Algorithms and Data Structures

Homework 1

14058024 Özlem Er

11.10.2019

Problem Definition: The problem is summation and multiplication of two polynomials.

Algorithm Analysis: There is a function called ‘SumPolynomials’ calculates the sum of two polynomials. It compares the degrees one by one and makes the summation. That is why the complexity of this function is O(m+n) where m and n are numbers of terms of polynomials.

The function called ‘MultiplyPolynomial’ and ‘Simplification’ calculates product of two polynomial. ‘MultiplyPolynomial’ calculates production of two polynomials terms. Complexity of this function is O(mxn) where m and n are numbers of terms of polynomials. ‘Simplification’ remove the duplication terms after production. Complexity of this function is also O(n^2) where n is numbers of terms of polynomial.

Output:

```
Please enter the number of terms for first polynomial: 3
Please enter the number of terms for second polynomial: 3
Please enter terms from highest degree to lowest degree for first polynomial
node 1 coefficient: 3
node 1 degree: 10
node 2 coefficient: 2
node 2 degree: 5
node 3 coefficient: 3
node 3 degree: 0
Please enter terms from highest degree to lowest degree for second polynomial
node 1 coefficient: 4
node 1 degree: 8
node 2 coefficient: 3
node 2 degree: 5
node 3 coefficient: 2
node 3 degree: 0
SUM: 3x^10 + 4x^8 + 5x^5 + 5x^0
MULTIPLY: 12x^18 + 9x^15 + 12x^10 + 8x^13 + 13x^5 + 12x^8
```

```
Please enter the number of terms for second polynomial: 5
Please enter terms from highest degree to lowest degree for first polynomial
node 1 coefficient: -3
node 1 degree: 7
node 2 coefficient: 3
node 2 degree: 6
node 3 coefficient: 4
node 3 degree: 5
node 4 coefficient: 4
node 4 degree: 3
Please enter terms from highest degree to lowest degree for second polynomial
node 1 coefficient: 7
node 1 degree: 9
node 2 coefficient: 5
node 2 degree: 4
node 3 coefficient: 5
node 3 degree: 3
node 4 coefficient: 2
node 4 degree: 2
node 5 coefficient: 1
node 5 degree: 1
SUM: 7x^9 + -3x^7 + 3x^6 + 4x^5 + 5x^4 + 9x^3 + 2x^2 + 1x^1
MULTIPLY: -21x^16 + -15x^11 + 0x^10 + 29x^9 + 23x^8 + 21x^15 + 31x^7 + 28x^14 + 24x^6 +
```

```
Please enter the number of terms for first polynomial: 4
Please enter the number of terms for second polynomial: 1
Please enter terms from highest degree to lowest degree for first polynomial
node 1 coefficient: 5
node 1 degree: 5
node 2 coefficient: 4
node 2 degree: 4
node 3 coefficient: 3
node 3 degree: 3
node 4 coefficient: 2
node 4 degree: 2
Please enter terms from highest degree to lowest degree for second polynomial
node 1 coefficient: 8
node 1 degree: 8
SUM: 8x^8 + 5x^5 + 4x^4 + 3x^3 + 2x^2
MULTIPLY: 40x^13 + 32x^12 + 24x^11 + 16x^10
```

```
Please enter the number of terms for first polynomial: 9
Please enter the number of terms for second polynomial: 17
Please enter terms from highest degree to lowest degree for first polynomial
node 1 coefficient: 16
node 1 degree: 15
node 2 coefficient: 14
node 2 degree: 13
node 3 coefficient: 12
node 3 degree: 11
node 4 coefficient: 10
node 4 degree: 9
node 5 coefficient: 8
node 5 degree: 6
node 6 coefficient: 5
node 6 degree: 5
node 7 coefficient: 4
node 7 degree: 4
node 8 coefficient: 3
node 8 degree: 3
node 9 coefficient: 2
node 9 degree: 2
Please enter terms from highest degree to lowest degree for second polynomial
node 1 coefficient: 4
node 1 degree: 25
node 2 coefficient: 4
node 2 degree: 23
node 3 coefficient: 4
node 3 degree: 22
node 4 coefficient: 4
node 4 degree: 21
node 5 coefficient: 4
node 5 degree: 20
node 7 degree: 18
node 8 coefficient: 4
node 8 degree: 17
node 9 coefficient: 4
node 9 degree: 15
node 10 coefficient: 4
node 10 degree: 14
node 11 coefficient: 4
node 11 degree: 13
node 12 coefficient: 4
node 12 degree: 12
node 13 coefficient: 5
node 13 degree: 11
node 14 coefficient: 4
node 14 degree: 10
node 15 coefficient: 4
node 15 degree: 9
node 16 coefficient: 4
node 16 degree: 8
node 17 coefficient: 4
node 17 degree: 7
SUM: 4x^25 + 4x^23 + 4x^22 + 4x^21 + 4x^20 + 4x^19 + 4x^18 + 4x^17 + 20x^15 + 4x^14 + 18
x^13 + 4x^12 + 17x^11 + 4x^10 + 14x^9 + 4x^8 + 4x^7 + 8x^6 + 5x^5 + 4x^4 + 3x^3 + 2x^2
MULTIPLY: 64x^40 + 120x^38 + 64x^37 + 168x^36 + 120x^35 + 208x^34 + 168x^33 + 208x^32 +
228x^30 + 200x^29 + 272x^28 + 236x^27 + 304x^26 + 256x^25 + 310x^24 + 296x^23 + 276x^22
+ 176x^31 + 212x^21 + 226x^20 + 164x^19 + 168x^18 + 136x^17 + 133x^16 + 92x^15 + 91x^14
+ 90x^13 + 56x^12 + 36x^11 + 20x^10 + 8x^9
```

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

typedef struct polynomial{
   int coef;
   int deg;
   struct polynomial *next;

}poly;

poly * CreateNode();
poly * CreatePoly(int);
void displayList(poly *);
void SumPolynomials(poly *, poly *,poly *);
void Simplification(poly *);
void MultiplyPolynomial(poly *, poly* ,poly *);


int main()
{
   printf("Hello World\n");
   poly *p1, *p2,*sum,* product;
   int n1,n2;
   printf("Please enter the number of terms for first polynomial: ");
   scanf("%d",&n1);
   printf("Please enter the number of terms for second polynomial: ");
   scanf("%d",&n2);
   printf("Please enter terms from highest degree to lowest degree for first polynomial\n");
   p1 = CreatePoly(n1);
   printf("Please enter terms from highest degree to lowest degree for second polynomial\n");
   p2 = CreatePoly(n2);
   sum = CreateNode();
   SumPolynomials(p1,p2,sum);
   printf("SUM: ");
   displayList(sum);
   product = CreateNode();
   MultiplyPolynomial(p1,p2,product);
   Simplification(product);
   printf("MULTIPLY: ");
   displayList(product);

   return 0;
}

poly * CreateNode(){
   poly *node;

   node = (poly *)calloc(sizeof(poly),1);
   if(node == NULL){
      printf("Memory can not be allocated! \'CreateNode\'");
      exit(0);
   }
   else{
      node->next = NULL;
      return node;
   }
}
```

```
poly * CreatePoly(int n){
   poly *polynomial,*tail, *tmp;
   int coef, deg,i;

   polynomial = (poly *)calloc(sizeof(poly),1);
   if(polynomial == NULL){
      printf("Memory can not be allocated! \'CreatePoly\'");
      exit(0);
   }
   else{
      printf("node 1 coefficient: ");
      scanf("%d",&coef);
      printf("node 1 degree: ");
      scanf("%d",&deg);
      polynomial->coef = coef;
      polynomial->deg = deg;
      polynomial->next = NULL;
      tmp = polynomial;
      for(i=2;i<=n;i++){
         tail = (poly *)calloc(sizeof(poly),1);
         if(tail == NULL){
            printf("Memory can not be allocated! \'CreatePoly\'");
            exit(0);
         }
         else{
            printf("node %d coefficient: ",i);
            scanf("%d",&coef);
            printf("node %d degree: ",i);
            scanf("%d",&deg);

            tail->coef = coef;
            tail->deg = deg;
            tail->next = NULL;

            tmp->next = tail;
            tmp = tmp->next;
         }
      }
      return polynomial;
   }
}
void displayList(poly *polynomial)
{
   poly *tmp;
   if(polynomial == NULL)
   {
      printf(" List is empty.");
   }
   else
   {
      tmp = polynomial;
      while(tmp->next != NULL)
      {
         printf("%dx\^%d",tmp->coef, tmp->deg);
         tmp = tmp->next;
         if(tmp->next !=NULL)
            printf(" + ");
      }
      printf("\n");
   }
}
```

```
void Simplification(poly * polyp){
   poly *ptr1,*ptr2,*delete;
   ptr1 = polyp;

   while(ptr1 && ptr1->next){
      ptr2 = ptr1;

      while(ptr2->next){
         if(ptr1->deg == ptr2->next->deg){
            ptr1->coef = ptr1->coef + ptr2->next->coef;
            delete = ptr2->next;
            ptr2->next = ptr2->next->next;

            free(delete);
         }
         else{
            ptr2 = ptr2->next;
         }
      }
      ptr1 = ptr1->next;
   }
}
void MultiplyPolynomial(poly *poly1, poly* poly2,poly *product){
   poly *ptr1, *ptr2, *ptrp;
   int coef, deg;

   ptrp = product;
   ptr1 = poly1;
   ptr2 = poly2;

   coef = 0; deg = 0;
   while(ptr1){
      while(ptr2){

         ptrp->coef = (ptr1->coef) * (ptr2->coef);
         ptrp->deg = (ptr1->deg)+ (ptr2->deg);
         ptrp->next = CreateNode();
         ptrp = ptrp->next;
         ptr2 = ptr2->next;
      }
      ptr2 = poly2;
      ptr1 = ptr1->next;
   }

}
```

```
void SumPolynomials(poly* poly1, poly *poly2, poly *sum){

   poly* ptr1,*ptr2,*ptrs,*ptrf,*head;
   ptr1 = poly1;
   ptr2 = poly2;
   ptrs = CreateNode();
   ptrs=sum;

   while(ptr1 && ptr2){
      if(ptr1->deg > ptr2->deg){
         ptrs->coef = ptr1->coef;
         ptrs->deg = ptr1->deg;
         ptr1 = ptr1->next;
      }
      else if(ptr1->deg < ptr2->deg){
         ptrs->coef = ptr2->coef;
         ptrs->deg = ptr2->deg;
         ptr2 = ptr2->next;
      }
      else{
         ptrs->coef = ptr1->coef + ptr2->coef;
         ptrs->deg = ptr1->deg;
         ptr1 = ptr1->next;
         ptr2 = ptr2->next;
      }

      ptrs->next = CreateNode();
      ptrs = ptrs->next;
   }



   while(ptr1 || ptr2){
      if(ptr1){
         ptrs->coef = ptr1->coef;
         ptrs->deg = ptr1->deg;
         ptr1 = ptr1->next;
      }
      if(ptr2){
         ptrs->coef = ptr2->coef;
         ptrs->deg = ptr2->deg;
         ptr2 = ptr2->next;
      }

      ptrs->next = CreateNode();
      ptrs = ptrs->next;
   }

   ptrs->next = NULL;

}
```