Homework #1

Shao Hua, Huang
DEE2505 - Data Structures

October 16, 2017

1 Objective

1. To understand how an implementation of an ADT is used by an application program.
2. To become familiar with basic rules of stack, queue, and deque.
3. Implement stack, queue and deque by using STL library in c++.

2 Pseudo code

Algorithm 2.1: MAIN($argc, *argv[]$)

comment: main program

main
$fin.open(argv[1])$
$fout.open(argv[2])$
$getline(fin, cur)$
while $cur \neq$ "#"
do
- **comment:** initialize mystack with input file
- split cur with space, then store them to deque1
- push deque1 to mystack, then clear deque1
- get line to cur with fin

while get a and b value with fin
do
- **comment:** take out deques of index a and b, combine them, and push back
- $takeTargetFromStack(mystack, myqueue, deque1, deque2, a, b)$
- $lastmatch = combineProcess(deque1, deque2)$
- $pushBackProcess(mystack, myqueue, deque1, lastmatch)$

$printdeque(fout, mystack.top())$
$fin.close()$
$fout.close()$

Algorithm 2.2: TAKETARGETFROMSTACK($mystack, myqueue, deque1, deque2, a, b$)

comment: Take out the two target deques, and put uncessary deques to queue

comment: This function works with call by reference

if $a < b$

then
- $needSwap \leftarrow$ **true**
- $swap(a, b)$
- **comment:** know whether deque1 and deque2 need to swap or not

for $i \leftarrow 0$ **to** $a - 1$

do push top of mystack to myqueue, and pop mystack

$deque1 \leftarrow mystack.top()$

pop mystack

for $i \leftarrow 0$ **to** $b - a - 1$

do push top of mystack to myqueue, and pop mystack

$deque2 \leftarrow mystack.top()$

pop mystack

if $needSwap =$ **true**

then swap deque1 and deque2

Algorithm 2.3: COMBINEPROCESS($deque1, deque2$)

comment: combine deque1 and deque2 to only one deque1

while deque2 is **not** empty

do
- **if** fronts of deque1 and deque2 match
  - **then**
    - push front of deque2 to front of deque1
    - pop front of deque2
    - $lastmatch \leftarrow$ **true**
  - **else**
    - push front of deque2 to back of deque1
    - pop front of deque2
    - $lastmatch \leftarrow$ **false**

return ($lastmatch$)

Algorithm 2.4: PUSHBACKPROCESS($mystack, myqueue, deque1, lastmatch$)

comment: push back deque1 and all deques stored in myqueue

if $lastmatch$

then push deque1, then push all deques in myqueue to mystack

else push all deques in myqueue, then push deque1 to mystack

Algorithm 2.5: PRINTDEQUE($deque1$)

comment: use ostream and iterator to print deque1 to file or stdout

$it \leftarrow deque1.begin()$
while $it \neq deque.end()$
  do print $*it$ and then plus 1 to $*it$

3 Time complexity analysis

There are 3 mainly procedure in my program.
They are reading cards to stack, all combining processes, and print final deque out.
We use step count to analysis time complexity and calculate a roughly value.

3.1 Reading cards to stack

It it obvious that time complexity depends on total number of cards.
If there are n cards, then step count is approxmiately $2n$ (split token and push to stack).

3.2 All combining processes

We discard step count about getting number index a and b.
$Stacks$ means total initial deques in mystack.
$n(x)$ means number of elements of index x in stack.

3.2.1 takeTargetFromStack

(a) push to queue or assign to deque

(b) pop from stack

Discard swap process, step count is $2 \times max(a, b)$.
In total, there will be $(Stacks - 1)$ processes, and add each step count above.

3.2.2 combineProcess

This process depends on the number of deque2 elements *i.e.* $n(b)$.
Step count will be 3 time that number (push to deque1, pop from deque2, and assign last-match)
In total, there will be $(Stacks - 1)$ processes, and add each step count above.

3.2.3 pushBackProcess

In this process, step count will be the number of combined deque elements.
In total, there will be $(Stacks - 1)$ processes, and add each step count above.

3.3 Print final deque

If there are n cards, the program will enter while n times.
In this process, step count will be n.

3.4 Total step count

$$\begin{aligned} stepcount &\approx 2n + \sum_{i=1}^{Stacks-1} [2 \times max(a,b) + n(b) + (n(a) + n(b))] + n \\ &= 3n + \sum_{i=1}^{Stacks-1} [2 \times max(a,b) + n(a) + 2n(b)] \end{aligned}$$

4 Conclusion

This is an interesting problem (although meaningless) and takes me some time to learn STL library. I learned how to use stack, queue, and deque when implementing this problem. Stack has member functions such as size(), top() and swap(), queue has member functions size(), front(), back(), and deque has *iterator*, front(), back(). All of them have push() and pop(). Using these member function makes me easier to solve the problem. Besides, I also learned how to use vector to split string instead of strtok that I used to learn in C. Looking forward to the next data structure homework.