04/11/2011

+ what we've done

- stack for expression evaluation
- stack for infix \( \Rightarrow \) postfix
- queue
- mazing problem : recursive, stack, queue
- STL : vector/stack/queue
- list

Reading Assignments : circular array / dynamic array for queue
multiple stack/queue in array
list in C

* midterm

- 04/24/2011 (Sunday) 3 pm \( \sim \) 5:30 pm
- 2.5 hrs, no extension
- open book or any printout material
- no electronic device
- English questions, English/Chinese answers
- range : textbook \& slides until 04/19/2011 class
(taught \& reading assignments)

can ask TA questions on English meaning

can use pen/pencil
Linked List

* sparse polynomial w/ dense index array (Subsec 4.4.1)

\[ x^3 + 2x^2 + 5 \]

- remove \( x^2 \)?

- insert \( 4x^1 \)?

(Sec 4.1)
* if "no moving" but still want to access sequentially?

First

graphically, \[ (3,1) \] \( \rightarrow \) \( (2,2) \) \( \rightarrow \) \( (1,4) \) \( \rightarrow \) \( (0,5) \) \( \rightarrow \) none

originally

after insertion

\[ \begin{align*}
(0,5) \rightarrow & (2,2) \\
& (3,1) \\
& (1,4) \rightarrow (0,5) \rightarrow \text{none}
\end{align*} \]

Singly linked list (chain)

* how about deletion?

\[ \begin{align*}
(3,1) \rightarrow & (2,2) \rightarrow (0,5) \rightarrow \text{none}
\end{align*} \]

1. \[ (2,2) \] . next = \( (2,2) \) . next
2. \[ \text{free} (2,2) \]

Double A
Sec 4.2) Linked List in C

READING ASSIGNMENT

Sec 4.3)

* stack w/ array

```
location 0  1  3  4  2  5  MAX-1
```

* stack w/ chain

```
host \arrow{left} \rightarrow 3 \rightarrow 4 \rightarrow 2 \rightarrow 5 \arrow{left} \rightarrow top \Rightarrow
```

* queue w/ circular array

```
front \arrow{left} \rightarrow 1 \rightarrow 3 \rightarrow 4 \rightarrow 2 \rightarrow 5 \arrow{left} \rightarrow rear
```

* queue w/ chain

```
host \arrow{left} \rightarrow 1 \rightarrow 3 \rightarrow 4 \rightarrow 2 \rightarrow 5 \arrow{left} \rightarrow last

does it work?

host \arrow{left} \rightarrow 2 \rightarrow 4 \rightarrow 3 \rightarrow 1 \arrow{left} \rightarrow rear
```

```
<table>
<thead>
<tr>
<th>\arrow{left}</th>
<th>\arrow{left}</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>\arrow{left}</td>
</tr>
</tbody>
</table>
```

front
(Subsec 4.4.3)

* erase a whole chain

* erase a whole chain w/ recycle bin

* erase a whole chain w/ recycle bin at once

* erase a whole chain w/ recycle bin "by keeping first & last"

* erase a whole circular list w/ recycle bin

(by only "last")

( because last.next is first )

1. temp = last.next
2. last.next = recycle
3. recycle = temp

* erase a whole circular list w/ recycle bin

(by only "first")

(how?)
(Sec 4.4) Other parts
* poly add
* "0" poly in circular list

READING ASSIGNMENT

(Sec 4.5)
* chain inverting
* chain concatenation
* circular list insert front
* circular list length

READING ASSIGNMENT

(Sec 4.8)

* chain: insertAfter(Δ) is easy
  \[ \text{Δ}.next = \text{Δ}.next \]
  \[ \text{Δ}.next = \text{location of } \text{Δ} \]
* insert Before(0) is hard
  1) find \( ? \) such that \( ?\.next \text{ is } 0 \)
  2) insert After(\( ? \))
* doubly linked list

trade-off:
more pointers, more expensive to maintain,
(Sec 4.6) Equivalence Class

\[ a \equiv b \]

- Symmetric: \( a \equiv b \iff b \equiv a \)
- Reflexive: \( a \equiv a \)
- Transitive: \( a \equiv b \) and \( b \equiv c \) \( \iff a \equiv c \)

* If \( 0 \equiv 4, 3 \equiv 1, 7 \equiv 4, 3 \equiv 5, 2 \equiv 6 \)

\( 0, 4, 7, 1, 3, 5, 2, 6 \)

* How?

1. Expand by symmetry
   - \( 0 \equiv 4, 4 \equiv 0 \)
   - \( 3 \equiv 1, 1 \equiv 3 \)
   - \( 7 \equiv 4, 4 \equiv 7 \)
   - \( 3 \equiv 5, 5 \equiv 3 \)
   - \( 2 \equiv 6, 6 \equiv 2 \)

2. Group
   - \( 0 \equiv 4 \)
   - \( 1 \equiv 3 \)
   - \( 2 \equiv 6 \)
   - \( 3 \equiv 1, 3 \equiv 5 \)
   - \( 4 \equiv 0, 4 \equiv 7 \)
   - \( 5 \equiv 3 \)
   - \( 6 \equiv 2 \)
   - \( 7 \equiv 4 \)

3. Search by transitivity
   - \( 0 \equiv 4, 4 \equiv 0, 4 \equiv 7, 7 \equiv 4 \)
   - \( 1 \equiv 3, 3 \equiv 1, 3 \equiv 5, 5 \equiv 3 \)
   - \( 2 \equiv 6, 6 \equiv 2 \)

* Analysis
  - \( n \) lists
  - \( m \) pairs, \( 2m \) expanded pairs \( \iff \) \( O(m+n) \)
  - Search \( n \) "starting pos", w/ \( 2m \) "moves"
(Sec 4.7) Sparse Matrix Again

* dense 2D: waste storage
* ordered triples: hard to insert & delete w/ dense array
* ordered triples w/ list

\[
\begin{bmatrix}
15 & 0 & 0 & 22 & 0 & -15 \\
0 & 11 & 3 & 0 & 0 & 0 \\
0 & 0 & 0 & -6 & 0 & 0 \\
\end{bmatrix}
\]

- easy to insert & delete
- '0' shows up many times
- hard to locate a[i][j] (linear search of \(O(#\text{element})\))

use a 'row entry' array,
- no need to store '0'
- locate a[i][j] takes linear search of \(O(#\text{col})\)

* if lazy and don't want to study Fast Transform?
  1. change next to next_of_same_row
  2. add next_of_same_col
  3. maintain 'col entry' (remove col id)

transpose: \(O(#\text{elements})\) by swapping next_of_same_row for each ele.
& next_of_same_col

can be easily improved to \(O(1)\) (how?)

* Subsec 4.7.2 ~ 4.7.4 READING ASSIGNMENT