第 10 讲: 基础数据结构

姓名：林凡琪　　**学号：**211240042

评分：_____　　**评阅：**_____

2022 年 4 月 27 日

请独立完成作业，不得抄袭。
若得到他人帮助, 请致谢。
若参考了其它资料，请给出引用。
鼓励讨论，但需独立书写解题过程。

1　作业 (必做部分)

题目 1 (MA 2.6)
(只要求考察 Push 操作)

解答：
Before:

Invariance 1:if there are two elems that (a. t) ∈elems and (b, t)∈elems

∵t is the same, so we can conclude that a = b

∴a≠b are false

Invariance 2: before the operation, we know that c is the current time-point, so it is greater than any other time-point in the stack

Invariance 3:we know that max is the maximum of allowable size, so any elem in the stack, |elem|≤max

After:

Invariance 1:after the operation, the time-point of the new elem that we pushed in is different than the former elems in the stack, so we cannot find that (a, t)∈elems, (b, t)∈elems, that a = b, invariance 1 holds

Invariance 2:c' stands for the current time-point, which is the time-point of the elem that we pushed in.Also, it is greater than the former c, and the former c is greater than other time-point in the stack, so invariance 2 holds

Incariance 3: because the elem that we can push in is smaller than max, and any other elem in the stack is smaller than max, so |elems|≤max, so invariance 3 holds

---

题目 2 (TC 10.1-4)
重写 ENQUEUE 和 DEQUEUE 的代码，使之能处理队列的下溢和上溢。

Algorithm 1 ENQUEUE

解答:

```
1:  procedure ENQUEUE(Q, x)
2:      if Q.head == Q.tail + 1 then
3:          error "overflow"
4:      else
5:          Q[Q.tail] = x
6:          if Q.tail == n then
7:              Q.tail = 1
8:          else
9:              Q.tail = Q.tail + 1
10:         end if
11:     end if
12: end procedure
```

```
    procedure DEQUEUE(Q)
2:      if Q.tail == Q.head then
            error "underflow"
4:      else
            x = Q[Q.head]
6:          if Q.head == n then
                Q.head = 1
8:          else
                Q.head = Q.head + 1
10:         end if
            return x
12:     end if
    end procedure
```

题目 3 (TC 10.1-6)

说明如何用两个栈实现一个队列，并分析相关队列操作的运行时间。

解答:

设两个栈 A 和 B。

ENQUEUE 操作：把元素 push 进 B。DEQUEUE 操作：把元素从 A 中 pop 出来

如果 A 是空的，B 中的内容就从 Bpop 出来，push 进 A。

DEQUEUE 操作能在 $\Theta(n)$ 的时间里完成，但这只能在 A 为空的时候发生。

如果许多 ENQUEUE 和 DEQUEUE 的操作被执行，总时间将接近元素数量的倍数，而不是队列的整体长度。

题目 4 (TC 10.2-6)

动态集合操作 UNION 以两个不相交的集合 S_1 和 S_2 作为输入，并返回集合 $S = S_1 \cup S_2$, 包含 S_1 和 S_2 的所有元素。该操作通常会破坏集合 S_1 和 S_2。试说明如何选用一种合适的表类数据结构，来支持 $O(1)$ 时间的 UNION 操作。

解答:

选用链表的数据结构。

不妨假设 S_1 和 S_2 都是双向链表，则将 $S_1.tail$ 与 $S_2.head$ 连接，构成新链表 S。

题目 5 (TC 10.3-4)
我们往往希望双向链表的所有元素在存储器中保持紧凑，例如，在若数组表示中占用前 m 个下表为止。（在页式虚拟存储的计算环境下，即为这种情况。）假设除指向链表本身的指针外没有其他指针指向该链表的元素，试说明如何实现过程 ALLOCATE-OBJECT 和 FREE-OBJECT，使得该表示保持紧凑（提示：使用栈的数组实现）

解答:

```
function ALLOCATE-OBJECT
    if STACK-EMPTY(F) then
        error "out of space"
    else
        x = POP(F)
        return x
    end if
end function
```

```
procedure FREE-OBJECT(x)
    p = F.top − 1
    p.prev.next = x
    p.next.prev = x
    x.key = p.key
    x.prev = p.prev
    x.next = p.next
    PUSH(F, p)
end procedure
```

题目 6 (TC 10.3-5)
设 L 是一个长度为 n 的双向链表，存储于长度为 m 的数组 key、prev、和 next 中。假设这些数组由维护双链自由表 F 的两个过程 ALLOCATE-OBJEXT 和 FREE-OBJECT 进行管理。又假设 m 个元素中，恰有 n 个元素在链表 L 上，m-n 个在自由表上。给定链表 L 和自由表 F，试写出一个过程 COMPACTIFY-LIST（L,F)，用来移动 L 中的元素使其占用数组中 1,2，...，n 的位置，调整自由表 F 以保持其正确性，并且占用数组中 n+1，n+2，...,m 的位置。要求所写的过程运行时间应为 $\Theta(n)$，且只使用固定量的额外存储空间。请证明写的过程是正确的。

解答:
We represent the combination of arrays key, prev, and next by a multible-array A. Each object of A's is either in list L or in the free list F, but not in both. The procedure COMPACTIFY-LIST transposes the first object in LL with the first object in A, the second objects until the list L is exhausted.

题目 7 (TC 10.4-2)

```
procedure COMPACTIFY-LIST(L, F)
    if n == m then
        return
    end if
    e = max{max_{i∈[m]}{|key[i]|}, max_{i∈L}{|key[i]|}}
    for i ← 1tom do
        k[i]+ = 2e
    end for
    for every element of L, if its key is greater than e, reduce it by 2e
    f = 1
    while key[f]<e do
        f + +
    end while
    a = L.head
    if a > m then
        next[prev[f]] = next[f]
        prev[next[f]] = prev[f]
        next[f] = next[a]
        key[f] = key[a]
        prev[f] = prev[a]
        FREE − OBJECT(a)
        f++
        while key[f] < e do
            f + +
        end while
    end if
    while a ≠ L.head do
        if a > m then
            next[prev[f]] = next[f]
            prev[next[f]] = prev[f]
            next[f] = next[a]
            key[f] = key[a]
            prev[f] = prev[a]
            FREE − OBJECT(a)
            f + +
            while key[f] < e do
                f + +
            end while
        end if
    end while
end procedure
```

解答:

```
procedure PRINT-TREE(T.root)
    if T, root == NIL then
        return
    else
        Print(T.root.key)
        PRINT − TREE(T.root.left)
        PRINT − TREE(T.root.right)
    end if
end procedure
```

题目 8 (TC 10.4-3)

解答:

```
procedure INORDER-PRINT(T)
    let S be an empty stack
    push(S, T)
    while !S.EMPTY do
        U = pop(S)
        if U ≠ NIL then
            print (U.key)
            push(S, U.lef t)
            push(S, U.right)
        end if
    end while
end procedure
```

题目 9 (TC Problem 10-3)

解答:

a. If the original version of the algorithm takes only t iterations, then, we have that it was only at most t random skips though the list to get to the desired value, since each iteration of the original while loop is a possible random jump followed by a normal step through the linked list.

b. The for loop on lines 2–7 will get run exactly tt times, each of which is constant runtime. After that, the while loop on lines 8–9 will be run exactly X_t times. So, the total runtime is O(t + E[X_t])

c.Using equation C.25, we have that E[X_t] = $\sum$Pr{$X_t \geq$i}.So we need to show that Pr{$X_t \geq$i}$\leq(1 - \frac{i}{n})^t$.This can be seen because having X_t being greater than i means that each random choice will result in an element that is either at least i steps before the desired element, or is after the desired element.There are n - i such elements, out of the total n elements that we were pricking from.So, for a single one of the choices to be from such a range, we have a probability of $\frac{n-i}{n}$ = 1 - $\frac{i}{n}$.Since each of the selections was independent, the total probability that all of them were is $(1 - \frac{i}{n})^t$, as desired.Lastly, we can note that since the linked list has length n, the probability that X_t is greater than n is equal to zero

d.we have that t > 0, and f(x) = x^t is increasing, so
$\sum r^t = \int_0^n [r]^t \mathrm{dr} \leq \int_0^n \mathrm{f(r)dr} = \frac{n^t+1}{t+1}$
e.E[X_t]$\leq\sum(1 - r/n)^t = \sum\frac{(n-r)^t}{n^t} = \frac{1}{n^t}\sum(n-r)^t$
and $\sum(n-r)^t = (n-2)^t + (n-2)^t + \cdots + 1^t + 0^t = \sum r^t$
so E[X_t] = $\sum r^t$
f.O($\frac{t+n}{t+1}$) = O($\frac{t+n}{t}$)

g.Since we have that for any number of iterations tt that the first algorithm takes to find its answer, the second algorithm will return it in time O($\frac{t+n}{t}$).In particular, if we just have that t = $\sqrt{n}$.The second algorithm takes time only O($\sqrt{n}$).This means

that tihe first list search algorithm is O($\sqrt{n}$) as well.

h.If we don't have distinct key values, then, we may randomly select an element that is further along than we had been before, but not jump to it because it has the same key as what we were currently at. The analysis will break when we try to bound the probability that$X_t \geq$i

---

2 作业 (选做部分)

题目 1 (TC 10.1-7)

解答:

---

3 Open Topics

Open Topics 1 (Stack & Queue)
Stack 与 Queue 之间的互模拟。
参考资料:

- TC 10.1-6, TC 10.1-7
- One Stack, Two Queues @ cstheory

Open Topics 2 (Searching a sorted compact list)
讲解 TC Problem 10-3.

4 反馈