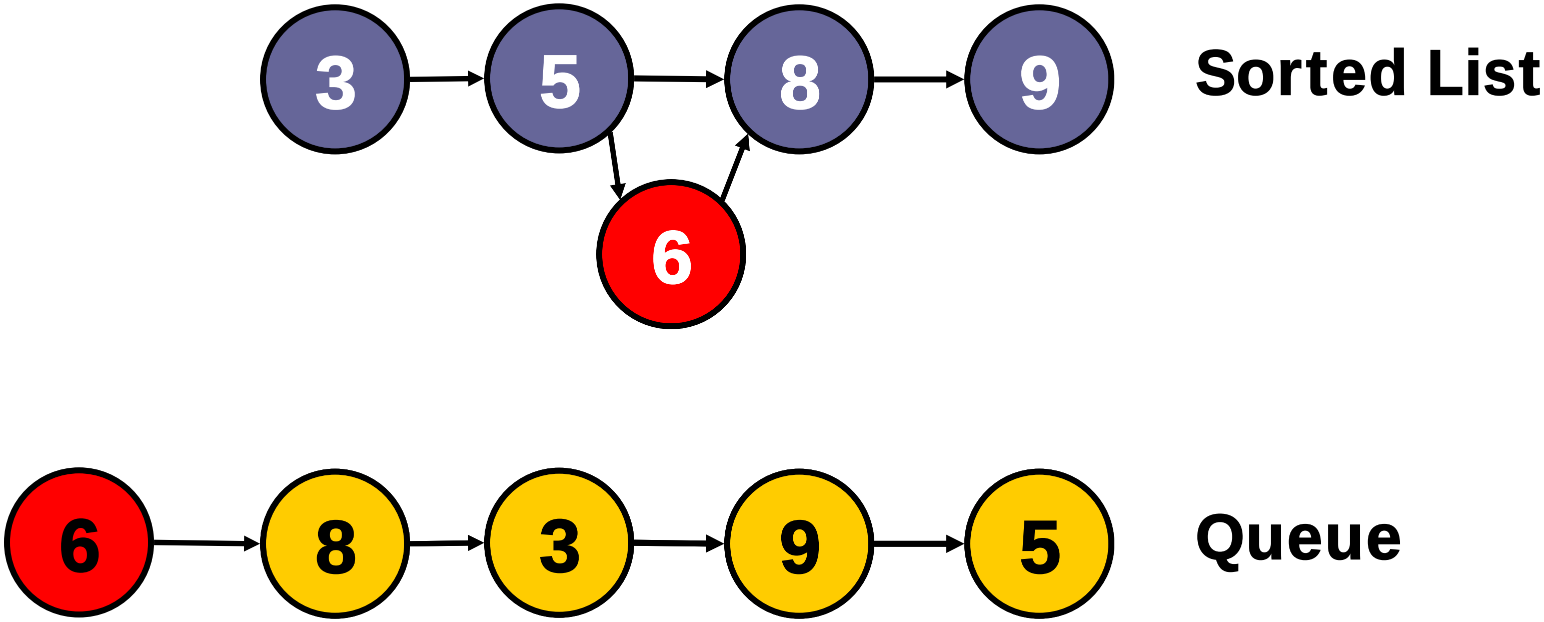
enqueue(item)	?
dequeue()	?
peek()	?
printInOrder()	O(N)

Use both: Queue + Sorted List ?

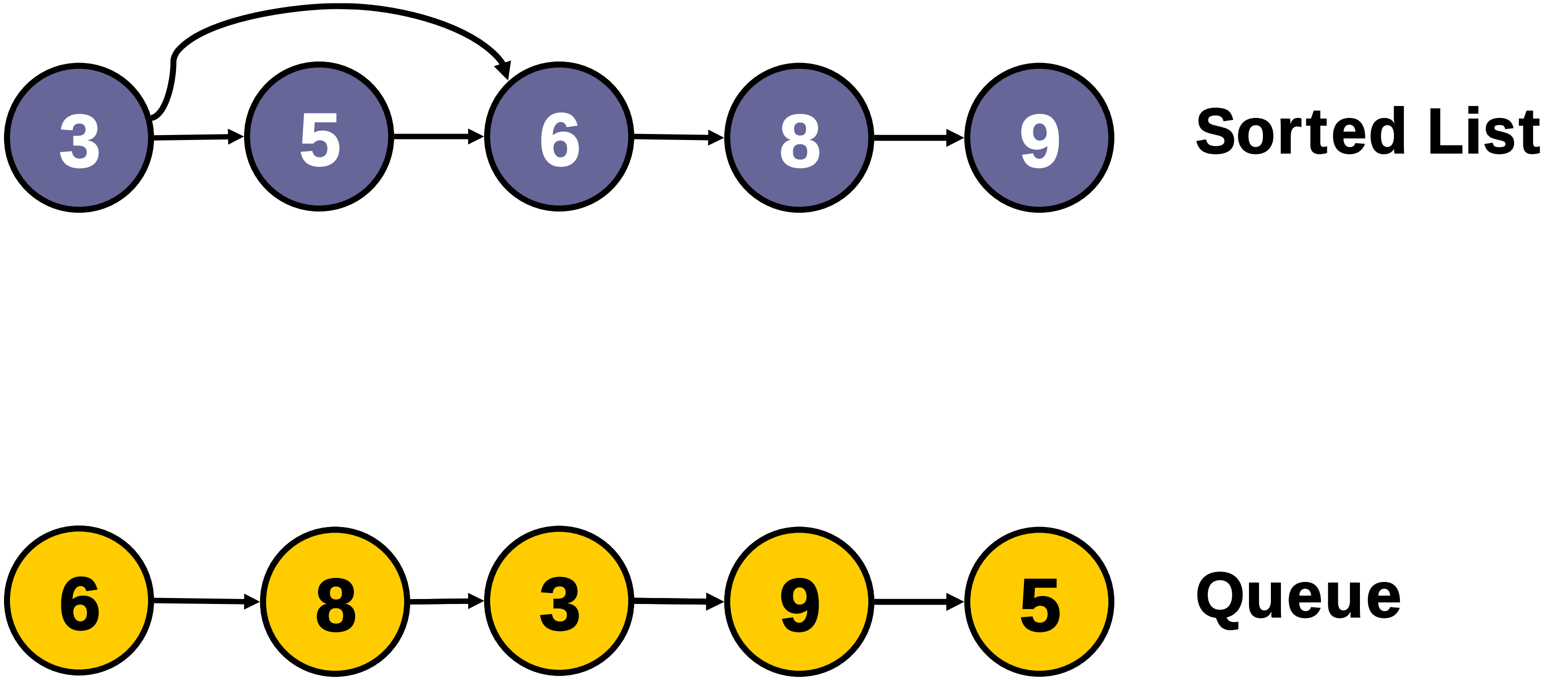


Trivial problem: Need to duplicate the data.

Enqueue(6)



Dequeue()



Use Queue + Sorted List

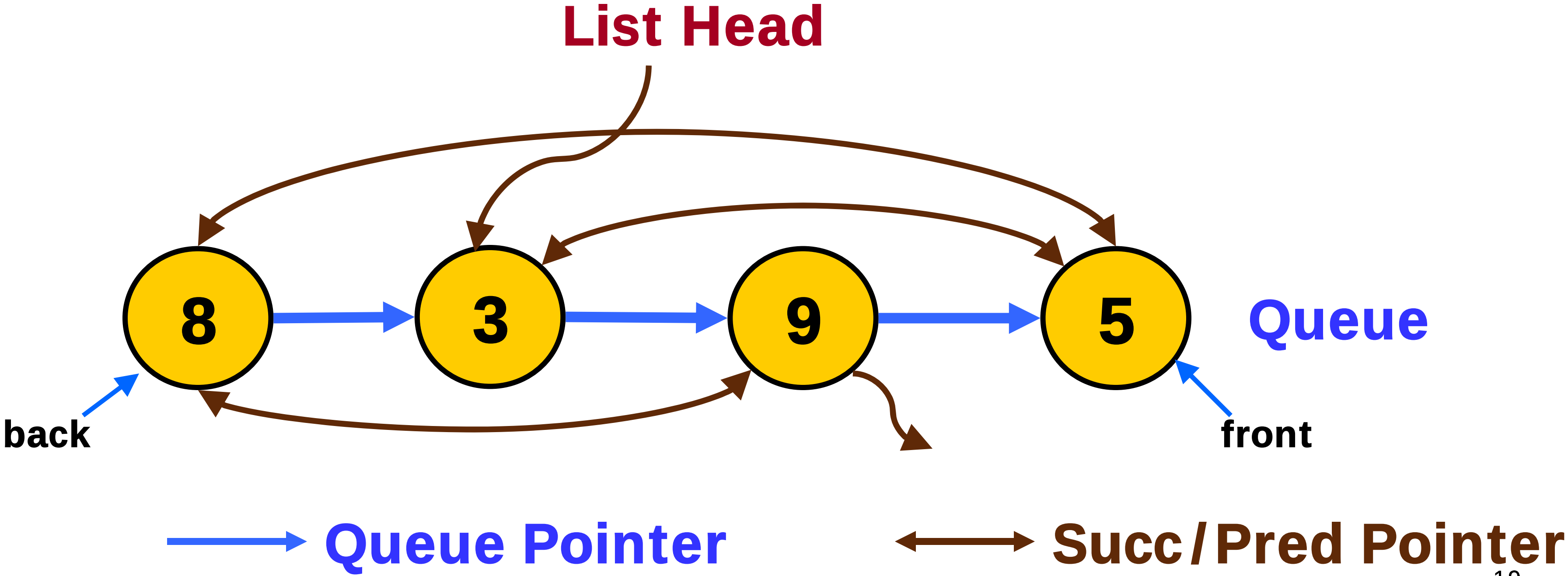
But then enqueue and dequeue take linear time O(N), because we have to look for the position of the item in the linked list to insert/delete. Too slow.

enqueue(item)	O(N)
dequeue()	O(N)
peek()	O(1)
printInOrder()	O(N)

Q: Can we improve them?

Improvement: Queue combines with DLinked List

- Only store one copy of each item
- Each node have 2 sets of pointers:
  - One for queue and One for a doubly linked list



Combine Queue and DLinked List

- Dequeue of a doubly linked list can be done in O(1) time.
  Q: How?
- However, enqueue is still O(N). Why? E.g. enqueue 4?

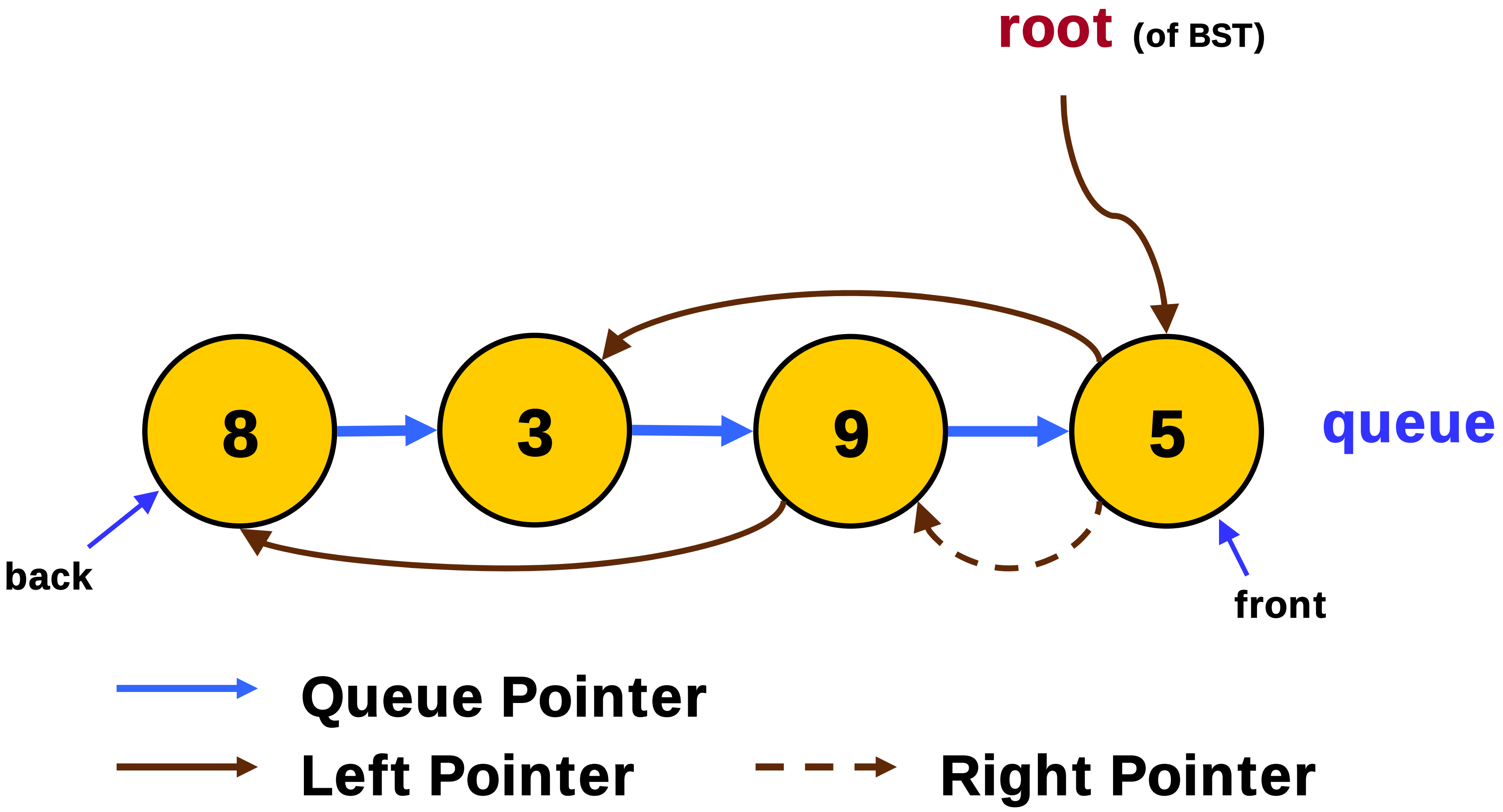
enqueue(item)	O(N)
dequeue()	O(1)
peek()	O(1)
printInOrder()	O(N)

Q: Can we improve it?

Combine Queue and BST

- We can improve enqueue to O(log N) by combing a queue with a BST instead of a linked list.

More improvement: Queue combines with BST



Combine Queue and BST

- But now dequeue also takes O(log N).

enqueue(item)	O(log N)
dequeue()	O(log N)
peek()	O(1)
printInOrder()	O(N)

Q: Is there a way to make dequeue O(1)?

Combine Queue and BST

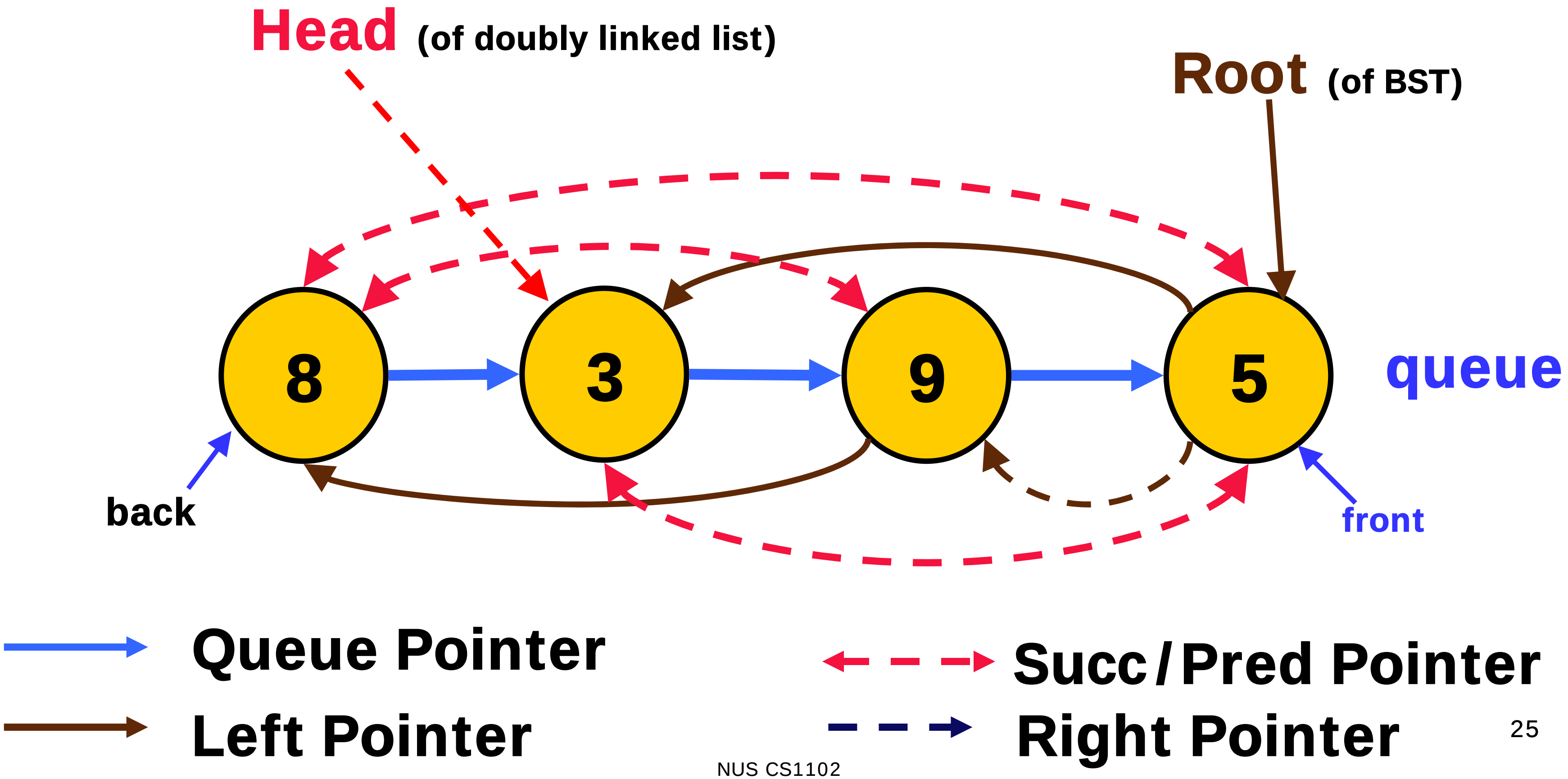
enqueue(item)	O(log N)
dequeue()	O(1) **?**
peek()	O(1)
printInOrder()	O(N)

Q: Is there a way to make dequeue O(1)?

Yes, use another doubly linked list, so that finding the replacement for BST deletion can be done in O(1) instead of O(log N).

More Improvement: combine Queue + BST + DList

- Use another doubly linked list.



Combine queue + BST + DList

enqueue(item)	O(log N)
dequeue()	O(1)
peek()	O(1)
printInOrder()	O(N)

Recall: use another doubly linked list, so that finding the replacement for BST deletions can be done in O(1) instead of O(log N). Why?

Improvement summary

- use a queue and a linked list
- combine queue with doubly linked list
- combine queue and BST
- combine queue, BST, and doubly linked list

Q: Which improvement should be used?

Depend on the application.

Consultation Hours

- **Dr. Tan Sun Teck**
  - Consultation hours:
    - No fixed consultation hours
    - Please call first and go to his office
    - Please attend his IVLE help sessions
  - Office: COM1-03-15

    Tel: 651 62778

Consultation Hours (cont.)

- **Prof. Tan Tiow Seng**
  - Consultation hours:
    - 16 April (Friday) 4-6pm
    - 19 April (Monday) 4:30-6pm
    - 20 April (Tuesday) 2-6pm
    - Other time, please call or email first
  - Office: AS6-04-10

    Tel: 651 66764

Consultation Hours (cont.)

- **Prof. Ling Tok Wang**
  - Consultation hours:
    - 14 April (Wednesday) 4:30-6pm
    - 16 April (Friday) 4:30-6pm
    - 21 April (Wednesday) 2-6pm
    - 23 April (Friday) 2-6pm
    - Other time, please call or email first
  - Office: COM1-03-14

    Tel: 651 62734