

```

def bounded_linear_search(sequence, P):
    found = False
    i = 0
    while not found and i < len(sequence) :
        found = P(sequence[i])
        i = i + 1
    if found :
        return i - 1
    else
        return -1

```

Diagram illustrating the cost analysis of the `bounded_linear_search` function. Red dashed boxes group code segments, and red labels on the right indicate the associated cost terms:

- `found = False` is associated with  $a_1$ .
- `i = 0` is associated with  $a_2(n + 1)$ .
- The loop condition `while not found and i < len(sequence) :` is associated with  $a_2(n + 1)$ .
- The loop body `found = P(sequence[i])` and `i = i + 1` is associated with  $a_3n$ .
- The `if found :` condition is associated with  $a_4$ .
- The `return i - 1` statement is associated with  $a_5$ .
- The `return -1` statement is associated with  $a_6$ .

$$\begin{aligned}
 T_{bls}(n) &= a_1 + a_2(n + 1) + a_3n + a_4 + \max(a_5, a_6) \\
 &= b_0 + b_1n
 \end{aligned}$$

```

def binary_search(sequence, T0, item):
    low = 0
    high = len(sequence)
    while high - low > 1:
        mid = (low + high) / 2
        compar = T0(item, sequence[mid])
        if compar < 0:
            high = mid
        elif compar > 0:
            low = mid + 1
        else:
            low = mid
            high = mid + 1
    if low < high and T0(item, sequence[low]) == 0:
        return low
    else:
        return -1

```

Diagram illustrating the execution flow of the `binary_search` function. Red dashed boxes group lines of code, and red labels  $C_1$  through  $C_{10}$  are placed to the right of each group, indicating the cost of each segment.

$$\begin{aligned}
 T_{bs}(n) &= c_1 + c_2(\lg n + 1) + (c_3 + \max(c_4, c_5 + c_6, c_5 + c_7)) \lg n \\
 &\quad + c_8 + \max(c_9, c_{10}) \\
 &= d_0 + d_1 \lg n
 \end{aligned}$$

def selection_sort(sequence, T0):	$e_1 + e_2 n$
for i in range(len(sequence)):	$e_3 n$
min_pos = i	
min = sequence[i]	$e_4 n + e_5 \sum_{i=0}^{n-1} (n - i)$
for j in range(i + 1, len(sequence)):	
if T0(sequence[j], min) < 0:	$e_6 \sum_{i=0}^{n-1} (n - i - 1)$
min = sequence[j]	
min_pos = j	
sequence[min_pos] = sequence[i]	
sequence[i] = min	

$$T_{sel}(n) = f_1 + f_2 n + f_3 n^2$$