알고리즘_과제_6

(Chained Matrix multiplication & Optimal binary tree & Traveling Salesman Problems)

학부:컴퓨터학부

학번:20162518

출석번호 : 156번

이름 : 최승서

14) 5개의 행렬 A1, A2, A3, A4, A5의 곱셉연산이 최적의 해가 나오도록 하는 순서를 찾고 Matrix M과 Matrix P를 구하라

Chained Matrix Multiplication 소스코드

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

int *make_d(int);
int **make_arr(int);
int minmult(int, const int[], int*[]);
void order(int, int, int*[]);
void print_map(int*[], int);


int main(void)
{
   int i;
   int n = 0, ret;
   int *d, **P;


   printf("[Minimum Multiplications algorithm]\n\n");
   printf("Input array number : ");
   scanf("%d", &n);

   d = make_d(n);
   P = make_arr(n + 1);
   ret = minmult(n, d, P);

   printf("\n  [P]");
   print_map(P, n);

   printf("Multiplications order is : ");
   order(1, n, P);
   printf("\nMinimum Multiplications is %d\n", ret);

   free(d);
   for (i = 0; i < n; i++)
      free(P[i]);
   free(P);

   getchar();
   getchar();
   system("clear");
   return 0;
}

int *make_d(int n)  // d배열생성
{
   int i;
   int *d;

   d = (int*)malloc(sizeof(int)*(n + 1));  // n크기만큼동적할당

   for (i = 0; i <= n; i++)
      {
      printf("input d[%d] : ", i);  // d배열의인자를받음
      scanf("%d", &d[i]);
      }
   return d;
```

```
}

int **make_arr(int n)  //2차원배열생성
{
    int i;
    int **Arr;

    Arr= (int**)malloc(sizeof(int*) * n);  //2차원배열동적할당
    for (i = 0; i < n; i++)
       Arr[i] = (int*)malloc(sizeof(int)*n);
    return Arr;
}

int minmult(int n, const int d[], int *P[])  //minimum matrix multiplication
{
    int i, j, k, diagonal;
    int **M, temp = 0, min_k = 0;

    M = make_arr(n + 1);

    for(i = 1; i <= n; i++)
       M[i][i] = 0;

    for (diagonal = 1; diagonal <= n - 1; diagonal++)
       for (i = 1; i <= n - diagonal; i++){
          j = i + diagonal;
          for (k = i; k <= j - 1; k++)
             {
             M[i][j] = M[i][k] + M[k + 1][j] + d[i - 1] * d[k] * d[j];

             if (i == k)
                {
                temp = M[i][j];
                min_k = k;
                }
             else if (M[i][j] > temp)
                M[i][j] = temp;
             else
                min_k = k;
             }
          P[i][j] = min_k;
          }
    printf("\n  [M]");
    print_map(M, n);
    return M[1][n];
}

void order(int i, int j, int *P[])  // 최적의 순서 출력함수
{
    int k;
    if (i == j)
       printf("A[%d]", i);
    else
       {
       k = P[i][j];
       printf("(");
       order(i, k, P);
       order(k + 1, j, P);
       printf(")");
       }
}

void print_map(int *map[], int n)  //map 출력을위한함수
```

```
{
    int i, j, i_cnt = 1, j_cnt = 1;

    for (i = 0; i <= n; i++)
        {
        for (j = 0; j <= n; j++)
            {
            printf("  ");
            if (i == 0)
                {
                if (i_cnt == 1)
                    printf(" ");
                if (i_cnt == n + 1)
                    continue;
                printf("[%d] ", i_cnt++);
                }
            else if (j == 0)
                printf("[%d]", j_cnt++);
            else
                if (i < j)
                    printf("%3d ", map[i][j]);
                else if (i >= j && map[i][j] < 0)
                    printf("%3c ", ' ');
                else
                    printf("%3d ", 0);
            }
        printf("\n");
        }
    printf("\n");
}
```

실행화면

```
[Minimum Multiplications algorithm]

Input array number : 5
input d[0] : 10
input d[1] : 4
input d[2] : 5
input d[3] : 20
input d[4] : 2
input d[5] : 50

  [M]   [1]   [2]   [3]   [4]   [5]
  [1]    0   200   1200   320   1320
  [2]    0     0   400   240   640
  [3]    0     0     0   200   700
  [4]    0     0     0     0   2000
  [5]    0     0     0     0     0


  [P]   [1]   [2]   [3]   [4]   [5]
  [1]    0     1     2     1     4
  [2]    0     0     2     2     4
  [3]    0     0     0     3     4
  [4]    0     0     0     0     4
  [5]    0     0           0     0

Multiplications order is : ((A[1](A[2](A[3]A[4])))A[5])
Minimum Multiplications is 1320
```

24) 주어진 6개의 키값의 최적 이진트리를 작성하시오.

Optimal Binary Search Tree 소스코드

```
//
//  main.c
//  algo_6_OBST
//
//  Created by James Choi on 12/10/2019.
//  Copyright © 2019 James Choi. All rights reserved.
//

#include <stdio.h>
#include <string.h>

#define MAX     100     //  최대 아이템의 수

// 아이템을저장할구조체
typedef struct {
    char  key[10];
    float  p;
} element;

// 트리를만들때노드를구성할구조체
typedef struct _nodetype {
    char key[10];
    struct _nodetype *left;
    struct _nodetype *right;
}nodetype;

// 노드구조체포인터
typedef nodetype* node_pointer;

float minimum(float a[MAX + 2][MAX + 1], int i, int j, int *k);     //
가장적은값을리턴하는함수
float sigma(int i, int j);          // m=i ~ m=j 까지합을구하여값을리턴하는함수
node_pointer tree_node(int i, int j);     // 최적이진트리를구축하는함수

int  n;               // 아이템의개수
element key[MAX + 1];          // 입력받을키구조체
int r[MAX + 2][MAX + 1];           // 배열R이저장될 배열
node_pointer node_root;     // 트리의루트를가리키는구조체포인터변수
float minavg;          //  최소 평균 검색 시간을 저장할 변수

// 최적이진트리검색함수
void search()
{
    int found;
    int result_compare;
    char search_word[10];
    node_pointer p;

    // 검색할키를입력받음
    printf("Input Search Key :");
    scanf("%s", search_word);

    //  루트 노드의 포인터 복사
```

```
    p = node_root;

    found = 0;
    while (1) {
        //  key  가 NULL  이면 while  종료 (찾지 못한 경우)
        if (p->key == NULL)
            break;

            //  검색 할 키와 저장된 키 비교
            result_compare = strcmp(search_word, p->key);

        //  일치 하면 found=1  저장 후 while  문 종료
        if (result_compare == 0) {
            found = 1;
            break;
        }
        else  if (result_compare  <  0)
            p = p->left;                            //  왼쪽 트리 검색
        else
            p = p->right;                            //  오른쪽 트리 검색
    }


    if (found)
        printf("Found  : %s\n", p->key);                            //  찾았을 때 출력
    else
        printf("Not found\n");   //  찾지 못했을 때 출력
}

//  최적 이진 트리 구축을 시작하는 함수
void  make_tree()
{
    node_root = tree_node(1, n);              //  root  노드의 포인터 저장
}

//  최적 이진 탐색 트리 구축 하는 함수 (재귀 호출)
node_pointer  tree_node(int  i, int  j)
{
    int  k;
    node_pointer  a;

    a = (node_pointer)malloc(sizeof(node_pointer));

    k = r[i][j];
    if (k == 0)
        return  NULL;
        else {
            strcpy(a->key, key[k].key);                            //  key  값을 노드로
복사
            a->left = tree_node(i, k - 1);                            //  왼쪽 서브 트리
구축
            a->right = tree_node(k + 1, j);              //  오른쪽 서브 트리 구축

            return  a;                            //  노드의 포인터 반환
        }
}
```

```
//  배열 R  출력
void  print_r()
{
   int  i, j;

   printf("\n배열 R  출력\n");
   for (i = 1; i <= n + 1; i++) {
      for (j = 0; j <= n; j++) {
         printf("[%2d ]\t", r[i][j]);
      }
      printf("\n");
   }
   printf("\n");
}


//  i  <=  k  <=  j  일때 a[i][k-1]  +  a[k+1][j]  값 중 최소값 반환
float  minimum(float  a[MAX + 2][MAX + 1], int  i, int  j, int  *k)
{
   int  tmp_k;
   float  tmp_value;
   float  tmp_value2;

   tmp_k = i;

   //  k=i  일때 계산
   tmp_value2 = a[i][tmp_k - 1] + a[tmp_k + 1][j];
   *k = tmp_k;

   for (tmp_k = i + 1; tmp_k <= j; tmp_k++) {
      tmp_value = a[i][tmp_k - 1] + a[tmp_k + 1][j];
      if (tmp_value  <  tmp_value2) {                          //  최소값인지 검사후
최소값 저장
         tmp_value2 = tmp_value;
         *k = tmp_k;
//  최소값일때의 k  값 저장
      }
   }
   return  tmp_value2;
}

//  m=i  에서 m=j  까지의 Pm  의 합 구하여 반환
float sum(int i, int j){
   float  tmp;
   int  m;

   tmp = 0;
   for (m = i; m <= j; m++)
      tmp += key[m].p;

   return  tmp;
}

//  키와 각 키에 할당될 확률 입력 받음
void init_element()
{
   int  i, j;

   printf("아이템의 개수를 입력하시오 : ");
   scanf("%d", &n);
```

```
    for (i = 1; i <= n; i++) {
        printf("키 입력 (%d)  >>  ", i);
        scanf("%s",key[i].key);
        printf("확률 입력 (%d)  >>  ",i);
        scanf("%f", &key[i].p);
    }

    //  배열 r  의 값을 초기화 (가비지 값 없애기 위함)
    for (i = 0; i <= n + 1; i++)
        for (j = 0; j <= n; j++)
            r[i][j] = 0;
}

          //  최적 이진 검색 트리 구축을 위한 r  배열 구성
void  optsearchtree()
{
    int  i, j, k, diagonal;
    float  a[MAX + 2][MAX + 1] ={0,};

    //  a  배열과 r  배열에 초기값 입력
    for (i = 1; i <= n; i++) {
        a[i][i - 1] = 0;
        a[i][i] = key[i].p;
        r[i][i] = i;
        r[i][i - 1] = 0;
    }

    a[n + 1][n] = 0;
    r[n + 1][n] = 0;

    for (diagonal = 1; diagonal <= n - 1; diagonal++)
        for (i = 1; i <= n - diagonal; i++) {
            j = i + diagonal;
            a[i][j] = minimum(a, i, j, &k) + sum(i, j);           //  일반식 구현
            r[i][j] = k;                        //  최소 값을 주는 k  값 저장
        }
    minavg = a[1][n];
    printf("\n배열 A  출력\n");
    for (i = 1; i <= n ; i++) {
        for (j = 0; j <= n; j++) {
            printf("[%.2lf]\t", a[i][j]);
        }
        printf("\n");
    }

}

          // 메인
int  main(void)
{
    init_element();     // 초기화
    optsearchtree();     // a배열과 r배열구성
    print_r();     // r배열출력
    printf("최소 평균 검색 시간 : %f\n\n",minavg);     // 최소평균검색시간출력
    make_tree();     // 최적이진탐색트리구축
    search();     // 검색
}
```

실행화면

```
아이템의 개수를 입력하시오 : 6
키 입력 (1)  >>  CASE
확률 입력 (1)  >>  0.05
키 입력 (2)  >>  ELSE
확률 입력 (2)  >>  0.15
키 입력 (3)  >>  END
확률 입력 (3)  >>  0.05
키 입력 (4)  >>  IF
확률 입력 (4)  >>  0.35
키 입력 (5)  >>  OF
확률 입력 (5)  >>  0.05
키 입력 (6)  >>  THEN
확률 입력 (6)  >>  0.35

배열 A  출력
[0.00]  [0.05]  [0.25]  [0.35]  [0.95]  [1.05]  [1.80]
[0.00]  [0.00]  [0.15]  [0.25]  [0.80]  [0.90]  [1.65]
[0.00]  [0.00]  [0.00]  [0.05]  [0.45]  [0.55]  [1.30]
[0.00]  [0.00]  [0.00]  [0.00]  [0.35]  [0.45]  [1.20]
[0.00]  [0.00]  [0.00]  [0.00]  [0.00]  [0.05]  [0.45]
[0.00]  [0.00]  [0.00]  [0.00]  [0.00]  [0.00]  [0.35]

배열 R  출력
[ 0 ]   [ 1 ]   [ 2 ]   [ 2 ]   [ 4 ]   [ 4 ]   [ 4 ]
[ 0 ]   [ 0 ]   [ 2 ]   [ 2 ]   [ 4 ]   [ 4 ]   [ 4 ]
[ 0 ]   [ 0 ]   [ 0 ]   [ 3 ]   [ 4 ]   [ 4 ]   [ 4 ]
[ 0 ]   [ 0 ]   [ 0 ]   [ 0 ]   [ 4 ]   [ 4 ]   [ 4 ]
[ 0 ]   [ 0 ]   [ 0 ]   [ 0 ]   [ 0 ]   [ 5 ]   [ 6 ]
[ 0 ]   [ 0 ]   [ 0 ]   [ 0 ]   [ 0 ]   [ 0 ]   [ 6 ]
[ 0 ]   [ 0 ]   [ 0 ]   [ 0 ]   [ 0 ]   [ 0 ]   [ 0 ]

최소 평균 검색 시간 : 1.800000

Input Search Key :END
Found  : END
Program ended with exit code: 0
```

최적 이진 트리는 수기로 작성하였습니다.