Να δώσετε την υπολογιστική πολυπλοκότητα των παραπάνω πράξεων για διπλά συνδεδεμένες λίστες. Η απάντηση σας για την υπολογιστική πολυπλοκότητα θα πρέπει να είναι σε ένα pdf αρχείο το οποίο θα βάλετε στον φάκελο solutions-ergasia1/question1 του repository σας.

ΑΠΑΝΤΗΣΗ:

1.Η Create συνάρτηση χρειάζεται έναν pointer.Επομένως έχει υπολογιστική πολυπλοκότητα O(1).

2. Η Size συνάρτηση χρειάζεται να διασχίσει όλη την λίστα για να επιστρέψει το μέγεθος της.Επομένως η υπολογιστική πολυπλοκότητα είναι O(n).

3. Η IsEmpty συνάρτηση χρειάζεται έναν pointer για να ελέγξει αν υπάρχει ένα τουλάχιστον στοιχείο εξετάζοντας αν υπάρχει το 1ο στοιχείο της λίστας ή είναι NULL.Επομένως έχει υπολογιστική πολυπλοκότητα O(1).

4. Η GetFirst συνάρτηση χρειάζεται έναν pointer (τον head) ώστε να επιστρέψει τον 1ο κόμβο.Επομένως έχει υπολογιστική πολυπλοκότητα O(1).

5. Η GetLast συνάρτηση χρειάζεται έναν pointer (τον tail) ώστε να επιστρέψει τον τελευταίο κόμβο.Επομένως έχει υπολογιστική πολυπλοκότητα O(1).

6.Η GetPrev συνάρτηση χρειάζεται έναν pointer αφού της έχουμε περάσει σαν όρισμα τον κόμβο στον οποίο θέλουμε να βρούμε τον προηγούμενο του.Επομένως έχει υπολογιστική πολυπλοκότητα O(1).

7. Η GetNext συνάρτηση χρειάζεται έναν pointer αφού της έχουμε περάσει σαν όρισμα τον κόμβο στον οποίο θέλουμε να βρούμε τον επόμενο του.Επομένως έχει υπολογιστική πολυπλοκότητα O(1).

8. Η AddBefore συνάρτηση αφού της περνάμε σαν όρισμα τον κόμβο της λίστας στο οποίο θέλουμε να εισάγουμε πριν από αυτόν έναν κόμβο έχει υπολογιστική πολυπλοκότητα O(1), αφού δεν χρειάζεται να διασχίσουμε την λίστα.

9. Η AddAfter συνάρτηση αφού της περνάμε σαν όρισμα τον κόμβο της λίστας στο οποίο θέλουμε να εισάγουμε μετά από αυτόν έναν κόμβο έχει υπολογιστική πολυπλοκότητα O(1), αφού δεν χρειάζεται να διασχίσουμε την λίστα.

10. Η AddFirst συνάρτηση χρειάζεται έναν pointer για να εισάγουμε στην αρχή της λίστας ένα στοιχείο, αφού έχουμε τον head κόμβο.Επομένως έχει υπολογιστική πολυπλοκότητα O(1).

11. Η AddLast συνάρτηση χρειάζεται έναν pointer για να εισάγουμε στο τέλος της λίστας ένα στοιχείο, αφού έχουμε τον tail κόμβο.Επομένως έχει υπολογιστική πολυπλοκότητα O(1).

12. Η Remove συνάρτηση αφού της περνάμε σαν όρισμα τον κόμβο της λίστας τον οποίο θέλουμε να διαγράψουμε δεν χρειάζεται να διασχίσουμε την λίστα οπότε έχει υπολογιστική πολυπλοκότητα O(1).

13. Η Print συνάρτηση χρειάζεται να διασχίσει όλη την λίστα για να εκτυπώσει όλους τους κόμβους.Επομένως έχει υπολογιστική πολυπλοκότητα O(n).

14&15. Οι άλλες δύο βοηθητικές συναρτήσεις που χρησιμοποιώ είναι απλές.Αναφέρομαι στην item οπου επιστρέφει την τιμή του κόμβου και την make που δημιουργεί έναν κόμβο και έχουν υπολογιστική πολυπλοκότητα O(1).
Από 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 έχω:

T(n)= O(1) +O(n) + O(1) + O(1) + O(1) + O(1) + O(1) + O(1) + O(1) + O(1) + O(1) + O(1) +O(n) + O(1) + O(1) = O(n)

Επομένως η υπολογιστική πολυπλοκότητα της διπλά συνδεδεμένης λίστας είναι **O(n).**