4.10 Παράσταση Αραιού Πολυωνύμου με Συνδεδεμένη Λίστα

Είναι γνωστό ότι σ' ένα πολυώνυμο P(x) της μορφής

$$P(x)= a_0 + a_1x + a_2x^2 + ... + a_nx^n$$

οι a_0, a_1, a_2, ..., a_n ονομάζονται **συντελεστές** (**coefficients**) του πολυωνύμου και ο αριθμός n που αντιστοιχεί στη μεγαλύτερη δύναμη του x με $a_n \neq 0$ ονομάζεται **βαθμός**(**degree**) του πολυωνύμου. Έτσι, για παράδειγμα, το πολυώνυμο

$$P(x)= 3 - 5x + 21x^2 + x^3$$

έχει συντελεστές 3, -5, 21, 1 και βαθμό 3, ενώ το πολυώνυμο

$$Q(x)= 5$$

έχει έναν συντελεστή, 5, και βαθμό 0.

Ένα πολυώνυμο μπορούμε να το δούμε σαν μια λίστα συντελεστών και να το παραστήσουμε χρησιμοποιώντας κάποια από τις υλοποιήσεις λιστών που εξετάσαμε μέχρι τώρα. Για παράδειγμα, το πολυώνυμο

$$P(x)= 3 + 10x - x^2 - 4x^4 + x^7$$

μπορεί να γραφτεί σαν μια λίστα συντελεστών

(3, 10, -1, 0, -4, 0, 0, 1)

και να αποθηκευτεί σ' έναν πίνακα P 10 θέσεων, όπως φαίνεται παρακάτω:

i	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
P[i]	3	10	-1	0	-4	0	0	1	0	0

Αν ο βαθμός του πολυωνύμου που αποθηκεύεται σε έναν τέτοιο πίνακα δεν διαφέρει πολύ από το άνω όριο που τίθεται από το μέγεθος του πίνακα και οι μηδενικοί συντελεστές δεν είναι πολλοί, τότε μια αναπαράσταση σαν την παραπάνω είναι ικανοποιητική. Ωστόσο, υπάρχουν πολυώνυμα που έχουν λίγους μη μηδενικούς συντελεστές. Τα πολυώνυμα αυτά ονομάζονται **αραιά πολυώνυμα** (**sparse polynomials**) και ένας πίνακας δεν είναι η κατάλληλη αποθηκευτική δομή για τους συντελεστές τους. Ένα παράδειγμα αραιού πολυωνύμου είναι το παρακάτω:

$$P(x)= 12x - 3x^2 + 3x^{70}$$

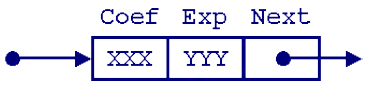
το οποίο μπορεί να γραφτεί και ως

$$P(x)= 0 + 12x - 3x^2 + 0x^3 + 0x^4 + ... + 0x^{69} + 3x^{70}$$

Η αποθήκευση αυτού του πολυωνύμου σε πίνακα θα απαιτούσε 71 θέσεις, από τις οποίες μόνο οι 3 θα είχαν μη μηδενικές τιμές, ενώ οι υπόλοιπες 68 θα είχαν μηδενικές. Κάτι τέτοιο συνεπάγεται σπατάλη μνήμης, την οποία μπορούμε να αποφύγουμε αν αποθηκεύσουμε μόνο τους μη μηδενικούς συντελεστές. Βέβαια, σ' αυτήν την περίπτωση θα πρέπει να αποθηκεύσουμε και τη δύναμη του x που αντιστοιχεί σε κάθε συντελεστή. Επομένως, μπορούμε να παραστήσουμε ένα πολυώνυμο σαν μια λίστα από ζεύγη συντελεστών-εκθετών. Για το παραπάνω πολυώνυμο η λίστα που σχηματίζεται είναι η ακόλουθη:

((12, 1),(-3, 2),(3, 70))

Για την αποθήκευση μιας τέτοιας λίστας μπορούμε να χρησιμοποιήσουμε έναν πίνακα εγγραφών με ένα πεδίο *Coefficient* για τον συντελεστή και ένα πεδίο *Exponent* για τον εκθέτη. Πάλι όμως θα έχουμε το πρόβλημα του σταθερού μεγέθους του πίνακα που περιορίζει το μέγεθος της λίστας. Κατά συνέπεια, μια πιο κατάλληλη δομή αποθήκευσης είναι η συνδεδεμένη λίστα. Κάθε κόμβος της λίστας θα περιέχει ένα τμήμα Coef, για την αποθήκευση ενός μη μηδενικού συντελεστή, ένα τμήμα Exp, για την αποθήκευση του αντίστοιχου εκθέτη, και ένα τμήμα Next, για την αποθήκευση του δείκτη που δείχνει στον επόμενο όρο του πολυωνύμου:



Για παράδειγμα, το πολυώνυμο

$$P(x)= 12x - 3x^2 + 3x^{70}$$

μπορεί να αποθηκευτεί σε μια συνδεδεμένη λίστα με κόμβο κεφαλή, όπου ο βαθμός του πολυωνύμου αποθηκεύεται στο τμήμα *Exp* του κόμβου κεφαλή:

Οι απαραίτητες δηλώσεις για ένα τέτοιο συνδεδεμένο πολυώνυμο είναι:

```
typedef int CoefficientType;
typedef  struct PolyNode *PolyPointer;
typedef  struct PolyNode {
                  CoefficientType Coef;
                   int  Expo;
                   PolyPointer next;
} PolyNode;
```

Για να γίνει κατανοητός ο τρόπος επεξεργασίας αυτών των πολυωνύμων, μπορούμε να θεωρήσουμε τη λειτουργία της πρόσθεσης δυο πολυωνύμων, π.χ. των P(x) και Q(x), που φαίνονται παρακάτω:

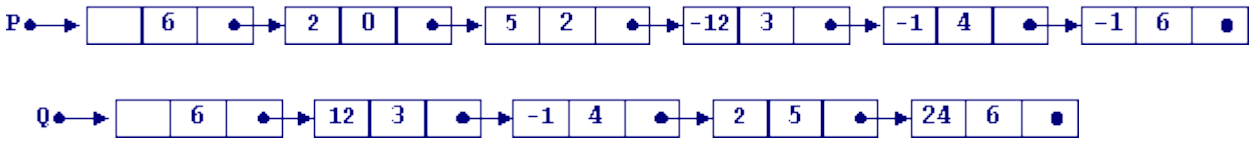
$$P(x)= 2 + 5x^2 - 12x^3 - x^4 - x^6$$

$$Q(x)= 12x^3 - x^4 + 2x^5 + 24x^6$$

Είναι γνωστό ότι, για να προσθέσουμε δύο πολυώνυμα, προσθέτουμε τους συντελεστές των όρων που έχουν τον ίδιο εκθέτη. Στη συγκεκριμένη περίπτωση, το αποτέλεσμα της πρόσθεσης είναι:

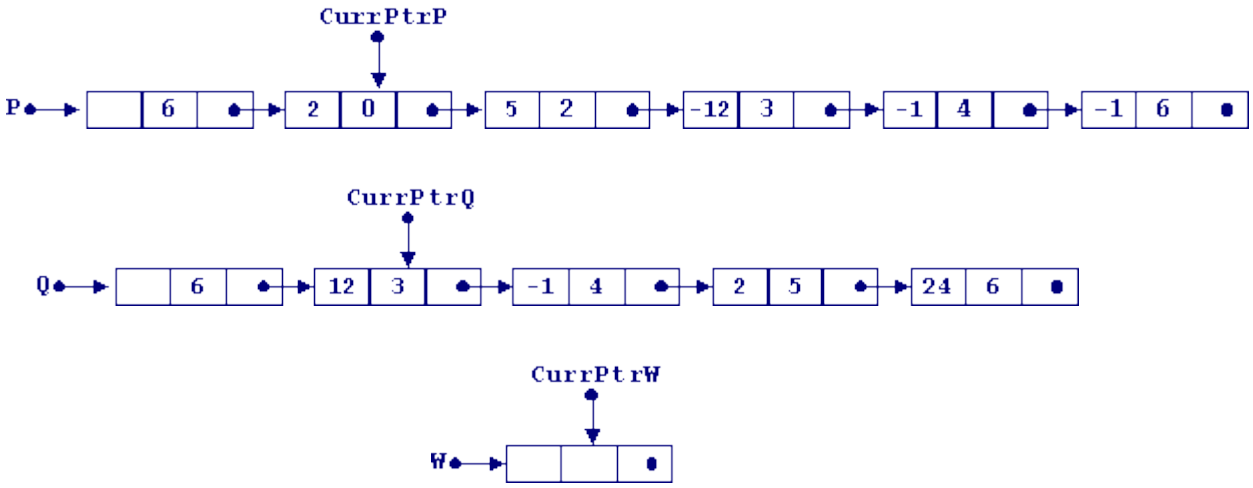
$$W(x)= P(x) + Q(x)= 2 + 5x^2 - 2x^4 + 2x^5 + 23x^6$$

Αν παραστήσουμε τα πολυώνυμα P(x) και Q(x) σαν συνδεδεμένες λίστες με κόμβους κεφαλή



τότε θα παραστήσουμε και το W(x) σαν συνδεδεμένη λίστα που αρχικά έχει μόνο τον κόμβο κεφαλή χωρίς τιμές:

Χρειαζόμαστε τρεις δείκτες, CurrPtrP, CurrPtrQ και CurrPtrW. Οι CurrPtrP και CurrPtrQ θα δείχνουν στον κόμβο των λιστών P και Q αντίστοιχα που επεξεργαζόμαστε την τρέχουσα στιγμή, ενώ ο CurrPtrW θα δείχνει στον τελευταίο κόμβο που προστέθηκε στη λίστα W. Αρχικά οι CurrPtrP και CurrPtrQ θα δείχνουν στους κόμβους P^.Next και Q^.Next και ο CurrPtrW θα δείχνει εκεί που δείχνει και ο W:



Κάθε φορά, συγκρίνουμε τους εκθέτες που είναι αποθηκευμένοι στους κόμβους στους οποίους δείχνουν οι CurrPtrP και CurrPtrQ. Αν οι εκθέτες είναι ίδιοι, τότε προσθέτουμε τους αντίστοιχους συντελεστές. Εδώ διακρίνουμε δύο περιπτώσεις:

α) αν το αποτέλεσμα δεν είναι μηδέν, τότε προσθέτουμε έναν καινούργιο κόμβο στη λίστα W και αποθηκεύουμε το άθροισμα των συντελεστών στο τμήμα Coef του και τον κοινό εκθέτη στο τμήμα Exp. Οι τρεις δείκτες CurrPtrP, CurrPtrQ και CurrPtrW προχωρούν στους επόμενους κόμβους αντίστοιχα.

β) αν το άθροισμα των συντελεστών είναι μηδέν, τότε δεν προσθέτουμε νέο κόμβο στη λίστα W, αλλά απλώς αυξάνουμε τους CurrPtrP και CurrPtrQ.

Αν οι εκθέτες είναι διαφορετικοί, τότε προστίθεται ένας νέος κόμβος στην λίστα W, με τον μικρότερο από τους δύο εκθέτες στο τμήμα Exp και τον αντίστοιχο συντελεστή στο τμήμα Coef. Ο δείκτης που έδειχνε στον μικρότερο εκθέτη και ο CurrPtrW προχωρούν στους επόμενους κόμβους. Η διαδικασία συνεχίζεται μέχρι να φτάσουμε στο τέλος της λίστας P ή Q, δηλαδή μέχρι ένας από τους CurrPtrP και CurrPtrQ να πάρει τιμή NULL. Αν δεν φτάσουμε συγχρόνως στο τέλος και της άλλης λίστας, τότε αντιγράφουμε τους υπόλοιπους κόμβους και τους προσθέτουμε στη λίστα W και θέτουμε το τμήμα δεσμού του τελευταίου κόμβου ίσο με NULL.

Σε μορφή κώδικα C η διαδικασία της πρόσθεσης δύο συνδεδεμένων πολυωνύμων είναι:

```
void LinkedPolyAdd(PolyPointer P,PolyPointer Q, PolyPointer *C);
/*Δέχεται:        Δύο πολυώνυμα, P και Q.
  Λειτουργία:     Υπολογίζει το άθροισμα W = P + Q.
  Επιστρέφει:     Το πολυώνυμο W.
  Σημείωση:       Τα πολυώνυμα παριστάνονται ως συνδεδεμένες λίστες με κόμβο
                  κεφαλή.*/
```

```
{
  PolyPointer ptrP, ptrQ, ptrW, TempPtr;
  CoefficientType sum;

  ptrP = P->next;
  ptrQ = Q->next;
  *W=(PolyPointer)malloc(sizeof(struct PolyNode));
  ptrW = *W;
  while (ptrP != NULL && ptrQ != NULL)
  {
      if (ptrP->Expo < ptrQ->Expo)         /*αντιγραφή του όρου από το P*/
      {
          Attach(ptrP->Coef, ptrP->Expo, &ptrW);
          ptrP = ptrP->next;
      }
      else if (ptrQ->Expo < ptrP->Expo)     /*αντιγραφή του όρου από το Q*/
      {
          Attach(ptrQ->Coef, ptrQ->Exp, &ptrW);
          ptrQ = ptrQ->Next;
      }
      else                                  /*ίσιοι εκθέτες*/
      {
          sum = ptrP->Coef + ptrQ->Coef;
          if (sum != 0)
              Attach(sum, ptrP->Expo, &ptrW);
          ptrP = ptrP->next;
          ptrQ = ptrQ->next;
      }
   }
  /*Αντιγραφή των υπόλοιπων όρων του P ή του Q στο W*/
  if (ptrP != NULL)
       TempPtr = ptrP;
  else
       TempPtr=ptrQ;
  while (TempPtr != NULL)
  {
      Attach(TempPtr->Coef, TempPtr->Expo, &ptrW);
      TempPtr = TempPtr->next;
  }
  pptrW->next = NULL;
}
```

Η void συνάρτηση Attach που χρησιμοποιείται είναι η ακόλουθη:

```
void Attach(CoefficientType Co, int Ex, PolyPointer *Last);
/*Δέχεται:      Έναν συντελεστή Coef, έναν εκθέτη Exp και έναν δείκτη Last.
  Λειτουργία:   Δημιουργεί έναν κόμβο που περιέχει τους Coef και Exp, τον συνδέει με
                τον κόμβο στον οποίο δείχνει ο Last και κάνει τον Last να δείχνει σ'
                αυτόν τον νέο κόμβο.
  Επιστρέφει:   Τον τροποποιημένο δείκτη Last.*/
{
  PolyPointer TempPtr;

  TempPtr= (PolyPointer)malloc(sizeof(struct PolyNode));
  TempPtr->Coef = Co;
  TempPtr->Exp = Exp;
  TempPtr->next = NULL;
  (*Last)->next = TempPtr;
  *Last= TempPtr
}
```

Το πρόγραμμα PolyAdd.c υλοποιεί την διαδικασία της πρόσθεσης δύο πολυωνύμων, που αποθηκεύονται ως συνδεδεμένες λίστες.

Αν χρησιμοποιηθεί πίνακας ως αποθηκευτική δομή των πολυωνύμων, όπως περιγράφτηκε πιο πάνω, τότε η διαδικασία της πρόσθεσης είναι αρκετά πιο απλή. Το μόνο που χρειάζεται είναι ο παρακάτω βρόχος for:

```
for (i=0; i<= MaxDegree+1; i++)
    W[i]=P[i]+Q[i];
```

όπου MaxDegree είναι ο μέγιστος βαθμός των πολυωνύμων P και Q.