CSE 373: Queues

Chapter 3

Queue:

Queue Operations

Main Operations:

void Enqueue(Queue, QType);
QType Dequeue(Queue);
QType Front(Queue);
int IsEmpty(Queue);

Other Operations:

- normal creation/deletion operations
- again, generally no iteration operations
Queue Example

Queue Q;
int frontval, newval;

Q = NewQueue();
Enqueue(Q, 1);
Enqueue(Q, 1);
for (i=2; i<n; i++) {
    frontval = Dequeue(Q);
    newval = frontval + Front(Q);
    Enqueue(Q, newval);
}

List-based Queue Implementation

- As with Stacks, Queues are a specialized List
  - Push() = Insert() at a specific end of the list
  - Pop() = Delete() from the opposite end
- Thus, Lists could be used to implement the Queue ADT
  - Similar advantages and disadvantages as the Stack case
Array-Based Queue Implementation

Naive approach:
- **Enqueue ()** = insert at end of array
- **Dequeue ()** = delete from front of array

Running Time:
- **Enqueue ()**:
- **Dequeue ()**:

How could we improve this?

---

Link-Based Queue Implementation

What are the challenges to making a link-based
- **Enqueue ()** and **Dequeue ()** efficient?
Evaluating Queue Implementations

Operations:
- Enqueue ()
- Dequeue ()
- Front ()
- IsEmpty ()

Space:

Other:

List-based   Array-based   Link-Based

Applications of Queues

Anything where “fairness” (FIFO) is required
- operating systems: printer queues, storing user input, servers, scheduling processes
- compilers (and in general): worklists
- graphics: queue of things to render
- applications: list of recently used files
- real-life: lines at fast-food restaurants, “waiting for next available operator” lists
- searching: “breadth-first” searches