Event Lists

Refs: Sections 2.2 and 2.8 in Law,
Section 5.3 in Leemis and Park

Peter J. Haas

CS 590M: Simulation
Spring Semester 2020

Event Lists (aka Pending Event Sets)

Fetch-next, insert, and cancel operations

- Fundamental operations in discrete-event simulations (up to 40% of sim time)
- So far we have used clock-reading vectors
- For M events, it takes $O(M)$ time to get next event
- Unsuitable for large-scale simulation

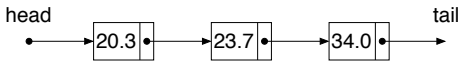
Alternative: event lists

- For GSMP's with unit speeds
- Idea: Maintain list of (event_type, event_time) pairs
  - event_time = (absolute) time when event is scheduled to occur
- Challenge: support operations efficiently (priority queue with removals)

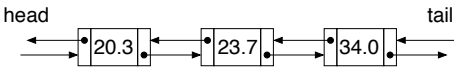
Linked Lists

Goal: Maintain events in sorted order

- Singly-linked lists

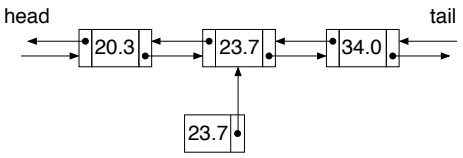


  - fetch-next is $O(1)$, insert and cancel are $O(M)$
- Doubly-linked lists



Linked Lists, Continued

- Indexed doubly-linked lists

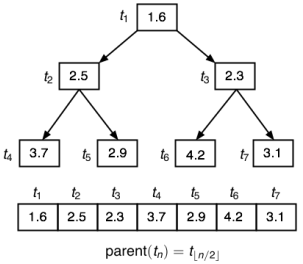


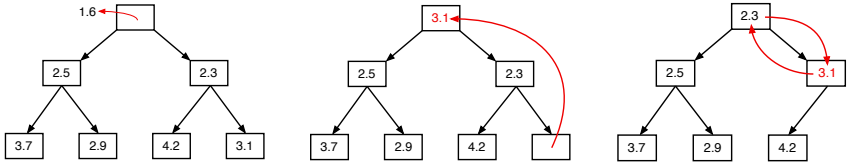
  - Faster lookup
  - Need to maintain median element
  - Cost outweighs benefit for more than one index

Implicit Binary Heaps

Binary tree that maintains
min-heap property

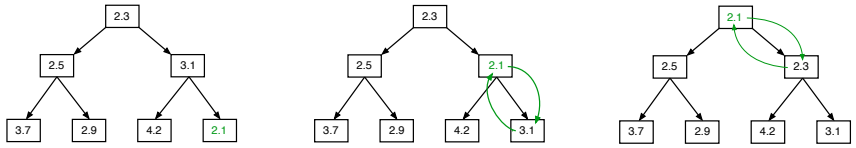
- Parent has smaller value than children
- Can store efficiently as an array
- Fetch-next is $O(1)$ plus an $O(\log M)$ update



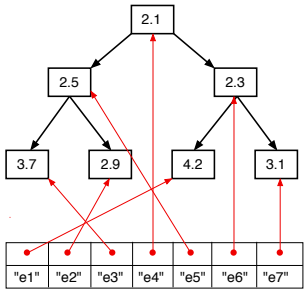


Heaps, Continued

- Insert is $O(\log M)$



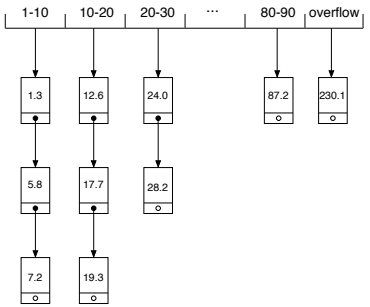
- Cancellation is $O(M)$ search + $O(\log M)$ update
- Python solution for $O(1)$ cancellation
  - Use heapq to implement heap
  - Use a dict for $O(1)$ find
  - Mark event as "canceled "and
  - Ignore cancelled events upon fetch
  - OK if not too many cancellations
  - See code on website



Hybrid Data Structures

Bucket System

- Event time "hashes" to a bucket
- Recycle buckets when they become empty



Henriksen's algorithm

- Used in many early commercial systems
- Combines binary search tree with doubly-linked list
- Can have bad worst-case behavior

Hybrid Data Structures, Continued

Lazy Queue [Ronngren et al. 1991]

- Three parts:
  - Near Future (NF): a sorted linked list
  - Far Future (FF): an unsorted bucket system
  - Very Far Future (VFF): an unsorted linked list

- Sorting only happens when FF bucket is moved to NF
- Occasional *adaptive* resizing of # and length of buckets
- Dominates most other event list schemes for > 50 events