```
def cauta_student_dupa_id_recursiv(self, id_student, poz):
        #functie recursiva care cauta studentul dupa id
        #input: id_student - int
        #       poz - pozitia din sirul de studenti
        #output: student, daca exista
        #        eroare de repo - in caz contrar
        if len(self._student) == poz:
            raise RepositoryError("Id student inexistent!\n")
        student_curent = self._student[poz]
        if student_curent.get_id_student() == id_student:
            return student_curent
        return self.cauta_student_dupa_id_recursiv(id_student, poz+1)
```

Analiza complexității:

1. Complexitatea spațiu de memorare: θ(1) – in place
2. Complexitatea timp de execuție:
   a. Caz favorabil: $T(n) \in \theta(1)$ – n = 1, adică studentul căutat se află pe prima poziție în șir
   b. Caz defavorabil: $T(n) \in \theta(n)$ – studentul nu se află în șir

      Aici putem scrie așa:

$$T(n) = \begin{cases} \theta(1), \text{ dacă } n = 1 \\ T(n-1) + \theta(1) \end{cases}$$

      Recursiv, vom avea:

$$\left.\begin{aligned} T(n-1) &= T(n-2) + \theta(1) \\ T(n-2) &= T(n-3) + \theta(1) \\ &\dots \\ T(2) &= T(1) + \theta(1) \\ T(1) &= \theta(1) \end{aligned}\right\} \Rightarrow T(n) = n \cdot \theta(1) = \theta(n)$$

   c. Caz mediu: $T(n) = (1+2+\dots+n-1)/n \in \theta(n)$
   d. Complexitatea generală: O(n)