Introduction to Simulation

- Simulation: Key Questions
- Introduction to Simulation
- Common Mistakes in Simulation
- Other Causes of Simulation Analysis Failure
- Checklist for Simulations
- Terminology
- Types of Models

Simulation: Key Questions

- What are the common mistakes in simulation and why most simulations fail?
- What language should be used for developing a simulation model?
- What are different types of simulations?
- How to schedule events in a simulation?
- How to verify and validate a model?
- How to determine that the simulation has reached a steady state?
- How long to run a simulation?

Simulation: Key Questions (Cont)

- How to generate uniform random numbers?
- How to verify that a given random number generator is good?
- How to select seeds for random number generators?
- How to generate random variables with a given distribution?
- What distributions should be used and when?

Introduction to Simulation

The best advice to those about to embark on a very large simulation is often the same as Punch's famous advice to those about to marry: Don't!

-Brately, Fox, and Schrage (1987)

Common Mistakes in Simulation

1. Inappropriate Level of Detail:
   More detail $\Rightarrow$ More time $\Rightarrow$ More Bugs $\Rightarrow$ More CPU $\Rightarrow$ More parameters $\neq$ More accurate
2. Improper Language
   General purpose $\Rightarrow$ More portable, More efficient, More time
3. Unverified Models: Bugs
4. Invalid Models: Model vs. reality
5. Improperly Handled Initial Conditions
6. Too Short Simulations: Need confidence intervals
7. Poor Random Number Generators: Safer to use a well-known generator
8. Improper Selection of Seeds: Zero seeds, Same seeds for all streams

Other Causes of Simulation Analysis Failure

1. Inadequate Time Estimate
2. No Achievable Goal
3. Incomplete Mix of Essential Skills
   (a) Project Leadership
   (b) Modeling and
   (c) Programming
   (d) Knowledge of the Modeled System
4. Inadequate Level of User Participation
5. Obsolete or Nonexistent Documentation
6. Inability to Manage the Development of a Large Complex Computer Program Need software engineering tools
7. Mysterious Results

Checklist for Simulations

1. Checks before developing a simulation:
   (a) Is the goal of the simulation properly specified?
   (b) Is the level of detail in the model appropriate for the goal?
   (c) Does the simulation team include personnel with project leadership, modeling, programming, and computer systems backgrounds?
   (d) Has sufficient time been planned for the project?
2. Checks during development:
   (a) Has the random number generator used in the simulation been tested for uniformity and independence?
   (b) Is the model reviewed regularly with the end user?
   (c) Is the model documented?

Checklist for Simulations (Cont)

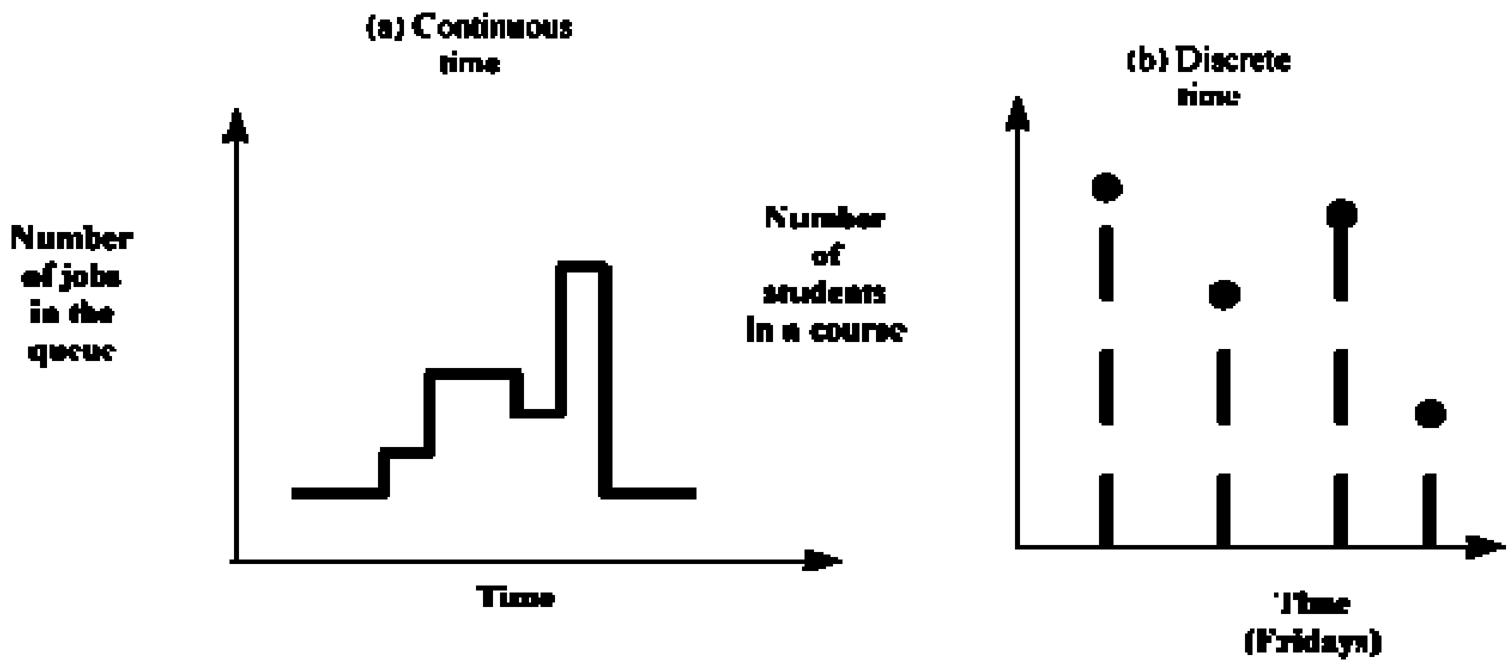
3. Checks after the simulation is running:
   (a) Is the simulation length appropriate?
   (b) Are the initial transients removed before computation?
   (c) Has the model been verified thoroughly?
   (d) Has the model been validated before using its results?
   (e) If there are any surprising results, have they been validated?
   (f) Are all seeds such that the random number streams will not overlap?

Terminology

- **State Variables**: Define the state of the system
  Can restart simulation from state variables
  E.g., length of the job queue.
- **Event**: Change in the system state.
  E.g., arrival, beginning of a new execution, departure

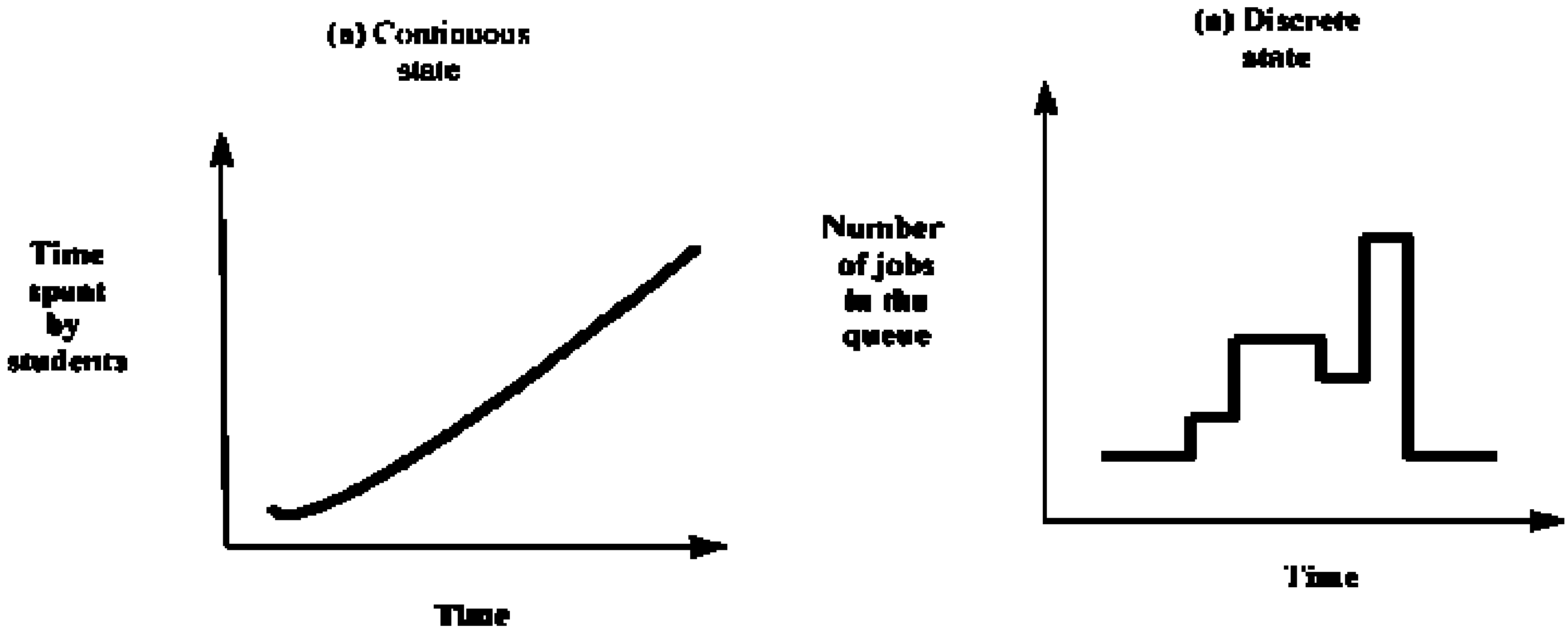
Types of Models

- **Continuous Time Model**: State is defined at all times
- **Discrete Time Models**: State is defined only at some instants



Types of Models (Cont)

- **Continuous State Model**: State variables are continuous
- **Discrete State Models**: State variables are discrete



Types of Models (Cont)

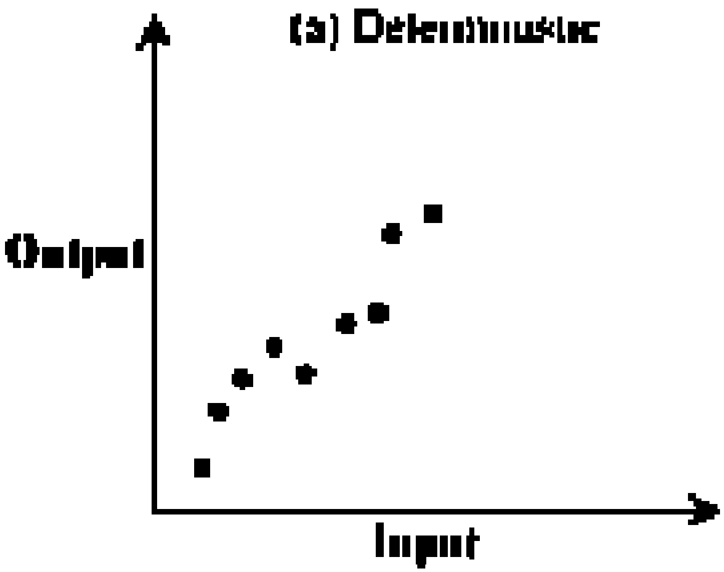
- Discrete state = Discrete event model
- Continuous state = Continuous event model
- Continuity of time ≠ Continuity of state

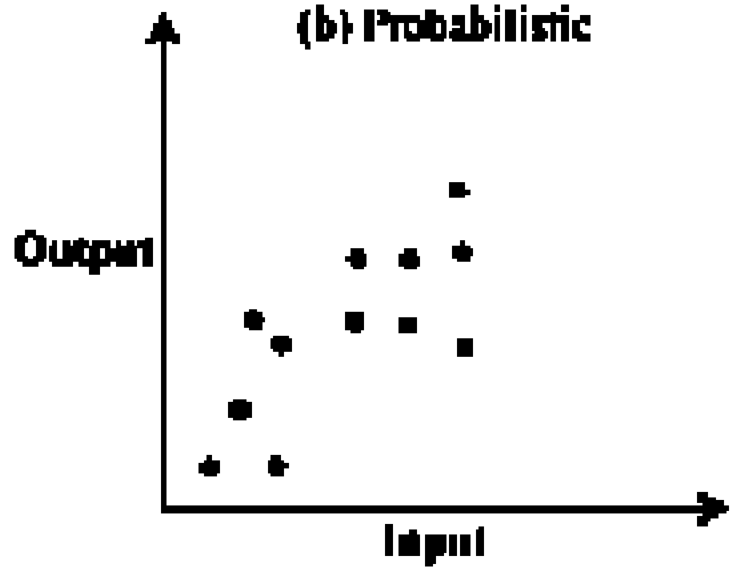
- Four possible combinations:

1. discrete state/discrete time
2. discrete state/continuous time
3. continuous state/discrete time
4. continuous state/continuous time models

Types of Models (Cont)

- **Deterministic and Probabilistic Models**:



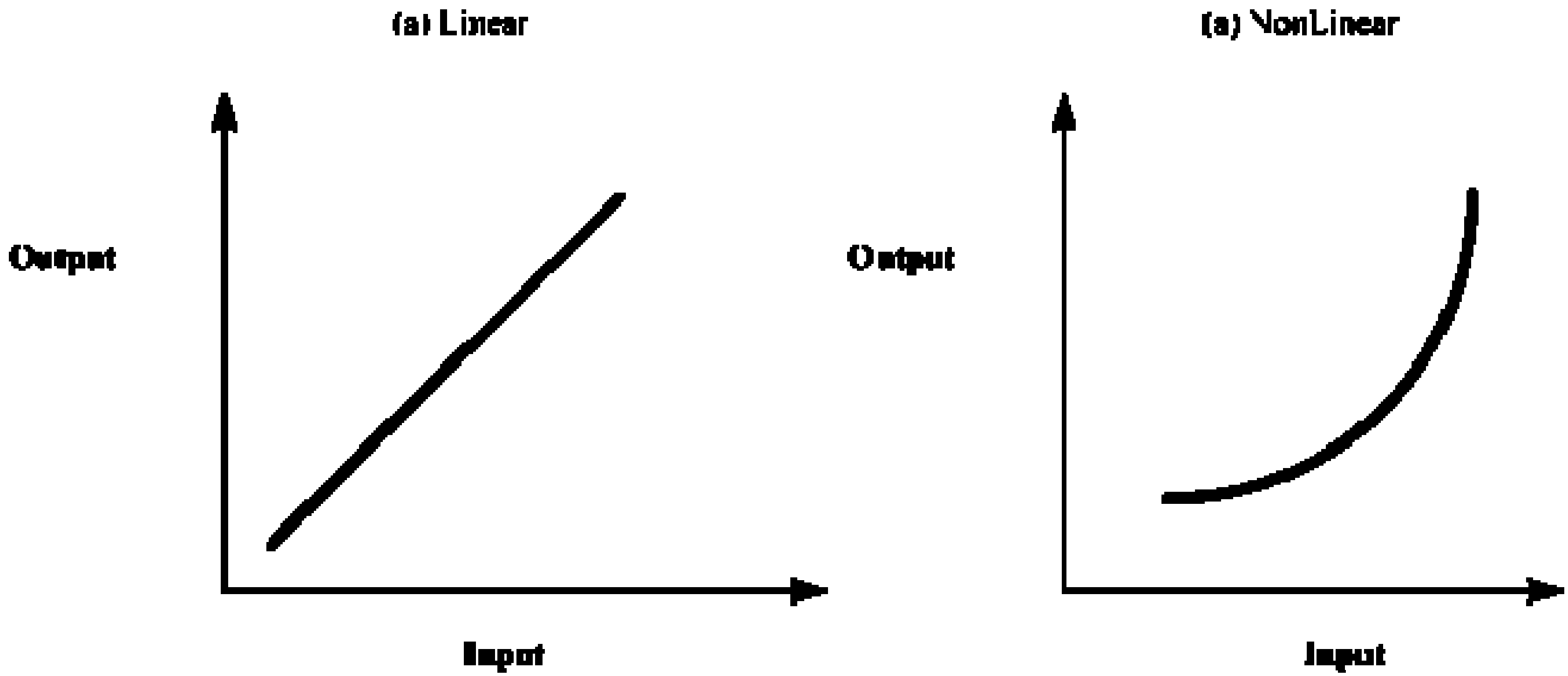


- **Static and Dynamic Models**:

CPU scheduling model vs. $E = mc^2$.

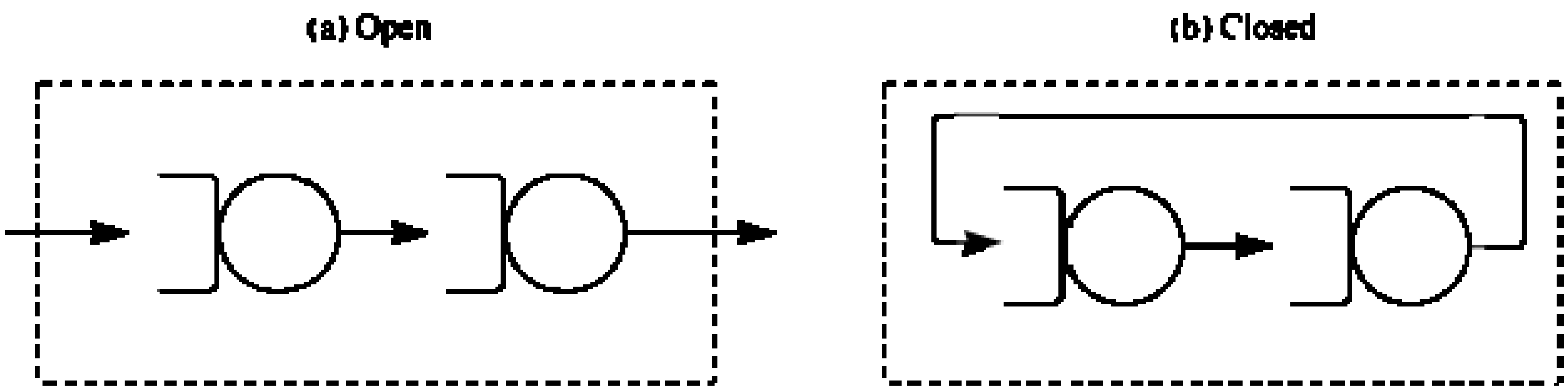
Linear and Nonlinear Models

- Output = fn(Input)



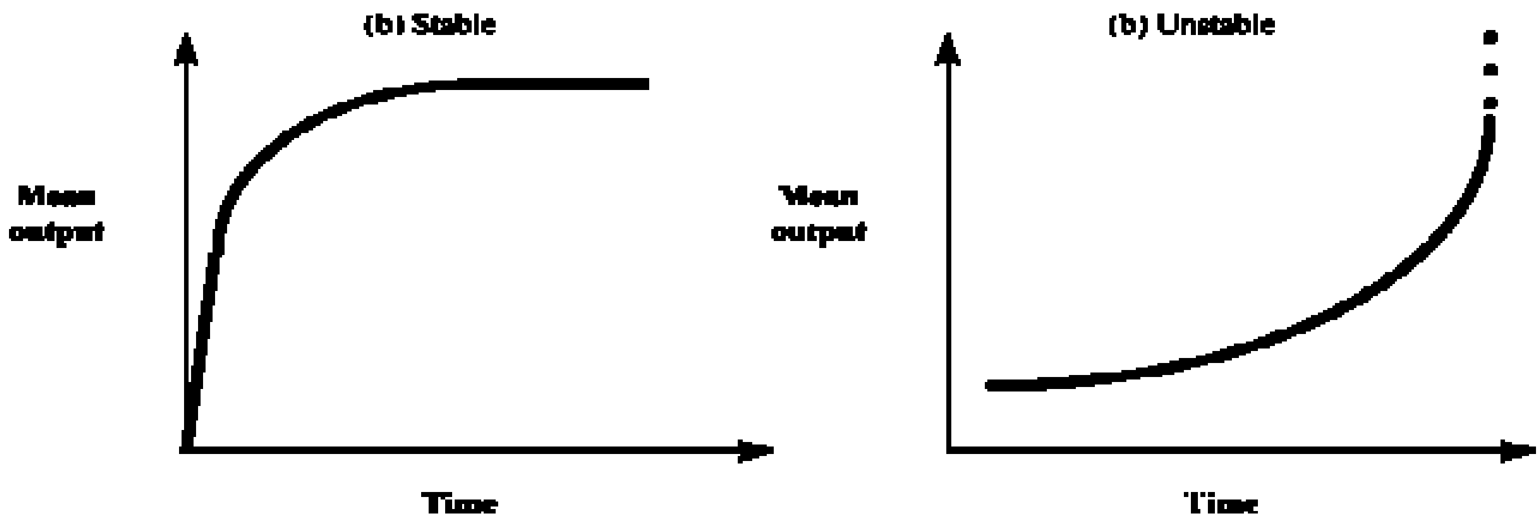
Open and Closed Models

- External input ⇒ open



Stable and Unstable Models

- Stable ⇒ Settles to steady state
- Unstable ⇒ Continuously changing.



Computer System Models

- Continuous time
- Discrete state
- Probabilistic
- Dynamic
- Nonlinear
- Open or closed
- Stable or unstable

Selecting a Language for Simulation

1. Simulation language
2. General purpose
3. Extension of a general purpose language
4. Simulation package

Simulation Languages

- Save development time
- Built-in facilities for time advancing, event scheduling, entity manipulation, random variate generation, statistical data collection, and report generation
- More time for system specific issues
- Very readable modular code

General Purpose Language

- Analyst's familiarity
- Easy availability
- Quick startup
- Time for routines for event handling, random number generation
- Other Issues: Efficiency, flexibility, and portability
- Recommendation: Learn at least one simulation language.

Extensions of a General Purpose Language

- Examples: GASP (for FORTRAN)
  - Collection of routines to handle simulation tasks
  - Compromise for efficiency, flexibility, and portability.

Simulation Packages

Example: QNET4, and RESQ

- Input dialog
- Library of data structures, routines, and algorithms
- Big time savings
- Inflexible ⇒ Simplification

Types of Simulation Languages

- **Continuous Simulation Languages**:
  - CSMP, DYNAMO
  - Differential equations
  - Used in chemical engineering
- **Discrete-event Simulation Languages**:
  - SIMULA and GPSS
- **Combined**:
  - SIMSCRIPT and GASP.
  - Allow discrete, continuous, as well as combined simulations.

Types of Simulations

1. Emulation: Using hardware or firmware
   E.g., Terminal emulator, processor emulator
   Mostly hardware design issues
2. Monte Carlo Simulation
3. Trace-Driven Simulation
4. Discrete Event Simulation

Types of Simulation (Cont)

Monte Carlo method *[Origin: after Count Montgomery de Carlo, Italian gambler and random-number generator (1792-1838).] A method of jazzing up the action in certain statistical and number-analytic environments by setting up a book and inviting bets on the outcome of a computation.*

- The Devil's DP Dictionary

McGraw Hill (1981)

Monte Carlo Simulation

- Static simulation (No time axis)
- To model probabilistic phenomenon
- Need pseudorandom numbers
- Used for evaluating non-probabilistic expressions using probabilistic methods.

Monte Carlo: Example

$$I = \int_0^2 e^{-x^2} dx$$

$$x \sim \text{Uniform}(0, 2)$$

Density function f(x) = $\frac{1}{2}$ iff $0 \leq x \leq 2$

$$y = 2e^{-x^2}$$

Monte Carlo: Example (Cont)

$$
\begin{aligned}
E(y) &= \int_0^2 2e^{-x^2} f(x)dx \\
&= \int_0^2 2e^{-x^2} \frac{1}{2} dx \\
&= \int_0^2 e^{-x^2} dx \\
&= I
\end{aligned}
$$

$$x_i \sim \text{Uniform}(0,2)$$

$$y_i = 2e^{-x_i^2}$$

$$I = E(y) = \frac{1}{n}\sum_{i=1}^{n} y_i$$

Trace-Driven Simulation

- Trace = Time ordered record of events on a system
- Trace-driven simulation = Trace input
- Used in analyzing or tuning resource management algorithms
  Paging, cache analysis, CPU scheduling, deadlock prevention
  dynamic storage allocation
- **Example**: Trace = Page reference patterns
- Should be independent of the system under study
  E.g., trace of pages fetched depends upon the working set size and page replacement policy
  - Not good for studying other page replacement policies
  - Better to use pages referenced

Advantages of Trace-Driven Simulations

1. Credibility
2. Easy Validation: Compare simulation with measured
3. Accurate Workload: Models correlation and interference
4. Detailed Trade-Offs:
   Detailed workload ⇒ Can study small changes in algorithms
5. Less Randomness:
   Trace ⇒ deterministic input ⇒ Fewer repetitions
6. Fair Comparison: Better than random input
7. Similarity to the Actual Implementation:
   Trace-driven model is similar to the system
   ⇒ Can understand complexity of implementation

Disadvantages of Trace-Driven Simulations

1. Complexity: More detailed
2. Representativeness: Workload changes with time, equipment
3. Finiteness: Few minutes fill up a disk
4. Single Point of Validation: One trace = one point
5. Detail
6. Trade-Off: Difficult to change workload

Discrete Event Simulations

- Concentration of a chemical substance
  $\Rightarrow$ **C**ontinuous event simulations
- Number of jobs $\Rightarrow$ Discrete event
- Discrete state $\neq$ discrete time

Components of Discrete Event Simulations

1. Event Scheduler
   (a) Schedule event X at time T.
   (b) Hold event X for a time interval dt.
   (c) Cancel a previously scheduled event X.
   (d) Hold event X indefinitely
   (e) Schedule an indefinitely held event.

2. Simulation Clock and a Time Advancing Mechanism
   (a) Unit-time approach
   (b) Event-driven approach

Components of Discrete Events Sims (Cont)

3. System State Variables

   Global = Number of jobs

   Local = CPU time required for a job

4. Event Routines: One per event.

   E.g., job arrivals, job scheduling, and job departure

5. Input Routines: Get model parameters Very parameters in a range.

6. Report Generator

7. Initialization Routines: Set the initial state. Initialize seeds.

8. Trace Routines: On/off feature

9. Dynamic Memory Management: Garbage collection

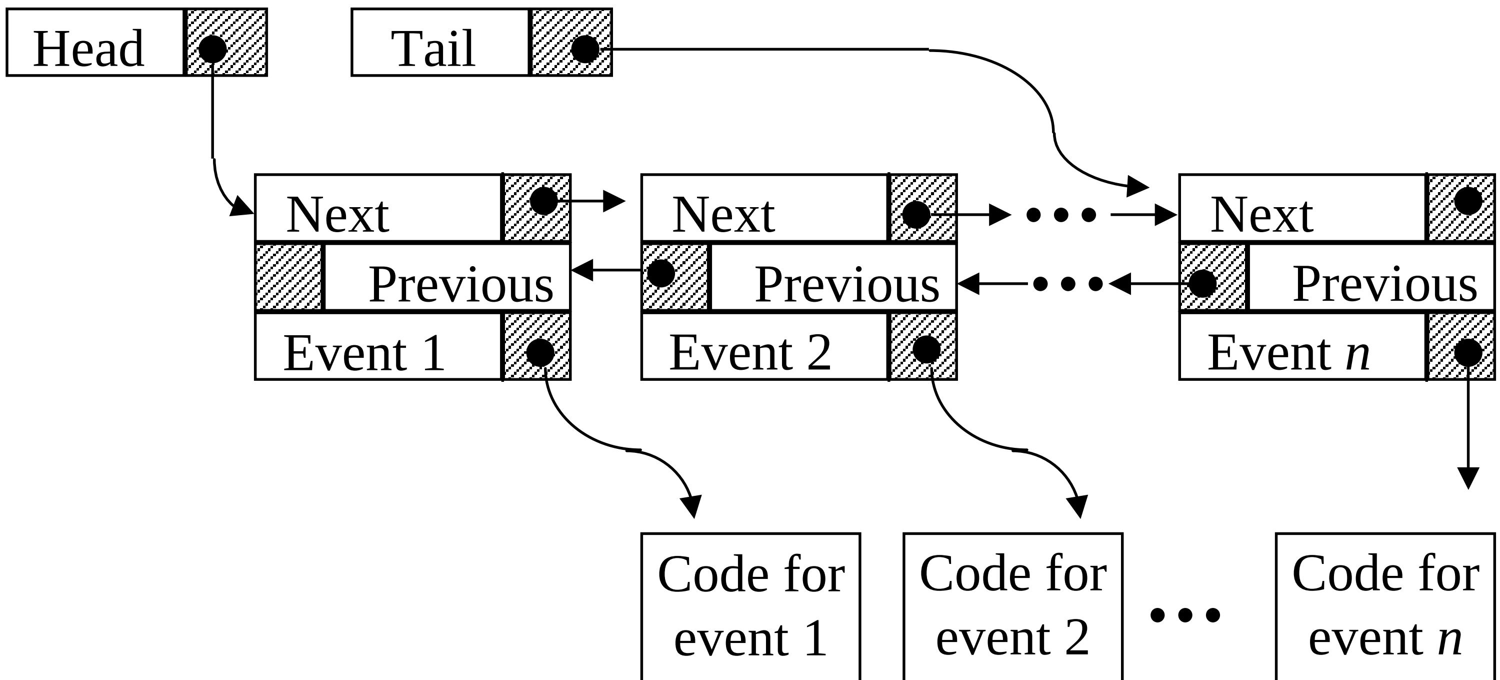
10. Main Program

Event-Set Algorithms

Event Set = Ordered linked list of future event notices

Insert vs. Execute next

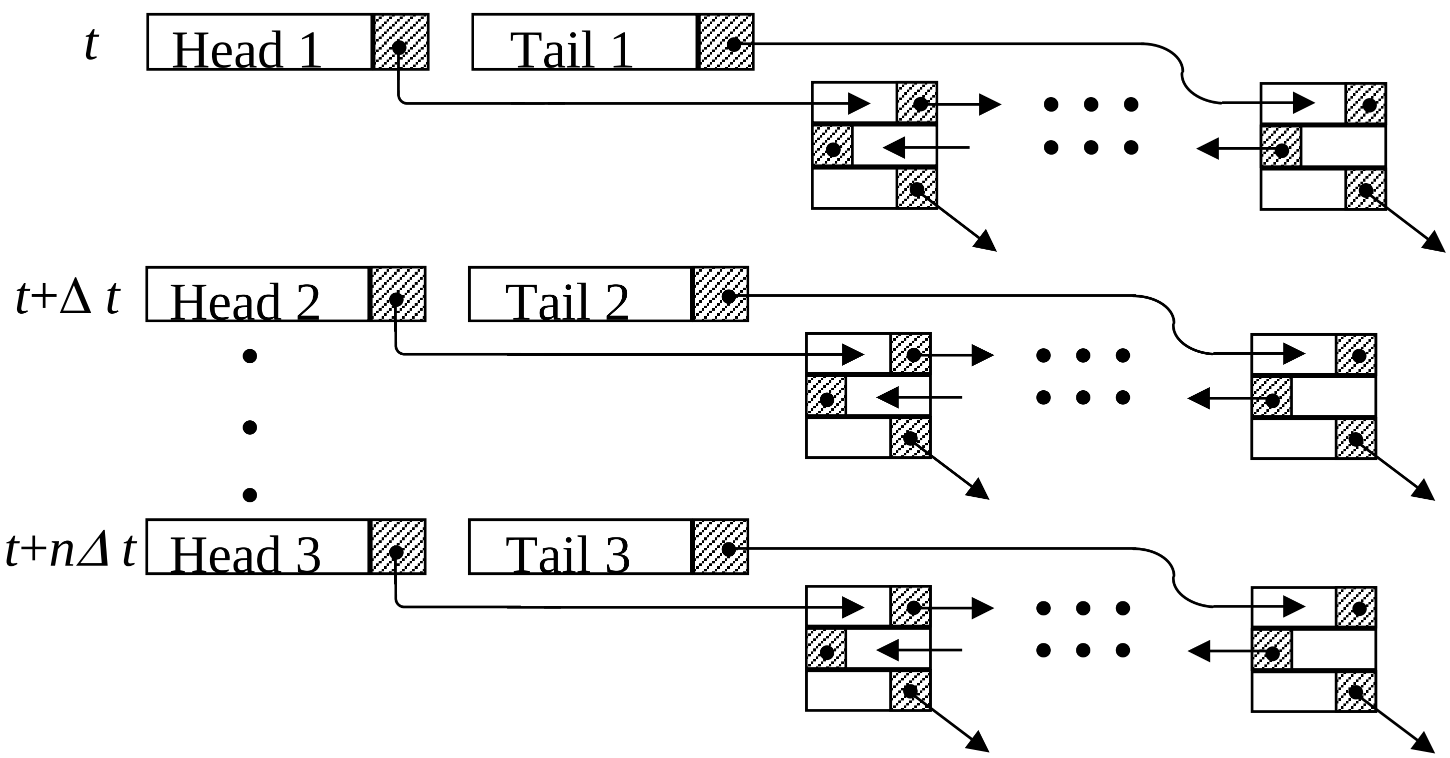
1. **Ordered Linked List**: SIMULA, GPSS, and GASP IV



Search from left or from right

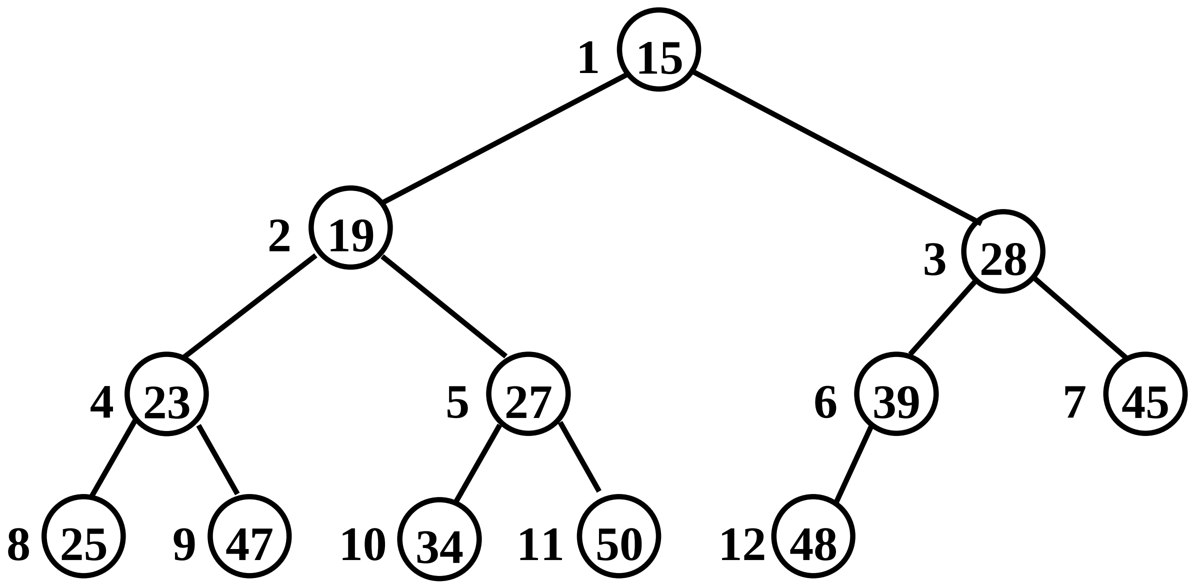
Event-Set Algorithms (Cont)

2. **Indexed Linear List**:
   - Array of indexes ⇒ No search to find the sub-list
   - Fixed or variable Δt. Only the first list is kept sorted



Event-Set Algorithms (Cont)

3. **Calendar Queues**: All events of Jan 1 on one page. 1995 or 1996.
4. **Tree Structures**: Binary tree $\Rightarrow \log_2 n$
5. **Heap**: Event is a node in binary tree



(a) Tree representation of a heap.

Event-Set Algorithms(Cont)

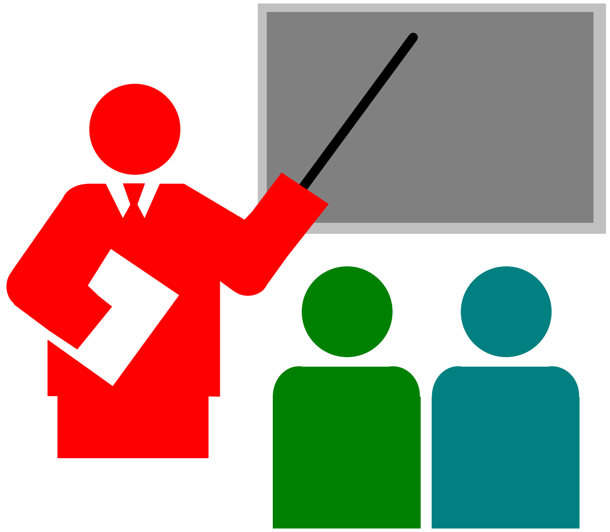
i	1	2	3	4	5	6	7	8	9	10	11	12
A[*i*]	**15**	**19**	**28**	**23**	**27**	**39**	**45**	**25**	**47**	**34**	**50**	**48**

- Event time for each node is smaller than that of its Children
  $\Rightarrow$ **R**oot is next
- Heap can be stored as arrays
- Children of node in position *i* are in positions *2i* and *2i+1*

6. ***k*-ary heaps**: *k*-ary trees

- 20-120 events: Index linear
- 120+ events: Heaps

Summary



1. Common Mistakes: Detail, Invalid, Short
2. Discrete Event, Continuous time, nonlinear models
3. Monte Carlo Simulation: Static models
4. Trace driven simulation: Credibility, difficult trade-offs
5. Even Set Algorithms: Linked list, indexed linear list, heaps

Exercise 24.1

For each of the following models, identify all classifications that apply to it:

a. $y(t)=t+0.2$

b. $y(t)=t^2$

c. $y(t+1)=y(t)+\Delta$, Δ is not an integer.

d. $n(t+1)=2n(t)+3$

e. $y(t)=\sin(wt)$

f. $\bar{y}(t+1) = \bar{y}(t) + \Delta$

Exercise 24.2

Which type of simulation would you use for the following problems:

1. To model destination address reference patterns in a network traffic, given that the pattern depends upon a large number of factors.
2. To model scheduling in a multiprocessor system, given that the request arrivals have a known distribution.
3. To determine the value of π

Exercise 24.3

What is unit-time approach and why is it not generally used?

Homework

For each of the following models, identify all classifications that apply to it:

1. $\bar{y}(t+1) = \bar{y}(t) + a$
2. $y(t+1) = y(t) + 3$
3. $y(t) = t^1.5$
4. $y(t) = a + bt + ct^2$
5. $n(t+1) = 3n(t) + 5$
6. $y(t) = cos(wt + \psi)$