pointers. I will try to provide material to help future learning of them all, and at least topic

Thank you. To students who responded to my request for topic suggestions. I

Balanced and Unbalanced Trees

More Mix-And-Match

CSI 310: Lecture 26
Librarians, the Iterator Trace tool, and the STL Memory Access Trace Package.

Execution behavior will also be illustrated: the Operation Counting Adapter under development at Rensselaer for measuring performance and visualizing.

Standard Template Library and the Boost Graph Library. New software tools

Implementations of concept-based standard libraries—relating to the C++

be given of how these concepts can be used in setting performance standards for
mainly in the areas of sequence algorithms and graph algorithms. Examples will

A taxonomy of algorithm concepts will be presented, with detailed development

TODAY: May 1, 2003 at 4:00pm. in IL-98 (CSI Conference Room)

Polytechnic Institute

Professor David R. Musser, Computer Science Department, Rensselaer

Concept-Based Approach

COLLOQUIUM: Toward Strong Guarantees of Algorithm Performance: A

```c
int A[5][7];
int A[2][7][3] = 9;
```

// array of ints.
// static sized 2-dim
```cpp
#include <iostream>

int main()
{
    int NROWS = 3;
    const int NCOLS = 2;

    for (int i = 0; i < NROWS; ++i)
        for (int j = 0; j < NCOLS; ++j)
            cout << i << j << endl;

    return 0;
}
```
Here is a "ragged array"; in particular, an array of pointers to C-strings:

Suppose you run your program named (under Unix) using the command line below:

```bash
proj7 inputfile.dat 35
```

Then, the above diagram describes the data structure partly contained in auto-parameter variables in main:
are needed for the 11 NON-ZERO entries.

Instead of 25 cells, only 11 nodes

of this SPARSE 5x5 MATRIX.

Above is the linked list representation

Here is this entry's column index.
its node.
is also stored in this entry's value

Column index

Array of row addresses
```c++
struct Node {
  Item data;
  Node *link;
};

template <class Item>
CACHED dynamic allocation
Node *array;
int capacity;
int used;
...

class CList {
  CList<int> MyCL(5);
  MyCL.insert(15);
  MyCL.insert(72);
  5
  2
  5
  MyCL = 2
  MyCL = 72
  MyCL = 15
};

class CList {
  template<class Item>
  class CList;
};

CACHED dynamic allocation;
```
(Linked) List of Statically Sized Lists:
buffers would be a better data structure choice. It is illustrated next.

For such new applications, one-by-one dynamic allocation of dynamically sized

small packets occur in telnet/shell protocol traffic.

Such a maximum sized buffer for a packet whose data size is only 1-2 bytes. Such

much larger, like several megabytes. So, it is very wasteful of memory to allocate

Future network hardware and protocols may have packet size limits that are

number generated by software and yet to be transmitted.

predictable is number of packets received and yet to be processed, and the

limited by a fixed amount, typically 1500 bytes or so. However, what is not

Network cards send and receive packets from the network media whose sizes are

Computer running a network interface.

The preceding slide illustrates a data structure like that in every modern
(Linked) List of Dynamically Sized Lists:
be useful. Change over the decades. But the structures themselves probably will continue to


The programming languages you use to express these structures are likely to

variables were not present in the common higher level languages of the time.

languages because features like pointers, structures, and dynamically allocated

languages since the 1960s. At that time, applications were programmed in assembly.

The data structure ideas illustrated with the diagrams were used by programmers

writing it is easy and practical when it fits your current application.

Learning to use a particular library (like STL is not a main goal; even though

3. Figuring out all the advantages and disadvantages of each choice

choice B) of data structures compared to another choice ("choice A").

structures taught in books and courses.

2. Combining of data structures by "mix-and-match" combining of data

structure definitions plus member functions.

diagrams, to implement them by writing C++ class definitions (data

1. After combining of data structures like those illustrated by the above

Some ultimate course objectives:
the oldest item in stock (FIFO)? This is taught in accounting courses.

In other words, a FIFO (First-in, First-out) store. (LIFO and FIFO are two of several accounting practices used by businesses to determine what profit results from selling a unit from an inventory. The wholesale price the business paid to buy one

to buy admission to a movie.

occur at OPPOSITE ENDS. It is like a line of people waiting at a ticket counter.

Queue: Sequence in which insertion (at the „front“) and removal (at the „rear“) (and removal) occurred only at ONE END.

Stack: Sequence in which insertion and removal occur at the „top“.

(We omit a review of stack, recursion, expression, tree relationships,}
queue is filled.

number of characters actually read, so the library can calculate how much the
function to read up to another block again. This function typically returns the
when the buffer becomes empty, the library will detect this and call the system
remove them when they are accessed by say getc one character at a time.
store the characters when they are read a block at a time, and to
in blocks of 256 or more characters. The istream library uses a buffer (i.e.,
is more efficient for the process to call a system function to read (disk) the data
input stream. However, except for hand-typed input from a terminal, it
member function istream remove and returns the earliest character
getc() member function istream remove and returns the earliest character
one. while avoiding the faster one from waiting for the slower one,
allows different parts of a system with different speeds to transmit data between
buffers. A buffer
reduction operator % is useful here.

There are more or less elegant ways to program this; the C++ modular
array length from it to make it “wrap around” to the beginning of the array.
If either position advanced beyond the end of the array, code will subtract the
they are equal, the queue is empty.
variables or pointer variables (indicate the rear and the front of the queue. If
manage an array as a circular buffer. Two position indicators (subscript
A new implementation idea for a queue (described in detail in the text) is to
implemented very much like those we covered for stacks.
implemented dynamically allocated arrays. The member functions of queue classes are
implemented several different ways: with linked lists, static sized arrays, and
Like stacks and other sequence or list type containers, queues can be
The details are given in the chapter on tree applications. Items in a heap-ordered balanced binary tree; and NOT store them fully sorted.

**IN FACT:** It is more efficient to implement a priority queue by storing the sorted list of elements that means NOT. The first element would be the one removed. This does NOT mean that they must be sorted. If the elements were sorted in decreasing order of priority, the greatest, among all the elements currently in the container.

The removal operation will remove an element whose priority is smaller than or equal to value from a list called its priority queue. A priority queue is a container where every inserted element must be given with its priority.
Computer scientists love to reduce one data structure problem to another.

item will always be the earliest inserted item that was not yet removed.

NEGATIVE of the current time (or count of items). Then, the top priority

To implement a queue, insert each item with its priority value given by the

most recently inserted item that was not yet removed.

current time (or count of items). Then, the top priority item will always be the

To implement a stack, insert each item with its priority value given by the

Either a stack or a queue can be implemented using a priority queue:
accumulate some elements.

which the number of arrivals exceeds the number served, the queue will
that the rate the "customers" are served. If there is a period of time during
We mentioned that the length of a queue would be zero if the arrival rate is less

for loop to simulate the second by second passage of time.

One variable current-second is incremented by 1 each time through
We covered the "car wash" simulation from Chapter 8. It is a time-step type of
{ do A.insert(E, -T)
with FUTURE times T, and
create new events E
.....

} Event: stimuliate(PO, 0)

.................

{ event: stimuliate(MyPo)

time = event.time;
in the priority queue
// in the earliest event
// in the earliest event
while (event = MyPo.top())

Discrete event simulation
("el-cheapo, standard, and super"") which take different amounts of time.
so several cars can be washed at the same time, and qualities of washes
implement more complex car wash simulations. There might be multiple "stalls"
With this priority queue driven discrete event simulation pattern, we can easily
priority.

events are stored in the priority queue, so the earliest events have greatest
more new events to be scheduled at various times in the future. Those new
When an event is selected and then simulated, the simulation may cause one or
occur in the future. The priority queue selects the next event scheduled to occur.
Discrete event simulation relies on a priority queue to hold the events that will
Discrete simulation. Such simulations are often done in social or scientific research.
Discrete event simulation is more efficient and more flexible than time step