1. PoP:

when we want to pop an element, we only need to operate dequeue one time.

Therefore, running time is O(1).

Push:

when we want to push an element, first we should dequeue the stack and enqueue them in ~~an~~ temporary stack, then we enqueue the element in the stack, dequeue all the elements from temporary stack and enqueue all the elements in the stack

Therefore, running time is O(n).

2. Selection Sort Steps:

Initial: {5, 7, 0, 3, 4, 2, 6, 1}.

① : {0, 7, 5, 3, 4, 2, 6, 1}

② : {0, 1, 5, 3, 4, 2, 6, 7}

③ : {0, 1, 2, 3, 4, 5, 6, 7}

④ : {0, 1, 2, 3, 4, 5, 6, 7},

⑤ : {0, 1, 2, 3, 4, 5, 6, 7},

⑥ : {0, 1, 2, 3, 4, 5, 6, 7},

Insertion Sort Steps:

Initial : {5, 7, 0, 3, 4, 2, 6, 1}.

① : {5, 7, 0, 3, 4, 2, 6, 1}

② : {0, 5, 7, 3, 4, 2, 6, 1}.

③ : {0, 3, 5, 7, 4, 2, 6, 1}.

④ : {0, 3, 4, 5, 7, 2, 6, 1}

⑤ : {0, 2, 3, 4, 5, 7, 6, 1}.

⑥ : {0, 2, 3, 4, 5, 6, 7, 1}.

⑦ : {0, 1, 2, 3, 4, 5, 6, 7}.

Bubble Sort Steps:

Initial: {5, 7, 0, 3, 4, 2, 6, 1}.

① : {5, 0, 3, 4, 2, 6, 1, 7}.

{0, 5, 3, 4, 2, 6, 1, 7},

{0, 3, 5, 4, 2, 6, 1, 7},

{0, 3, 4, 5, 2, 6, 1, 7}, -1 Wrong understanding of Bubble sort

{0, 3, 4, 2, 5, 6, 1, 7}.

② {0, 3, 4, 2, 5, 1, 6, 7}

{0, 3, 4, 2, 5, 1, 6, 7}.

{0, 3, 4, 2, 5, 1, 6, 7}.

{0, 3, 2, 4, 5, 1, 6, 7}.

{0, 3, 2, 4, 1, 5, 6, 7}.

{0, 3, 2, 4, 1, 5, 6, 7}.
③ {0, 3, 2, 4, 1, 5, 6, 7}.
{0, 3, 2, 4, 1, 5, 6, 7}
{0, 2, 3, 4, 1, 5, 6, 7}.
{0, 2, 3, 4, 1, 5, 6, 7}.
{0, 2, 3, 1, 4, 5, 6, 7}.
{0, 2, 3, 1, 4, 5, 6, 7}
{0, 2, 3, 1, 4, 5, 6, 7}.
④ {0, 2, 3, 1, 4, 5, 6, 7}.
{0, 2, 3, 1, 4, 5, 6, 7}
{0, 2, 3, 1, 4, 5, 6, 7}.
{0, 2, 1, 3, 4, 5, 6, 7}.
{0, 2, 1, 3, 4, 5, 6, 7}
{0, 2, 1, 3, 4, 5, 6, 7}
⑤ {0, 2, 1, 3, 4, 5, 6, 7}
{0, 2, 1, 3, 4, 5, 6, 7}.
{0, 1, 2, 3, 4, 5, 6, 7}
{0, 1, 2, 3, 4, 5, 6, 7}
{0, 1, 2, 3, 4, 5, 6, 7}
{0, 1, 2, 3, 4, 5, 6, 7}
{0, 1, 2, 3, 4, 5, 6, 7}
⑥ {0, 1, 2, 3, 4, 5, 6, 7}

3. $S_1 \leftarrow$ Sum up from 0 to n-1;
$S_2 \leftarrow$ Sum up from A[0] to A[n-2]; // because there are (n-1) elements.
Let the missing number be m;
$m = S_1 - S_2$; -1 Didn't explain the time complexity and space complexity
return m;

4. ① $f(N) = N + C$.
$1 + 2 + 3 + \cdots + (N-1) = \frac{((N-1)+1)(N-1)}{2}$

-1

$= \boxed{\frac{N^2-N}{2} \sim O(N^2).}$

② $f(N) = 2N$.

Assume the array is doubled k times.

$\because N \le 2^k$.

$\therefore \log_2 N \le k$.

that is $1+2+4+8+\cdots+2^{\log_2 N}$

$= 2^{\log_2 N + 1} - 1$

$= 2 \cdot N - 1 \sim O(N)$

Therefore, the second approach is better because $N < N^2$.

5. a. $T(n) = 2T(n/2) + \log n$.

$a=2,\ b=2$. $\log_b a = 1$, so Case 3, $T(n) = \boxed{\Theta(\log n).}$

-1

b. $T(n) = 8T(n/2) + n^2$.

$a=8,\ b=2$ $\log_b a = \log_2 8 = 3$. So Case 1, $T(n) = \Theta(n^3)$.

c. $T(n) = 16T(n/2) + (n\log n)^4$

$a=16,\ b=2$ $\log_b a = \log_2 16 = 4$. So Case 2 $T(n) = \Theta(n^4 \log^5 n)$.

d. $T(n) = 7T(n/3) + n$

$a=7,\ b=3$ $\log_b a = \log_3 7 > 1$, So Case 1, $T(n) = \Theta(n^{\log_3 7})$

e. $T(n) = 9T(n/3) + n^3 \log n$

$a=9.\ b=3.$ $\log_b a = 2$, So Case 3, $T(n) = \Theta(n^3 \log n)$.

-1 Didn't check the condition that af(n/b)<=\delta*f(n)