

Algorithms assignment #4

응용통계학과 20202850 김지민

1.

- code

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#define MAX 30

int arr[MAX];

int fibo(int n) {
    if (n <= 1)
        return n;


    if (arr[n] > 0)
        return arr[n];

    arr[n] = fibo(n - 1) + fibo(n - 2);
    return arr[n];
}

int main() {
    int num1, num2;
    scanf("%d", &num1);
    while (num1 >= MAX) {
        printf("out of range\n");
        scanf("%d", &num1);
    }
    printf("n = %d : %d\n", num1, fibo(num1));
    scanf("%d", &num2);
    while (num2 >= MAX) {
        printf("out of range\n");
        scanf("%d", &num2);
    }
    printf("n = %d : %d", num2, fibo(num2));

    return 0;
}
```

- result

 Microsoft Visual Studio 디버그 콘솔

```
5
n = 5 : 5
10
n = 10 : 55
C:\Users\jimin.DESKTOP-8V20QSQ\source\repos\Project1\Debug\Project1.exe
)
이 창을 닫으려면 아무 키나 누르세요...
```

2.

- code

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>

int** make_matrix(int num1, int num2) {
    int** matrix = malloc(sizeof(int*) * num1);
    for (int i = 0; i < num1; i++) {
        matrix[i] = malloc(sizeof(int) * num2);
    }
    for (int i = 0; i < num1; i++) {
        for (int j = 0; j < num2; j++) {
            matrix[i][j] = (rand() % 100) + 1;
        }
    }

    return matrix;
}

int minimum_comp(int* num, int n, int** P) {
    int j, q;

    int** m = malloc(sizeof(int*) * n);
    for (int i = 0; i < n; i++) {
        m[i] = malloc(sizeof(int) * n);
    }

    // 0th row and 0th column of m are not used.
    for (int i = 1; i < n; i++)
        m[i][i] = 0;

    // l is chain length.
    for (int l = 2; l < n; l++) {
        for (int i = 1; i < n - l + 1; i++) {
```

```

        j = i + l - 1;
        m[i][j] = INT_MAX;
        for (int k = i; k <= j - 1; k++) {
            q = m[i][k] + m[k + 1][j] + num[i - 1] * num[k] * num[j];
            if (q < m[i][j]) {
                m[i][j] = q;
                P[i][j] = k;
            }
        }
    }
}
return m[1][n - 1];
}

```

```

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

```

```

void print_matrix(int** matrix, int num1, int num2) {
    for (int i = 0; i < num1; i++) {
        for (int j = 0; j < num2; j++) {
            printf("%d ", matrix[i][j]);
        }
        printf("\n");
    }
}

```

```

int** multi_matrix(int** matrix, int** matrix2, int** matrix3, int* p, int** P) {

    int** final = malloc(sizeof(int*) * p[0]);
    for (int i = 0; i < p[0]; i++) {
        final[i] = malloc(sizeof(int) * p[3]);
    }

    int fir, sec, thr;
    if (P[1][3] == 2) {
        int** result = malloc(sizeof(int*) * p[0]);
        for (int i = 0; i < p[0]; i++) {
            result[i] = malloc(sizeof(int) * p[2]);
        }
        fir = p[0];
        sec = p[1];
        thr = p[2];
        for (int i = 0; i < fir; i++) {
            for (int j = 0; j < thr; j++) {
                result[i][j] = 0;
                for (int k = 0; k < sec; k++)
                    result[i][j] += matrix[i][k] * matrix2[k][j];
            }
        }
    }
}

```

```

    fir = p[0];
    sec = p[2];
    thr = p[3];

    for (int i = 0; i < fir; i++) {
        for (int j = 0; j < thr; j++) {
            final[i][j] = 0;
            for (int k = 0; k < sec; k++)
                final[i][j] += result[i][k] * matrix3[k][j];
        }
    }
}
else if (P[1][3] == 1) {
    int** result = malloc(sizeof(int*) * p[1]);
    for (int i = 0; i < p[1]; i++) {
        result[i] = malloc(sizeof(int) * p[3]);
    }
    fir = p[1];
    sec = p[2];
    thr = p[3];
    for (int i = 0; i < fir; i++) {
        for (int j = 0; j < thr; j++) {
            result[i][j] = 0;
            for (int k = 0; k < sec; k++) {
                result[i][j] += matrix2[i][k] * matrix3[k][j];
            }
        }
    }
    fir = p[0];
    sec = p[1];
    thr = p[3];
}

```

```

        for (int i = 0; i < fir; i++) {
            for (int j = 0; j < thr; j++) {
                final[i][j] = 0;
                for (int k = 0; k < sec; k++)
                    final[i][j] += matrix[i][k] * result[k][j];
            }
        }
    }
    else
        return;
    return final;
}

int main() {
    int num1, num2, num3, num4;
    scanf("%d %d %d %d", &num1, &num2, &num3, &num4);
    int num[] = { num1, num2, num3, num4 };
    int size = sizeof(num) / sizeof(num[0]);

    int** matrix = make_matrix(num1, num2);
    int** matrix2 = make_matrix(num2, num3);
    int** matrix3 = make_matrix(num3, num4);

    int** P = malloc(sizeof(int*) * (size-1));
    for (int i = 0; i < size - 1; i++) {
        P[i] = malloc(sizeof(int) * size);
    }
    printf("minimum number of computations : %d\n", minimum_comp(num, 4, P));

    printf("optimal chain order : ");
    order(1, size-1, P);
    printf("\n");


    int** final = multi_matrix(matrix, matrix2, matrix3, num, P);

    printf("output matrix : \n");
    print_matrix(final, num1, num4);

    return 0;
}

```

- result

 Microsoft Visual Studio 디버그 콘솔

```
5 3 7 10
minimum number of computations : 360
optimal chain order : (A1(A2A3))
output matrix :
2068550 2384629 2366254 2893500 1608537 1330228 3093046 2004817 2922379 2118737
1136106 1339470 1471398 1774264 1005758 775968 2099924 1237288 1847406 1194664
3169095 3643731 3408673 4179862 2293719 1938448 4205708 2872718 4165713 3213417
2090940 2392731 2088280 2572964 1389215 1211524 2383402 1751973 2518185 2095021
2906108 3333640 2999252 3686560 2004888 1722608 3547248 2520204 3638416 2927228

C:\Users\jimin\DESKTOP-8V20QSQ\source\repos\Project1\Debug\Project1.exe(프로세스
):
이 창을 닫으려면 아무 키나 누르세요...
```

3.

- code

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>

typedef struct item {
    int item;
    int weight;
    int value;
} item;

int compare(const void* a, const void* b) {
    return (((item*)b)->value / ((item*)b)->weight) - (((item*)a)->value / ((item*)a)->weight));
}

void greedy(struct item* arr, int n, int max_weight) {
    float benefit = 0;
    int left = max_weight;


    for (int i = 0; i < n; i++) {
        // Split it up and put it in.
        if (arr[i].weight > left) {
            int frac = arr[i].value / arr[i].weight * left;
            benefit += frac;
            printf("item index : %d (value/weight %d), fraction number : %d/%d\n",
                arr[i].item, arr[i].value / arr[i].weight, left, arr[i].weight);
            break;
        }
        // There's space to put it in. Put it all of it.
        else {
            left -= arr[i].weight;
            benefit += arr[i].value;
            printf("item index : %d (value/weight %d), fraction number : %d/%d = 1\n",
                arr[i].item, arr[i].value / arr[i].weight, arr[i].weight, arr[i].weight);
            if (left == 0)
                break;
        }
    }
    printf("the maximum value : %.0f\n", benefit);
}

int main() {
    item array[6] = { {1, 6, 60}, {2, 10, 20}, {3, 3, 12}, {4, 5, 80}, {5, 1, 30}, {6, 3, 60} };
    int size = sizeof(array) / sizeof(array[0]);
    int max_weight = 16;
    qsort(array, size, sizeof(array[0]), compare);

    greedy(array, size, max_weight);

    return 0;
}
```

- result

 Microsoft Visual Studio 디버그 콘솔

```
item index : 5 (value/weight 30), fraction number : 1/1 = 1  
item index : 6 (value/weight 20), fraction number : 3/3 = 1  
item index : 4 (value/weight 16), fraction number : 5/5 = 1  
item index : 1 (value/weight 10), fraction number : 6/6 = 1  
item index : 3 (value/weight 4), fraction number : 1/3  
the maximum value : 234
```

```
C:\Users\jimin.DESKTOP-8V2QSQ\source\repos\Project1\Debug\Project1.exe
```