Discrete event simulation: An application of priority queues.

Time step simulation

Stacks, Queues, Priority Queues

Mix-and-Match

CSI 310: Lecture 25
Be Smart: Let's MIX and MATCH!
class SList {
    Item A[N];
    ...
    static const int N=5;
} SList MySL;

Element-by-element dynamic allocation

class DList {
    struct Node {
        Item data;
        Node *link;
    } Node;
    Node *Head;
    ...
} DList<int> MyDL;
MyDL.insert(15);
MyDL.insert(72);

Static sized allocation of a List.
template <class Item>
Node *array;
int capacity;
int used;

class CList {
public:
    CList<int> MyCL(5);
    MyCL.insert(15);
    MyCL.insert(72);

private:
    CList<int>& MyCL;
    MyCL.insert(72);
    MyCL.insert(15);
    CList<int>& MyCL;

};

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/ssh 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.

The preceding slide illustrates a data structure like that in every modern
computer running a network interface.
The value of expression $A[I]$ is the address of row $A$. 

```c
int A[5][7];
```

// array of ints.
// static sized 2-dim

```c
```

18, 14
10, 11, 12
7, 8
4, 5, 6
3, 2
} = [N_COLS][2][4
{ /* Row 1 was printed */
cout >> A
for (int j = 0; j < N_COLS; j++)
    cout >> i
    cout >> " Row "
for (int i = 0; i < N_ROWS; i++)
    cout << A[i][j]
}

#include <iostream>
#include <fstream>

int main()
{
    
    fun (IA, 2, {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15})

    }
Linked List of Dynamically Sized Lists:
Here is a "ragged array", in particular, an array of pointers to C-strings:

Suppose you run your program named

```
proj7 inputfile.dat 35
```

Then, the above diagram describes the data structure partly contained in auto. parameters:

```
main(int argc, char *argv[]) {
    ... variables: main(int argc, char *argv[]) ...
}
```

In particular, an array of
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.
Here is this entry's row address.
Array of

etc.

\[
\begin{array}{ccc}
0 & 0 & -1.0 \\
0 & 0 & -1.2 \\
0 & -1.2 & 1.0 \\
0 & -1.2 & 1.0 \\
2 & 0 & 0 \\
\end{array}
\]
be useful. Changes over the decades. But the structures themselves probably will continue to
change over the decades. But the structures themselves probably will continue to
The programming languages you use to express these structures are likely 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.
variables were not present in the common higher level languages of the time.
Language because features like pointers, structures, and dynamically allocated
Language because features like pointers, structures, and dynamically allocated
Since the 1960s. At that time, applications were programmed in assembly
Since the 1960s. At that time, applications were programmed in assembly
The data structure ideas illustrated with the diagrams were used by programmers
The data structure ideas illustrated with the diagrams were used by programmers
using it is easy and practical when it fits your current application.
using 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
Learning to use a particular library (like STL) is not a main goal; even though
(“choice B”) of data structure compared to another choice (“choice A”).
(“choice B”) of data structure compared to another choice (“choice A”).
3. Figuring out all the advantages and disadvantages of making one choice
3. Figuring out all the advantages and disadvantages of making one choice
Strutures taught in books and courses.
Strutures taught in books and courses.
2. Concepting of data structures by “mix-and-match” combining of data
2. Concepting of data structures by “mix-and-match” combining of data
1. After concepting of data structures like those illustrated by the above
1. After concepting of data structures like those illustrated by the above
Some ultimate course objectives:
Some ultimate course objectives:

University at Albany Computer Science Dept.
the oldest item in stock (FIFO). This is taught in accounting courses.

Now be based on the wholesale price paid last (LIFO) or the wholesale price paid for
inventory items typically varies with when it was bought. Should the profit from a sale
selling a unit from an inventory. The wholesale price the business had paid to buy one
several accounting practices used by businesses to calculate what profit results from

In other words, a FIFO (First-in, First-out) store. (LIFO and FIFO are two of

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 „front“)

In other words, a LIFO (Last-in, First-out) store.

Stack: Sequence in which insertion and removal occur only at ONE END (the

call. We omit a review of stack, recursion, expression, tree relationships, „top“).
queue is filled.

When the buffer becomes empty, the library will detect this and call the system function to read up to another block again. This function typically returns the number of characters actually read, so the library can calculate how much the

remove them when they are accessed by say `get` 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 `ifstream` library uses a buffer (i.e., a

is more efficient for the process to call a system function to read (disk) the data

from an input stream. However, except for hand-typed input from a terminal, it

The `get`(...) member function `ifstream::remove` removes and returns the earliest character

The most prevalent use of queues in computers worldwide is for

buffers. A buffer
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. If 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 several different ways: with linked lists, static sized arrays, and

Like stacks and other sequence or list type containers, queues can be

reduction operator % is useful here.
A priority queue is a container where every inserted element must be given with a numerical value (or other kind of value from a linearly ordered set) called its priority. The removal operation will remove an element whose priority is greatest, among all the elements currently in the container.

In other words, if the elements were sorted in decreasing order of priority, the first element would be the one removed. This does NOT mean they must be sorted!

**In fact:** It is more efficient to implement a priority queue by storing the items in a heap-ordered balanced binary tree; and NOT store them fully sorted!
Important: insertions ONLY (don't count deletions)

That's how we (1) solve new problems and (2) prove new problems are hard.

Computer scientists love to reduce one data structure problem to another! Priority item will always be the earliest inserted item that was not yet removed.

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

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

most recently inserted item that was not yet removed.

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 grow.

If there is a period of time during which the rate of "customers" are served, if there is a period of time during which the rate of "customers" are served.

We mentioned that the length of a queue would be zero if the arrival rate is less than the departure rate.

We covered the "car wash" simulation from Chapter 8. It is a time-step type of simulation. One variable current-second is incremented by 1 each time through a for-loop to simulate the second by second passage of time.
do Q.insert(E', T)?
with FUTURE times T, and
create new events E

}

Event::initialize(PQ 0)
......................

{

Event::initialize(WPQ 0);  
time = event.time;

// in the priority queue
// event is the earliest

while (event = WPQ.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 event simulation is more efficient and more flexible than time step

Such simulations are often done in social or scientific research.