CSC 143

Applications of Stacks and Queues

Lists, Queues, Stacks, and Searching

- Lists are fine for searching
  - especially once they have been sorted.
- Applications that search lists have a hidden assumption: you know in advance what all the data is.
- Sometimes you don't!
  - Sometimes you discover things in the process of searching.
  - Other times the list is too long to compile before searching it.
  - Other times the list has no obvious order.
  - Other times the cost of getting all the needed information is too high.

Queues and Searching

- Queues and stacks are often appropriate structures for organizing a partial list as a process is on-going.
- Example: finding the cheapest non-stop fare from Sea-Tac to Cleveland, Dec. 24.
  - Ahead of time, you don't have a list of all flights to search through.
  - Possible process:

```
Think of the possible airlines and put them in a queue.
Take first item off the queue.
   if "airline", find all flights from Sea-Tac to
Cleveland 12/23 or 12/24 and add each to queue.
   if "flight", examine price, time, day, etc. and
decide if it's good enough to stop
Keep going until queue is empty, or until you decide to
stop.
```


Travel Search Refinements

When adding flights to the queue... does it matter what order they are added in??

- Answer: if you are looking for the absolute best, it doesn't
  - Eventually you have to look at all possibilities
- If your goal is less strict, you might stop earlier
  - "Find an *affordable* flight from Sea-Tac to Cleveland, *preferably* non-stop, Dec. 24, or Dec. 23 *if necessary*"
  - The order in which you examine the possibilities affects the answer.
- Interpret items on the queue not as individual flights, but as flight characteristics (that might expand to one, many, or no flights)

Searching: Queue vs. Stack

- Instead of a Queue, a Stack could be used.
  - This primarily affects the order in which the possibilities are expanded and examined.
- "Find a flight which costs under $200", starting with a list of airlines to consider.
- Several airlines might have a flight that qualifies. Which will be chosen...
  - using a Queue:
  - using a Stack:
- The usual picture that is drawn (a tree) gives rise to the expressions "depth first" and "breadth first".

Another Search Application

- Searching for a path to escape a maze
- Algorithm: try all possible sequences of moves in the maze until either
  - you find a sequence that works, or...
  - no more to try
- An all-possibilities search is called and “exhaustive search”
- A stack helps keep track of the possibilities
  - Traces a path of moves
  - Popping the stack moves you backwards
  - Can get a similar effect without a stack, by using recursion

Another Application: Palindromes

- "Madam, I'm Adam."

- "Enid and Edna dine."
- "A man, a plan, a canal – Panama!"

- Capitalization, spacing, and punctuation are usually ignored.
- Suppose characters are arriving on a Stream Reader. Suggest an algorithm to see if the string forms a palindrome.
  - Hint: this lecture is about stacks and queues...

Computers and Simulation

- Computer programs are often used to "simulate" some aspect of the real world
  - Movement of people and things
  - Economic trends
  - Weather forecasting
  - Physical, chemical, industrial processes

- Why?
  - Cheaper, safer, more humane
  - But have to worry about accuracy and faithfulness to real world

Queues and Simulations

- Queues are often useful in simulations
- Common considerations
  - Time between arrival
  - Service time
  - Number of servers
- Often want to investigate/predict
  - Time spend waiting in queue
  - Effect of more/fewer servers
  - Effect of different arrival rates

Example: Simulation of a Bank

- People arrive and get in line for a teller
  - Arrival patterns may depend on time of day, day of week, etc.
- When a teller is free, person at the head of the line gets served
  - Sounds like a queue is the right data model
- A bank might have different kinds of "tellers" (commercial tellers, loan officers, etc)
  - different queues for each one
- Simulation can be used to answer questions like
  - What is the average or longest wait in line
  - What would be the effect of hiring another teller

Simulations in Science

- Classical physics: describe the physical world with (differential) equations
  - Problem: too many interactions, equations too numerous and complex to solve exactly
- Alternative: build a model to simulate the operation
- Zillions of applications in physics, weather, astronomy, chemistry, biology, ecology, economics, etc. etc.
- Ideal model would allow safe virtual experiments and dependable conclusions

Time-Based Simulations

- Time-based simulation
  - Look and see what happens at every "tick" of the clock
- Might "throw dice" to determine what happens
  - Random number or probability distribution
- Size of time step?
  - A day, a millisecond, etc. depending on application

Event-Based Simulations

- Event-based simulation
  - Schedule future events and process each event as its time arrives
- Bank simulation events
  - "Customer arrives" could be one event (external)
  - "Customer starts getting service" (internal)
  - "Customer finishes transaction"
  - "Teller goes to lunch"...
- Event list holds the events waiting to happen
  - Each one is processed in chronological order
  - External events might come from a file, user input, etc.
  - Internal events are generated by other events

Another Application: Evaluating Expressions

- Expressions like "3 * (4 + 5)" have to be evaluated by calculators and compilers
- We'll look first at another form of expression, called "postfix" or "reverse Polish notation"
- Turns out a stack algorithm works like magic to do postfix evaluation
- And... another stack algorithm can be used to convert from infix to postfix!

Postfix vs. Infix

- Review: Expressions have *operators* (+, -, *, /, etc) and *operands* (numbers, variables)
- In everyday use, we write the binary operators in between the operands
  - "**4 + 5**" means "add 4 and 5"
  - called ***in**fix* notation
- No reason why we couldn't write the two operands first, then the operator
  - "**4 5 +**" would mean "add 4 and 5"
  - called ***post**fix* notation

More on Postfix

- **3 4 5 * -** means same as (3 (4 5 *) -)
  - infix: **3 - (4 * 5)**
- Parentheses aren't needed!
  - When you see an operator:

```
both operands must already be available.
Stop and apply the operator, then go on
```

- Precedence is implicit
  - Do the operators in the order found, period!
- Practice converting and evaluating:
  - 1 2 + 7 * 2 %
  - (3 + (5 / 3) * 6) - 4

Why Postfix?

- Does not require parentheses!
- Some calculators make you type in that way
- Easy to process by a program
- The processing algorithm uses a stack for operands (data)
  - simple and efficient

Postfix Evaluation via a Stack

- Read in the next "token" (operator or data)
  - If data, push it on the data stack
  - If (binary) operator (call it "op"):

```
Pop off the most recent data (B) and next most recent (A)
Perform the operation R = A op B
Push R on the stack
```

- Continue with the next token
- When finished, the answer is the stack top.
- Simple, but works like magic!
- Note: "tokens" are not necessarily single characters
  - In the expression **2002 56 +** there are three tokens
  - White space is generally ignored

Refinements and Errors

- If data stack is ever empty when data is needed for an operation:
  - Then the original expression was bad
  - Too many operators up to that point
- If the data stack is not empty after the last token has been processed and the stack popped:
  - Then the original expression was bad
  - Too few operators or too many operands

Example: 3 4 5 - *

Draw the stack at each step!

- Read 3. Push it (because it's data)
- Read 4. Push it.
- Read 5. Push it.
- Read -. Pop 5, pop 4, perform 4 - 5. Push -1
- Read *. Pop -1, pop 3, perform 3 * -1. Push -3.
- No more tokens. Final answer: pop the -3.
  - note that stack is now empty

Infix vs. Postfix

- Everyday life uses infix notation for expressions
- Computer languages most often use infix notation
- Parenthesis may be used
  - May be necessary to overcome precedence
  - May be helpful to clarify the expression
- (and) are tokens
- Our postfix evaluation algorithm doesn't work with infix.
- Solution: convert infix to postfix, then apply postfix evaluation algorithm.

Infix to Postfix

- Algorithm:
  - Read a token
  - If operand, output it immediately
  - If '(', push the '(' on stack
  - If operator:

```
if stack top is an op of => precedence: pop and output
stop when '(' is on top or stack empty
push the new operator
```

  - If ')', pop and output until '(' has been popped
  - Repeat until end of input

```
pop rest of stack
```

- Try it out!