

Computer Systems Performance Analysis and Benchmarking (37-235)

Analytic Modeling Simulation

Measurements / Benchmarking

Lecture/Assignments/Projects:
Dr. Christian Kurmann

Textbook:

Raj Jain, "The Art of Computer Systems Performance Analysis", 1991 Wiley & Sons, New York

Topic of Today:

- Proof of SST=SSA+SSB+SSAB
- Two factor full factorial design (without replications)
- Introduction to Simulation

Proof for Allocation of Variation

$$SST = 2^2 q_A^2 + 2^2 q_B^2 + 2^2 q_{AB}^2$$

$$SST = SSA + SSB + SSAB$$

$$\text{Fraction of variation explained by } A = \frac{SSA}{SST}$$

TABLE 17.3 Sign Table Method of Calculating Effects in a 2² Design

I	A	B	AB	y
1	-1	-1	1	15
1	1	-1	-1	45
1	-1	1	-1	25
1	1	1	1	75
160	80	40	20	Total
40	20	10	5	Total/4

The model used in a 2² design is

$$y_i = q_0 + q_A x_{Ai} + q_B x_{Bi} + q_{AB} x_{Ai} x_{Bi}$$

Columns x_A , x_B , and x_{AB} of the design matrix in Table 17.3 have the following properties:

$$\sum_{i=1}^4 x_{Ai} = 0; \quad \sum_{i=1}^4 x_{Bi} = 0; \quad \sum_{i=1}^4 x_{Ai} x_{Bi} = 0$$

2. The sum of the squares of entries in each column is 4:

$$\sum_{i=1}^4 x_{Ai}^2 = 4; \quad \sum_{i=1}^4 x_{Bi}^2 = 4; \quad \sum_{i=1}^4 (x_{Ai} x_{Bi})^2 = 4$$

3. The columns are orthogonal since the inner product of any two columns is zero:

$$\sum_{i=1}^4 x_{Ai} x_{Bi} = 0; \quad \sum_{i=1}^4 x_{Ai} (x_{Ai} x_{Bi}) = 0 = \sum_{i=1}^4 x_{Bi} (x_{Ai} x_{Bi})$$

Proof ...

These properties allow us to compute the total variation

$$\begin{aligned} \text{Sample mean } \bar{y} &= \frac{1}{4} \sum_{i=1}^4 y_i \\ &= \frac{1}{4} \sum_{i=1}^4 (q_0 + q_A x_{Ai} + q_B x_{Bi} + q_{AB} x_{Ai} x_{Bi}) \\ &= \frac{1}{4} \sum_{i=1}^4 q_0 + \frac{1}{4} q_A \sum_{i=1}^4 x_{Ai} + q_B \frac{1}{4} \sum_{i=1}^4 x_{Bi} + q_{AB} \frac{1}{4} \sum_{i=1}^4 x_{Ai} x_{Bi} \\ &= q_0 \end{aligned}$$

and

$$\begin{aligned} \text{Total variation} &= \sum_{i=1}^4 (y_i - \bar{y})^2 \\ &= \sum_{i=1}^4 (q_A x_{Ai} + q_B x_{Bi} + q_{AB} x_{Ai} x_{Bi})^2 \\ &= \sum_{i=1}^4 (q_A x_{Ai})^2 + \sum_{i=1}^4 (q_B x_{Bi})^2 + \sum_{i=1}^4 (q_{AB} x_{Ai} x_{Bi})^2 \\ &\quad + \text{product terms} \\ &= q_A^2 \sum_{i=1}^4 (x_{Ai})^2 + q_B^2 \sum_{i=1}^4 (x_{Bi})^2 + q_{AB}^2 \sum_{i=1}^4 (x_{Ai} x_{Bi})^2 + 0 \\ &= 4q_A^2 + 4q_B^2 + 4q_{AB}^2 \end{aligned}$$

Two Factor Experiments

- Used when two parameters that are carefully controlled and varied to study their impact on the performance.
- E.g. compare several processors using several workloads
- Full factorial design with two factors A and B having a and b levels requires ab experiments.
- Concepts are extensions of those on one factor designs.

Statistical model

The model for a two-factor design without replications is

$$y_{ij} = \mu + \alpha_j + \beta_i + e_{ij}$$

Here, y_{ij} is the observation in the experiment with the first factor A being at level j and the second factor B being at level i , μ is the mean response, α_j is the effect of factor A at level j , β_i is the effect of factor B at level i , and e_{ij} is the error term. The effects α_j and β_i are computed so that their sums are zero:

$$\sum \alpha_j = 0$$

$$\sum \beta_i = 0$$

Computation of the effects

The observations are assumed to be arranged in a two-dimensional matrix of b rows and a columns such that: the (i, j) th entry y_{ij} corresponds to the response in the experiment in which factor A is at level j and factor B is at level i . In other words, the columns correspond to the levels of A and rows correspond to levels of the factor B .

The values of model parameters μ , α_j 's, and β_i 's are computed such that the error has a zero mean. This means that the sum of error terms along each column and along each row is zero.

Averaging the j th column produces

$$\bar{y}_{.j} = \mu + \alpha_j + \frac{1}{b} \sum_i \beta_i + \frac{1}{b} \sum_i e_{ij}$$

Since the last two terms are zero, we have

$$\bar{y}_{.j} = \mu + \alpha_j$$

Similarly, averaging along rows produces

$$\bar{y}_{i.} = \mu + \beta_i$$

Averaging all observations produces

$$\bar{y}_{..} = \mu$$

Thus, the estimates of the model parameters are

$$\mu = \bar{y}_{..}$$

$$\alpha_j = \bar{y}_{.j} - \bar{y}_{..}$$

$$\beta_i = \bar{y}_{i.} - \bar{y}_{..}$$

Example:

Example 21.1 In a study to compare three different cache choices, the processor times to execute five different workloads was measured on three different configurations of a processor. The three configurations differed only in their cache designs. The three cache options were to use two set associative caches, one set associative cache, or no cache. The measured times in milliseconds are shown in Table 21.1. The analysis to compute the effects of the caches and workloads is shown in Table 21.2. For each row (or column), we compute the mean of observations in that row (or column). Overall sum and means are also computed. The difference between a row (or column) mean and overall mean gives the row (or column) effect.

The results of the analysis are interpreted as follows. An average workload on an average processor requires 72.2 milliseconds of processor time.

TABLE 21.1 Measured Processor Times for the Cache Comparison Study

Workload	Two Caches	One Cache	No Cache
ASM	54.0	55.0	106.0
TECO	60.0	60.0	123.0
SIEVE	43.0	43.0	120.0
DHRYSTONE	49.0	52.0	111.0
SORT	49.0	50.0	108.0

TABLE 21.2 Computation of Effects for the Cache Comparison Study

Workload	Two Caches	One Cache	No Cache	Row Sum	Row Mean	Row Effect
ASM	54.0	55.0	106.0	215.0	71.7	-0.5
TECO	60.0	60.0	123.0	243.0	81.0	8.8
SIEVE	43.0	43.0	120.0	206.0	68.7	-3.5
DHRYSTONE	49.0	52.0	111.0	212.0	70.7	-1.5
SORT	49.0	50.0	108.0	207.0	69.0	-3.2
Column sum	255.0	260.0	568.0	1083.0		
Column mean	51.0	52.0	113.6		72.2	
Column effect	-21.2	-20.2	41.4			

The time with two caches is 21.2 milliseconds lower than that on an average processor, and the time with one cache is 20.2 milliseconds lower than that on an average processor. The time without a cache is 41.4 milliseconds higher than the average. This is equivalent to saying that the mean difference between a two-cache processor and a one-cache processor is 1

millisecond. Similarly, the difference between a one-cache processor and a no-cache processor is $41.4 - 20.2$, or 21.2, milliseconds.

The workloads also affect the processor time required. An average workload on an average processor takes 72.2 milliseconds. The ASM workload takes 0.5 millisecond less than the average. TECO takes 8.8 milliseconds higher than the average, and so on. □

Errors

Having computed the model parameters, the estimated response in the (i, j) th experiment is given by

$$\hat{y}_{ij} = \mu + \alpha_j + \beta_i$$

The difference between the estimated response and the measured response y_{ij} is attributed to experimental errors. In other words,

$$e_{ij} = y_{ij} - \hat{y}_{ij} = y_{ij} - \mu - \alpha_j - \beta_i$$

The variance of errors can be estimated from the Sum of Squared Errors (SSE):

$$SSE = \sum_{i=1}^b \sum_{j=1}^a e_{ij}^2$$

Example 21.2 For the cache comparison study of Example 21.1, the errors in each of the 15 observations are shown in Table 21.3. To see how the entries are computed, consider the first experiment (with ASM workload on a two-cache processor). The estimated processor time is

$$\hat{y}_{11} = \mu + \alpha_1 + \beta_1 = 72.2 - 21.2 - 0.5 = 50.5$$

TABLE 21.3 Error Computation for the Cache Comparison Study

Workload	Two Caches	One Cache	No Cache
ASM	3.5	3.5	-7.1
TECO	0.2	-0.8	0.6
SIEVE	-4.5	-5.5	9.9
DHRYSTONE	-0.5	1.5	-1.1
SORT	1.2	1.2	-2.4

The measured processor time is 54 milliseconds. The difference $54 - 50.5 = 3.5$ is the error. The sum of squared errors is

$$SSE = (3.5)^2 + (0.2)^2 + \dots + (-2.4)^2 = 2368.00 \quad \square$$

Allocation of Variation

As in the case of one-factor designs and in 2^2r designs, the total variation of y in a two-factor design can be allocated to the two factors and to the experimental errors. To do so, we square both sides of the model equation and add across all observations. The cross-product terms cancel, and we obtain

$$\sum_{ij} y_{ij}^2 = ab\mu^2 + b \sum_j \alpha_j^2 + a \sum_i \beta_i^2 + \sum_{ij} e_{ij}^2$$

$$SSY = SS0 + SSA + SSB + SSE$$

where various sums of squares have been appropriately placed below their corresponding terms. The total variation (SST), as before, is

$$SST = SSY - SS0 = SSA + SSB + SSE$$

Thus, the total variation can be divided into parts explained by factors A and B and an unexplained part due to experimental errors. This equation can also be used to compute SSE, since the other sums can be easily computed. The percentage of variation explained by a factor can be used to measure the importance of the factor.

Example 21.3 For the cache comparison study of Example 21.1, the sums of squares are

$$SSY = \sum_{ij} y_{ij}^2 = 91,595$$

$$SS0 = ab\mu^2 = 3 \times 5 \times (72.2)^2 = 78,192.59$$

$$SSA = b \sum_j \alpha_j^2 = 5 \times [(-21.2)^2 + (-20.2)^2 + (41.4)^2] = 12,857.20$$

$$SSB = a \sum_i \beta_i^2 = 3 \times [(-0.5)^2 + (8.8)^2 + (-3.5)^2 + (-1.5)^2 + (3.2)^2] = 308.40$$

$$SST = SSY - SS0 = 91,595 - 78,192.59 = 13,402.41$$

$$SSE = SST - SSA - SSB = 13,402.41 - 12,857.20 - 308.40 = 236.80$$

The percentage of variation explained by the caches is

$$100 \times \frac{SSA}{SST} = 100 \times \frac{12,857.20}{13,402.41} = 95.9\%$$

The percentage of variation due to workloads is

$$100 \times \frac{SSB}{SST} = 100 \times \frac{308.40}{13,402.41} = 2.3\%$$

The unexplained variation is

$$100 \times \frac{SSE}{SST} = 100 \times \frac{236.80}{13,402.41} = 1.8\%$$

Looking at these percentages, we conclude that the choice of caches is an important parameter in the processor design. □

Analysis of Variance

- To statistically test the significance of a factor, the ANOVA Table for two factors must be calculated.

TABLE 21.4 ANOVA Table for Two Factors without Replications

Component	Sum of Squares	Percentage of Variation	Degrees of Freedom	Mean Square	F-Computed	F-Table
y	$SSY = \sum y_{ij}^2$		ab			
$\bar{y}_.$	$SS0 = ab\mu^2$		1			
$y - \bar{y}_.$	$SST = SSY - SS0$	100	$ab - 1$			
A	$SSA = b \sum \alpha_j^2$	$100 \left(\frac{SSA}{SST} \right)$	$a - 1$	$MSA = \frac{SSA}{a - 1}$	$\frac{MSA}{MSE}$	$F_{[1-\alpha, a-1, (a-1)(b-1)]}$
B	$SSB = a \sum \beta_i^2$	$100 \left(\frac{SSB}{SST} \right)$	$b - 1$	$MSB = \frac{SSB}{b - 1}$	$\frac{MSB}{MSE}$	$F_{[1-\alpha, b-1, (a-1)(b-1)]}$
e	$SSE = SST - (SSA + SSB)$	$100 \left(\frac{SSE}{SST} \right)$	$(a - 1)(b - 1)$	$MSE = \frac{SSE}{(a - 1)(b - 1)}$		

$s_e = \sqrt{MSE}$

Multiplicative Models

In the analysis presented so far, the following additive model was assumed:

$$y_i = \mu + \alpha_j + \beta_i + e_{ij}$$

This model assumes that the effects of the factors are additive.

But often the effects are multiplicative (e.g. in many cases involving Processors or Workloads).

Trick:

In such cases, the log of the responses follows an additive model.

A logarithmic transformation is also indicated if the spread in the residuals increases with the mean response.

Example ANOVA:

Example 21.4 The ANOVA for the cache comparison study of Example 21.1 is shown in Table 21.5. Notice that the F -ratio for caches is more than that obtained from the table. This reconfirms our conclusion that the choice of the cache would make a significant difference in the performance of the processor. The F -ratio for the workloads is less than that obtained from the table. Thus, in this set of experiments, the workloads did not make any significant impact on the performance. Actually, this is a real case study

TABLE 21.1 Measured Processor Times for the Cache Comparison Study

Workload	Two Caches	One Cache	No Cache
ASM	54.0	55.0	106.0
TECO	60.0	60.0	123.0
SIEVE	43.0	43.0	120.0
DHRYSTONE	49.0	52.0	111.0
SORT	49.0	50.0	108.0

TABLE 21.5 ANOVA Table for the Cache Comparison Study

Component	Sum of Squares	Percentage of Variation	Degrees of Freedom	Mean Square	F-Computed	F-Table
y	91,595.00					
$\bar{y}_.$	78,192.59					
$y - \bar{y}_.$	13,402.41	100.0	14			
Caches	12,857.20	95.9	2	6428.60	217.2	3.1
Workloads	308.40	2.3	4	77.10	2.6	2.8
Errors	236.80	1.8	8	29.60		

$$s_e = \sqrt{MSE} = \sqrt{29.60} = 5.44$$

in which the experimenter had taken precautions in choosing workloads that had comparable run times. This helped ensure that the effect of caches, if any, is not overshadowed by that due to differences in the workloads. This is an important point that is commonly missed by inexperienced analysts.

Case Study

Case Study 21.2 Two approaches to design high-speed processors are to use parallelism in time and parallelism in space. Parallelism in time is achieved by pipelining stages of execution in a single processor while parallelism in space is obtained by having several execution units that operate simultaneously. In this case study, to compare the two approaches, implementations of two architectures called Spectrum and Scheme86 were compared using several workloads. Spectrum is a Reduced Instruction Set Computer (RISC) architecture. One of its implementations, which is sold commercially as the HP9000/840 computer, is implemented with three pipelined stages. Each stage has a latency of 125 nanoseconds. The Scheme86 architecture uses a long instruction word with several independent execution units. It was designed at the Artificial Intelligence Laboratory at the Massachusetts Institute of Technology. A simulation model capable of executing any specified workloads on machines of the two architectures was used to compare their performance during design stages of Scheme86. Two different processor cycle times (125 and 62.5 nanoseconds) were used for Spectrum. They were compared with a Scheme86 implementation with 150-nanosecond technology. Execution times of the three implementations for five different workloads are listed in Table 21.11.

TABLE 21.11 Execution Times for the Scheme versus Spectrum Study

Workload	Scheme86	Spectrum125	Spectrum62.5
Garbage Collection	39.97	99.06	56.24
Pattern Match	0.958	1.672	1.252
Bignum Addition	0.01910	0.03175	0.01844
Bignum Multiplication	0.256	0.423	0.236
Fast Fourier Transform (1024)	10.21	20.28	10.14

An analysis using an additive model would conclude that there is no significant difference among the processors.

The right model is a multiplicative model. Also, statistically it is not appropriate to add observations that vary as much.

To analyze the data, take the log of the execution times.

Effects:

TABLE 21.12 Computation of Effects for the Scheme versus Spectrum Study

Workload	Scheme 86	Spectrum 125	Spectrum 62.5	Row Sum	Row Mean	Row Effect
Garbage						
Collection	1.6017	1.9959	1.7500	5.3477	1.7826	1.6212
Pattern Match	-0.0186	0.2232	0.0976	0.3022	0.1007	-0.0607
Bignum						
Addition	-1.7212	-1.4949	-1.7447	-4.9608	-1.6536	-1.8150
Bignum						
Multiplication	-0.5918	-0.3737	-0.6271	-1.5925	-0.5308	-0.6922
Fast Fourier Transform (1024)	1.0090	1.3092	1.0060	3.3243	1.1081	0.9467
Column sum	0.2791	1.6598	0.4819	2.4208		
Column mean	0.0558	0.3320	0.0964		0.1614	
Column effect	-0.1056	0.1706	-0.0650			

Special Cases

Unequal Sample Sizes

- Until now we assumed the same number of observations r (when repetitions) at each of a levels
- If number are different at different levels, the analysis is only slightly different.
- see Book p.337

Missing Observations

- If a few of the ab observations of a two factor experiment are missing the same methodology can still be used.
- see Book p.360

ANOVA:

TABLE 21.13 ANOVA Table for the Scheme versus Spectrum Study

Component	Sum of Squares	Percentage of Variation	Degrees of Freedom	Mean Square	F-Computed	F-Table
y	22.54					
$\bar{y}_.$	0.39					
$y - \bar{y}_.$	22.15	100.00	14			
Processors	0.22	1.00	2	0.11	39.29	3.11
Workloads	21.90	98.89	4	5.48	1935.48	2.81
Errors	0.02	0.10	8	0.0025		

$$s_e = \sqrt{MSE} = \sqrt{0.0025} = 0.05$$

The effect of the processors is significant. The model explains 99.9% of the variation as compared to 88% in the additive model.

Confidence Intervals for differences in processors:

TABLE 21.14 Confidence Intervals for Effect Differences in the Scheme versus Spectrum Study

	Scheme86	Spectrum125	Spectrum62.5
Scheme86			
Spectrum125		(-0.3387, -0.2136)	(-0.1031, 0.0220)*
			(0.1730, 0.2982)

* Not significant.

Spectrum86 and Spectrum62.5 are comparable since zero in interval. Spectrum125 is slower.

Simulation

- If system to be characterized is not available (during design or procurement stage)
- simulation model provides an easy way to predict the performance or compare several alternative.
- but simulation often produce no useful or misleading results.
- simulation models take a long time to develop

Always assume that your assumption is invalid.

Robert F. Tatman

Common Mistakes

- Inappropriate level of detail
- Improper language
- Unverified models
- Invalid models
- Improperly handled initial conditions
- Too short simulations
- Poor random number generators
- Improper selection of seeds

...and all the common mistakes in software engineering

Checklist

Box 24.1 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?

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?
- See also Box 2.1 for a checklist applicable to all performance evaluation projects.

Terminology

- State variable (set of all state variables)
- Event
- Time models:
 - Continuous time
 - Discrete time
- State models
 - Continuous state/continuous event
 - Discrete state/discrete event
- Determinism Property:
 - Deterministic model
 - Probabilistic model
- Time Property:
 - Static model
 - Dynamic model

- Model Equation Property
 - Linear model
 - Nonlinear model
- System Property
 - Open model
 - Closed model
- Convergence
 - Stable model
 - Unstable model

Examples:

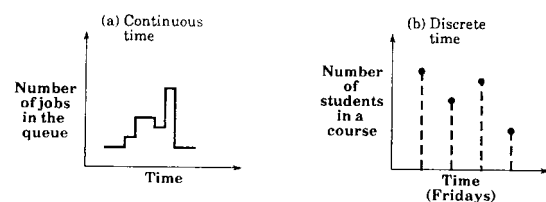


FIGURE 24.1 Continuous-time versus discrete-time models.

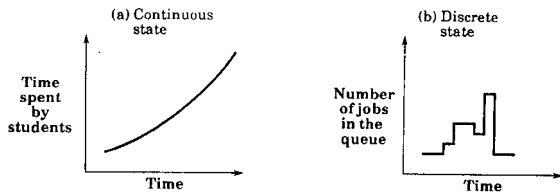


FIGURE 24.2 Continuous-state versus discrete-state models.

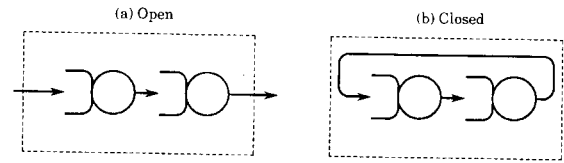


FIGURE 24.5 Open and closed models.

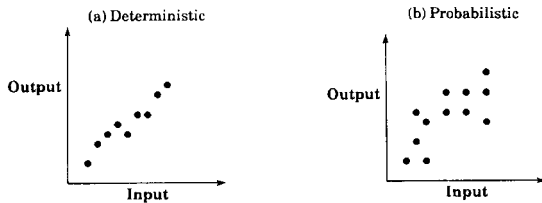


FIGURE 24.3 Deterministic and probabilistic models.

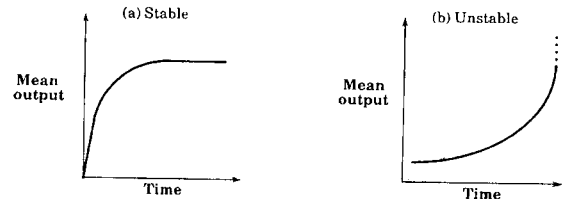


FIGURE 24.6 Stable and unstable models.

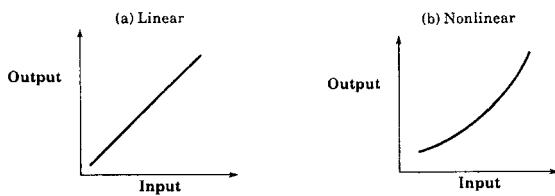


FIGURE 24.4 Linear and nonlinear models.

Languages for Simulation

Specialized languages:

- Simula
- SIMSCRIPT

General Purpose

- Oberon, Fortran, C

General Purpose with Toolkit

- Fortran + GASP
- QNET and RESQ

Differential Equation Systems

- CSMP and Dynamo

Discrete Event Systems

- Simula and GPSS

Monte Carlo Simulations

Example 24.1 The following integral is to be evaluated:

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

One way to evaluate this integral is to generate uniformly distributed random numbers x and for each number compute a function y as follows:

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

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

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

The expected value of y is

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

Thus, the integral can be evaluated by generating uniformly distributed random numbers x_i , computing y_i , and then averaging as follows:

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

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

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

Trace Driven Simulation

Advantages:

- Credibility
- Easy validation
- Accurate workload
- Detailed trade-offs
- Less randomness
- Fair comparison
- Similarity to the actual implementation

Disadvantages:

- Complexity
- Representativeness
- Finiteness
- Single point of validation
- Detail
- Trade-offs (no change to trace)

Discrete Event Simulation

- Event scheduler
- Simulation clock and a time advancing mechanism
 - unit time
 - event driven
- System state variables
- Event routines
- Input routines
- Report generators
- Initialization routines
- Trace routines
- Dynamic memory management
- Driver: main program

Data structures for the event queue

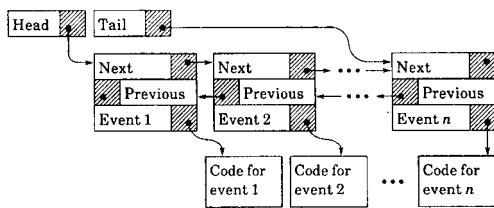


FIGURE 24.7 Doubly linked list.

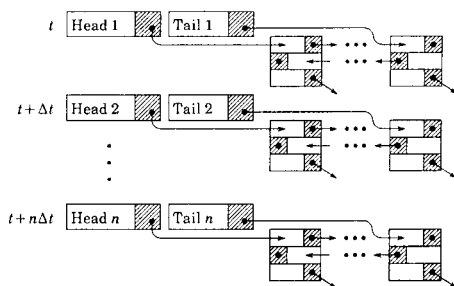
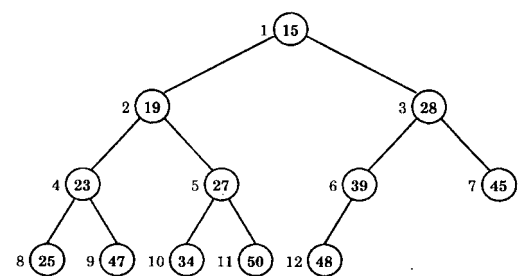


FIGURE 24.8 Indexed list.



(a) Tree representation of a heap.

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

(b) Array representation of a heap.

FIGURE 24.9 A heap.