SOLVED PROBLEMS

STACKS

6.1 Consider the following stack of characters, where STACK is allocated N = 8 memory cells:

STACK: A, C, D, F, K, ___, ___, ___,

(For notational convenience, we use "___" to denote an empty memory cell.) Describe the stack as the following operations take place:

(a) POP(STACK, ITEM)
(b) POP(STACK, ITEM)
(c) PUSH(STACK, L)
(d) PUSH(STACK, P)
(e) POP(STACK, ITEM)
(f) PUSH(STACK, R)
(g) PUSH(STACK, S)
(h) POP(STACK, ITEM)

The POP procedure always deletes the top element from the stack, and the PUSH procedure always adds the new element to the top of the stack. Accordingly:

(a) STACK: A, C, D, F, ___, ___, ___, ___
(b) STACK: A, C, D, ___, ___, ___, ___, ___
(c) STACK: A, C, D, L, ___, ___, ___, ___
(d) STACK: A, C, D, L, P, ___, ___, ___
(e) STACK: A, C, D, L, ___, ___, ___, ___
(f) STACK: A, C, D, L, R, ___, ___, ___
(g) STACK: A, C, D, L, R, S, ___, ___
(h) STACK: A, C, D, L, R, ___, ___, ___

6.2 Consider the data in Problem 6.1. (a) When will overflow occur? (b) When will C be deleted before D?

(a) Since STACK has been allocated N = 8 memory cells, overflow will occur when STACK contains 8 elements and there is a PUSH operation to add another element to STACK.
(b) Since STACK is implemented as a stack, C will never be deleted before D.

6.3 Consider the following stack, where STACK is allocated N = 6 memory cells: _____

STACK: AAA, DDD, EEE, FFF, GGG, _____

Describe the stack as the following operations take place: (a) PUSH(STACK, KKK), (b) POP(STACK, ITEM), (c) PUSH(STACK, LLL), (d) PUSH(STACK, SSS), (e) POP(STACK, ITEM) and (f) PUSH(STACK, TTT).

(a) KKK is added to the top of STACK, yielding

STACK: AAA, DDD, EEE, FFF, GGG, KKK

(b) The top element is removed from STACK, yielding

STACK: AAA, DDD, EEE, FFF, GGG, _____

(c) LLL is added to the top of STACK, yielding

STACK: AAA, DDD, EEE, FFF, GGG, LLL

(d) Overflow occurs, since STACK is full and another element SSS is to be added to STACK.

No further operations can take place until the overflow is resolved—by adding additional space for STACK, for example.

6.4 Suppose STACK is allocated N = 6 memory cells and initially STACK is empty, or, in other words, TOP = 0. Find the output of the following module:

1. Set AAA := 2 and BBB := 5.
2. Call PUSH(STACK, AAA).
Call PUSH(STACK, 4).
Call PUSH(STACK, BBB + 2).
Call PUSH(STACK, 9).
Call PUSH(STACK, AAA + BBB).
3. Repeat while TOP ≠ 0:
Call POP(STACK, ITEM).
Write: ITEM.
[End of loop.]
4. Return.

Step 1. Sets AAA = 2 and BBB = 5.

Step 2. Pushes AAA = 2, 4, BBB + 2 = 7, 9 and AAA + BBB = 7 onto STACK, yielding

STACK: 2, 4, 7, 9, 7, __

Step 3. Pops and prints the elements of STACK until STACK is empty. Since the top element is always popped, the output consists of the following sequence:

7, 9, 7, 4, 2

Observe that this is the reverse of the order in which the elements were added to STACK.

6.5 Suppose a given space S of N contiguous memory cells is allocated to K = 6 stacks. Describe ways that the stacks may be maintained in S.

Suppose no prior data indicate that any one stack will grow more rapidly than any of the other stacks. Then one may reserve N/K cells for each stack, as in Fig. 6.31(a), where B_1, B_2, ..., B_6 denote, respectively, the bottoms of the stacks. Alternatively, one can partition the stacks into pairs and reserve 2N/K cells for each pair of stacks, as in Fig. 6.31(b). The second method may decrease the number of times overflow will occur.

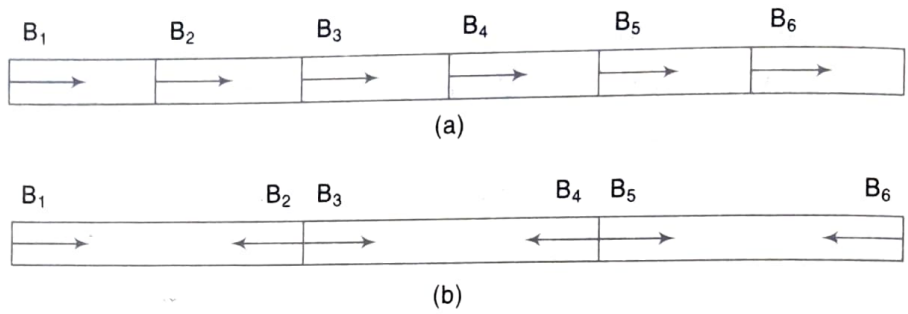

Fig. 6.31

6.6 A Programming language provides two functions ALLOCATE (X) and FREE(X) for the maintenance of linked list structures. ALLOCATE(X) allots a node with address X for use in the linked list structure and FREE(X) frees the node with address X used in the application to the AVAIL list. Assuming the AVAIL list to be maintained as a linked stack, write procedures to implement the functions ALLOCATE and FREE.

With the AVAIL list maintained as a linked stack, the procedure ALLOCATE(X) performs a pop operation on the linked stack to release the top node whose address is X. Also, the procedure FREE(X) performs a push operation on the linked stack, inserting the node that has been deleted from the application and whose address is X, to the top of the stack.

Procedure ALLOCATE(X)

1. If AVAIL = NULL then NO_MORE_NODES
2. X = AVAIL [Allot top node of AVAIL to X]
3. AVAIL = LINK(AVAIL) [Reset AVAIL to point to the next node]
4. Exit.

Procedure FREE(X)

1. LINK(X) = AVAIL
2. AVAIL = X
3. Exit

Polish Notation

6.7 Translate, by inspection and hand, each infix expression into its equivalent postfix expression:

(a) (A – B) * (D/E) (b) (A + B ↑ D)/(E – F) + G

(c) A * (B + D)/E – F * (G + H/K)

Using the order in which the operators are executed, translate each operator from infix to postfix notation, (We use brackets [ ] to denote a partial translation.)

(a) (A – B)*(D/E) = [AB–]*[DE/] = AB – DE/*

(b) (A + B ↑ D)/(E – F) + G = (A + [BD↑])/[EF –] + G = [ABD↑+]/[EF–] + G
= [ABD↑ +EF – /] + G = ABD↑+EF–/G +

(c) A*(B + D) /E – F*(G + H/K) = A*[BD +]/E – F*(G + [HK/])
= [ABD + *]/E – F*[GHK/ +]

$$= [ABD + *E/] - [FGHK/ + *]$$
$$= ABD + *E/FGHK / + * -$$

Observe that we did translate more than one operator in a single step when the operands did not overlap.

6.8 Consider the following arithmetic expression P, written in postfix notation:

P: 12, 7, 3, –, /, 2, 1, 5, +, *, +

(a) Translate P, by inspection and hand, into its equivalent infix expression.
(b) Evaluate the infix expression.

(a) Scanning from left to right, translate each operator from postfix to infix notation. (We use brackets [ ] to denote a partial translation.)

$$P = 12, [7 - 3], /, 2, 1, 5, +, *, +$$
$$= [12/(7 - 3)], 2, 1, 5, +, *, +$$
$$= [12/(7 - 3)], 2, [1 + 5], *, +$$
$$= [12/(7 - 3)], [2 * (1 + 5)], +$$
$$= 12/(7 - 3) + 2 * (1 + 5)$$

(b) Using the infix expression, we obtain:

$$P = 12/(7 - 3) + 2 * (1 + 5) = 12/4 + 2 * 6 = 3 + 12 = 15$$

6.9 Consider the postfix expression P in Problem 6.8. Evaluate P using Algorithm 6.5.

First add a sentinel right parenthesis at the end of P to obtain:

P: 12, 7, 3, –, /, 2, 1, 5, +, *, +,)

Scan P from left to right. If a constant is encountered, put it on a stack, but if an operator is encountered, evaluate the two top constants on the stack. Figure 6.32 shows the contents of STACK as each element of P is scanned. The final number, 15, in STACK, when the sentinel right parenthesis is scanned, is the value of P. This agrees with the result in Problem 6.8(b).

Symbol	STACK
12	12
7	12, 7
3	12, 7, 3
–	12, 4
/	3
2	3, 2
1	3, 2, 1
5	3, 2, 1, 5
+	3, 2, 6
*	3, 12
+	15
)	15

Fig. 6.32

6.10 Consider the following infix expression Q:

$$Q: \quad ((A + B) * D) \uparrow (E - F)$$

Use Algorithm 6.6 to translate Q into its equivalent postfix expression P.

First push a left parenthesis onto STACK, and then add a right parenthesis to the end of Q to obtain

$$Q: \quad (\,(\,A + B\,) * D\,) \uparrow (\,E - F\,)\,)$$

(Note that Q now contains 16 elements.) Scan Q from left to right. Recall that (1) if a constant is encountered, it is added to P; (2) if a left parenthesis is encountered, it is put on the stack; (3) if an operator is encountered, it "sinks" to its own level; and (4) if a right parenthesis is encountered, it "sinks" to the first left parenthesis. Figure 6.33 shows pictures of STACK and the string P as each element of Q is scanned. When STACK is empty, the final right parenthesis has been scanned and the result is

$$P: \quad A \quad B \quad + \quad D \quad * \quad E \quad F \quad - \quad \uparrow$$

which is the required postfix equivalent of Q.

Symbol	STACK	Expression P
(	((	
(	(((	
A	(((	A
+	(((+	A
B	(((+	A B
)	((	A B +
*	((*	A B +
D	((*	A B + D
)	(	A B + D *
↑	(↑	A B + D *
(	(↑ (	A B + D *
E	(↑ (	A B + D * E
−	(↑ (−	A B + D * E
F	(↑ (−	A B + D * E F
)	(↑	A B + D * E F −
)		A B + D * E F − ↑

Fig. 6.33

6.11 Translate, by inspection and hand, each infix expression into its equivalent prefix expression:

(a) (A − B) * (D / E)
(b) (A + B ↑ D)/(E − F) + G

Is there any relationship between the prefix expressions and the equivalent postfix expressions obtained in Solved Problem 6.7.

Using the order in which the operators are executed, translate each operator from infix to prefix notation.

(a) (A − B)*(D/E) = [−AB]*[/DE] = * − A B / D E

(b) (A + B ↑ D)/(E − F) + G = (A + [↑ BD])/[−EF] + G
= [+ A ↑ BD]/[−EF] + G
= [/ + A ↑ BD − EF] + G
= + / + A ↑ B D − E F G

The prefix expression is not the reverse of the postfix expression. However, the order of the operands—A, B, D and E in part (a) and A, B, D, E, F and G in part (b)—is the same for all three expressions, infix, postfix and prefix.

Quicksort

6.12 Suppose S is the following list of 14 alphabetic characters:

(D) A T A S T R U C T U R E (S)

Suppose the characters in S are to be sorted alphabetically. Use the quicksort algorithm to find the final position of the first character D.

Beginning with the last character S, scan the list from right to left until finding a character which precedes D alphabetically. It is C. Interchange D and C to obtain the list:

(C) A T A S T R U (D) T U R E S

Beginning with this C, scan the list toward D, i.e., from left to right, until finding a character which succeeds D alphabetically. It is T. Interchange D and T to obtain the list:

C A (D) A S (T) R U T T U R E S

Beginning with this T, scan the list toward D until finding a character which precedes D. It is A. Interchange D and A to obtain the list:

C A (A) (D) S T R U T T U R E S

Beginning with this A, scan the list toward D until finding a character which succeeds D. There is no such letter. This means D is in its final position. Furthermore, the letters before D form a sublist consisting of all letters preceding D alphabetically, and the letters after D form a sublist consisting of all the letters succeeding D alphabetically, as follows:

C A A (D) S T R U T T U R E S
Sublist Sublist

Sorting S is now reduced to sorting each sublist.

6.13 Suppose S consists of the following $n = 5$ letters:

(A) B C D (E)

Find the number C of comparisons to sort S using quicksort. What general conclusion can one make, if any?

Beginning with E, it takes $n - 1 = 4$ comparisons to recognize that the first letter A is already in its correct position. Sorting S is now reduced to sorting the following sublist with $n - 1 = 4$ letters:

(2) Translation of "Step K. Call P."

(a) Push the current values of the parameters and local variables and the current return address ADD onto the appropriate stacks.

(b) Reset the parameters using the new argument values, and set ADD : = [Step] K + 1.

(c) Go to Step 1. [The beginning of the procedure P.]

(3) Translation of "Step J. Return."

(a) If ADD = Main, then: Return. [Control is transferred to the main program.]

(b) Set SAVE := ADD.

(c) Restore the top values of the stacks. That is, set the parameters and local variables equal to the top values on the stacks, and set ADD equal to the top value on the stack STADD.

(d) Go to Step SAVE.

(Compare this translation algorithm with the algorithm in Sec. 6.9.)

Queues, Deques

6.22 Consider the following queue of characters, where QUEUE is a circular array which is allocated six memory cells:

FRONT = 2, REAR = 4 QUEUE: ___, A, C, D, ___, ___

(For notational convenience, we use "___" to denote an empty memory cell.) Describe the queue as the following operations take place:

(a) F is added to the queue.
(b) two letters are deleted.
(c) K, L and M are added to the queue.
(d) two letters are deleted.
(e) R is added to the queue.
(f) two letters are deleted.
(g) S is added to the queue.
(h) two letters are deleted.
(i) one letter is deleted.
(j) one letter is deleted.

(a) F is added to the rear of the queue, yielding

FRONT = 2, REAR = 5 QUEUE: ___, A, C, D, F, ___

Note that REAR is increased by 1.

(b) The two letters, A and C, are deleted, leaving

FRONT = 4, REAR = 5 QUEUE: ___, ___, ___, D, F, ___

Note that FRONT is increased by 2.

(c) K, L and M are added to the rear of the queue. Since K is placed in the last memory cell of QUEUE, L and M are placed in the first two memory cells. This yields

FRONT = 4, REAR = 2 QUEUE: L, M, ___, D, F, K

Note that REAR is increased by 3 but the arithmetic is modulo 6:

REAR = 5 + 3 = 8 = 2 (mod 6)

(d) The two front letters, D and F are deleted, leaving

FRONT = 6, REAR = 2 QUEUE: L, M, ___, ___, ___, K

(e) R is added to the rear of the queue, yielding

FRONT = 6, REAR = 3 QUEUE: L, M, R, ___, ___, K

(f) The two front letters, K and L, are deleted, leaving

FRONT = 2, REAR = 3 QUEUE: ___, M, R, ___, ___, ___

Note that FRONT is increased by 2 but the arithmetic is modulo 6:

FRONT = 6 + 2 = 8 = 2 (mod 6)

(g) S is added to the rear of the queue, yielding

FRONT = 2, REAR = 4 QUEUE: ___, M, R, S, ___, ___

(h) The two front letters, M and R, are deleted, leaving

FRONT = 4, REAR = 4 QUEUE: ___, ___, ___, S, ___, ___

(i) The front letter S is deleted. Since FRONT = REAR, this means that the queue is empty; hence we assign NULL to FRONT and REAR. Thus

FRONT = 0, REAR = 0 QUEUE: ___, ___, ___, ___, ___, ___

(j) Since FRONT = NULL, no deletion can take place. That is, underflow has occurred.

6.23 Suppose each data structure is stored in a circular array with N memory cells.

(a) Find the number NUMB of elements in a queue in terms of FRONT and REAR.
(b) Find the number NUMB of elements in a deque in terms of LEFT and RIGHT.
(c) When will the array be filled?

(a) If FRONT ≤ REAR, then NUMB = REAR – FRONT + 1. For example, consider the following queue with N = 12:

FRONT = 3, REAR = 9 QUEUE: ___, ___, *, *, *, *, *, *, *, ___, ___, ___

Then NUMB = 9 – 3 + 1 = 7, as pictured.

If REAR < FRONT, then FRONT – REAR – 1 is the number of empty cells, so

$$NUMB = N - (FRONT - REAR - 1) = N + REAR - FRONT + 1$$

For example, consider the following queue with N = 12:

FRONT = 9, REAR = 4 QUEUE: *, *, *, *, ___, ___, ___, ___, *, *, *, *

Then NUMB = 12 + 4 – 9 + 1 = 8, as pictured.

Using arithmetic modulo N, we need only one formula, as follows:

$$NUMB = REAR - FRONT + 1 \pmod N$$

(b) The same result holds for deques except that FRONT is replaced by RIGHT. That is,

$$NUMB = RIGHT - LEFT + 1 \pmod N$$

(c) With a queue, the array is full when

(i) FRONT = 1 and REAR = N or (ii) FRONT = REAR + 1

Similarly, with a deque, the array is full when

(i) LEFT = 1 and RIGHT = N or (ii) LEFT = RIGHT + 1

Each of these conditions implies NUMB = N.

6.24 Consider the following deque of characters where DEQUE is a circular array which is allocated six memory cells:

LEFT = 2, RIGHT = 4 DEQUE: ___, A, C, D, ___, ___

Describe the deque while the following operations take place.

(a) F is added to the right of the deque.
(b) Two letters on the right are deleted.
(c) K, L and M are added to the left of the deque.
(d) One letter on the left is deleted.
(e) R is added to the left of the deque.
(f) S is added to the right of the deque.
(g) T is added to the right of the deque.

(a) F is added on the right, yielding

LEFT = 2, RIGHT = 5 DEQUE: ___, A, C, D, F, ___

Note that RIGHT is increased by 1.

(b) The two right letters, F and D, are deleted, yielding

LEFT = 2, RIGHT = 3 DEQUE: ___, A, C, ___, ___, ___

Note that RIGHT is decreased by 2.

(c) K, L and M are added on the left. Since K is placed in the first memory cell, L is placed in the last memory cell and M is placed in the next-to-last memory cell. This yields

LEFT = 5, RIGHT = 3 DEQUE: K, A, C, ___, M, L

Note that LEFT is decreased by 3 but the arithmetic is modulo 6:

$$\text{LEFT} = 2 - 3 = -1 = 5 \pmod 6$$

(d) The left letter, M, is deleted, leaving

LEFT = 6, RIGHT = 3 DEQUE: K, A, C, ___, ___, L

Note that LEFT is increased by 1.

(e) R is added on the left, yielding

LEFT = 5, RIGHT = 3 DEQUE: K, A, C, —, R, L

Note that LEFT is decreased by 1.

(f) S is added on the right, yielding

LEFT = 5, RIGHT = 4 DEQUE: K, A, C, S, R, L

(g) Since LEFT = RIGHT + 1, the array is full, and hence T cannot be added to the deque. That is, overflow has occurred.