2-1 : median of 3 partitioning

2016025187 김도은

함수

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

void insertionSort(int A[], int start, int end);
int select_pivot(int A[], int left, int right);
void quickSort(int A[], int left, int right);
void threeSort(int A[], int left, int m, int right);
int partition(int A[], int left, int right);

void swap(int* a, int* b){
    int temp = *a;
    *a = *b;
    *b = temp;
}
```

1. 정렬할 원소 중 랜덤하게 3개 값을 고른 뒤 중앙값을 pivot으로 설정하는 select_pivot함수

```c
int select_pivot(int A[], int left, int right){
    srand((unsigned)time(NULL));
//    printf("left : %d, right : %d\n", left, right);
    int random1 = rand()%(right - left + 1) + left;
    int random2 = rand()%(right - left + 1) + left;
    int random3 = rand()%(right - left + 1) + left;
//    printf("random1 : %d. random2 : %d, ramdon3 : %d\n", random1, random2, random3);
//    printf("random1 : %d. random2 : %d, ramdon3 : %d\n", A[random1], A[random2], A[random3]);

    if( (A[random1] < A[random2] && A[random2] < A[random3]) ||
       (A[random3] < A[random2] && A[random2] < A[random1]) ){
//        printf("random2\n\n");
        swap(&A[random2], &A[right]);
    }
    else if( (A[random2] < A[random1] && A[random1] < A[random3]) ||
            (A[random3] < A[random1] && A[random1] < A[random2]) ){
//        printf("random1\n\n");
        swap(&A[random1], &A[right]);
    }
    else{
//        printf("random3\n\n");
        swap(&A[random3], &A[right]);
    }

    return A[right];
}
```

2. Quick sort에 대해서 원소의 개수가 10개 이하일때 insertion sort를 하는 insertionSort함수

```
57  void insertionSort(int A[], int start, int end){
58      int x;
59      for(x=start+1; x<end; x++){
60          int val = A[x];
61          int j = x-1;
62          while(j>=0 && val<A[j]){
63              A[j+1] = A[j];
64              j--;
65          }
66          A[j+1] = val;
67      }
68  }
```

3. Input값 3개에 대해 선행sort를 하는 threeSort함수

```
117  void threeSort(int A[], int left, int m, int right){
118      if(A[left]>A[m]) swap(&A[left], &A[m]);
119      if(A[m]>A[right]) swap(&A[m], &A[right]);
120      if(A[right]>A[m]) swap(&A[left], &A[m]);
121  }
```

4. 연산 속도를 높인 hybrid quick sort를 구현한 quickSort함수는 partition함수를 이용하여 재귀적으로 수행

```
97  void quickSort(int A[], int left, int right){
98      int l = left;
99      int r = right;
100
101     if(l<r){
102         if((r-l) <= 10){
103             insertionSort(A, l, r+1);
104         }
105         else{
106             int p = partition(A, l, r);
107             if(p == -1){
108                 printf("input num must bigger than 3\n");
109                 return;
110             }
111             quickSort(A, l, p-1);
112             quickSort(A, p+1, r);
113         }
114     }
115 }
```

```
int partition(int A[], int left, int right){
    int i, j, pivot;
    /*
     * m = (front + left) / 2 경우 int의 범위를 초과해버리는 경우가 있기때문에
     * 아래와 같이 mid값을 정한다
     * 여기서는 명세 조건이 "각 원소는 0 이상 1000 미만의 값을 가짐" 이기에 상관없다
     */
    int m = left + (right - left) / 2;

    /*
     * 3개 우선정렬
     */
    threeSort(A, left, m, right);


    /*
     * 중간값 퀵소트에서는 우선적으로 3개의 값들에 대해 정렬을 수행한다
     * 만약 정렬대상이 3개 이하라면 위의 3개의 if문을 수행하는 것만으로도 정렬이 모두 끝난다
     * 따라서 데이터의 개수가 3개 초과가 될 때만 추가적인 정렬을 수행한다
     */
    if(right - left + 1 > 3){
        pivot = select_pivot(A, left, right);
        i = left-1;
        j = right;

        for(;;){
            while(A[--j] > pivot);
            while(A[++i] < pivot);
            if(i<j) swap(&A[i], &A[j]);
            else{
                swap(&A[i], &A[right]);
                return i;
            }
        }
    }

    return -1;
}
```

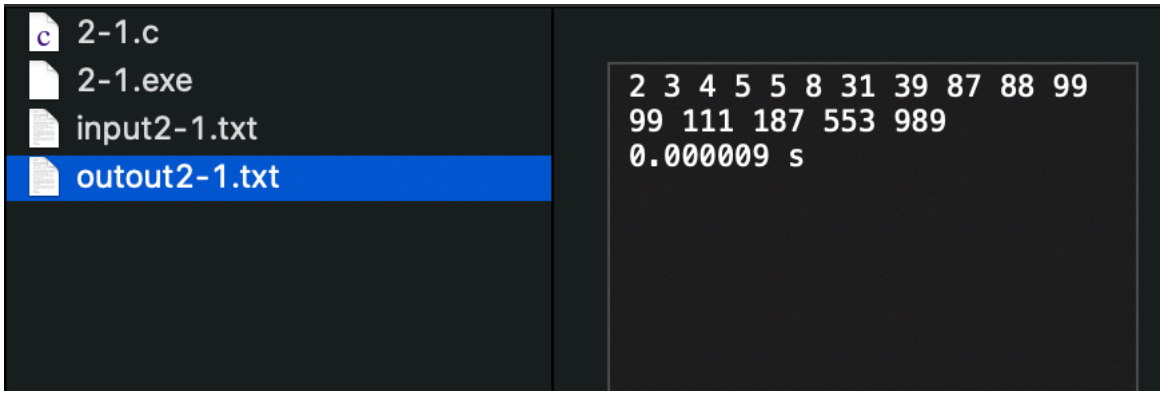
Output

Pivot 설정에 따라 함수 수행시간이 줄어듦을 확인할 수 있다

```
left : 0, right : 15
random1 : 6. random2 : 6, ramdon3 : 12
random1 : 111. random2 : 111, ramdon3 : 4
random3

left : 3, right : 15
random1 : 10. random2 : 15, ramdon3 : 14
random1 : 87. random2 : 99, ramdon3 : 553
random2

result
2 3 4 5 5 8 31 39 87 88 99 99 111 187 553 989
0.000031 s
Program ended with exit code: 0
```

```
left : 0, right : 15
random1 : 14. random2 : 10, ramdon3 : 2
random1 : 553. random2 : 87, ramdon3 : 99
random3

result
2 3 4 5 5 8 31 39 87 88 99 99 111 187 553 989
0.000023 s
Program ended with exit code: 0
```

```
2 3 4 5 5 8 31 39 87 88 99 99 111 187 553 989
0.000017 s
Program ended with exit code: 0
```



Gcc 환경

```
❯ cd Documents/2-2/알 고 /hw/project2/HW2_2016025187

~/Documents/2-2/알 고 /hw/project2/HW2_2016025187
❯ ls
2-1.c         2-1.exe       input2-1.txt  input2-2.txt  outout2-1.txt

~/Documents/2-2/알 고 /hw/project2/HW2_2016025187
❯ ./2-1.exe
```

```
~/Documents/2-2/알 고 /hw/project2/HW2_2016025187
❯ ls
2-1.c         2-1.exe       input2-1.txt  input2-2.txt  outout2-1.txt

~/Documents/2-2/알 고 /hw/project2/HW2_2016025187
❯ gcc -o 2-1.exe 2-1.c

~/Documents/2-2/알 고 /hw/project2/HW2_2016025187
❯ ./2-1.exe

~/Documents/2-2/알 고 /hw/project2/HW2_2016025187
❯ cat output2-1.txt
2 3 4 5 5 8 31 39 87 88 99 99 111 187 553 989
0.000010 s
```

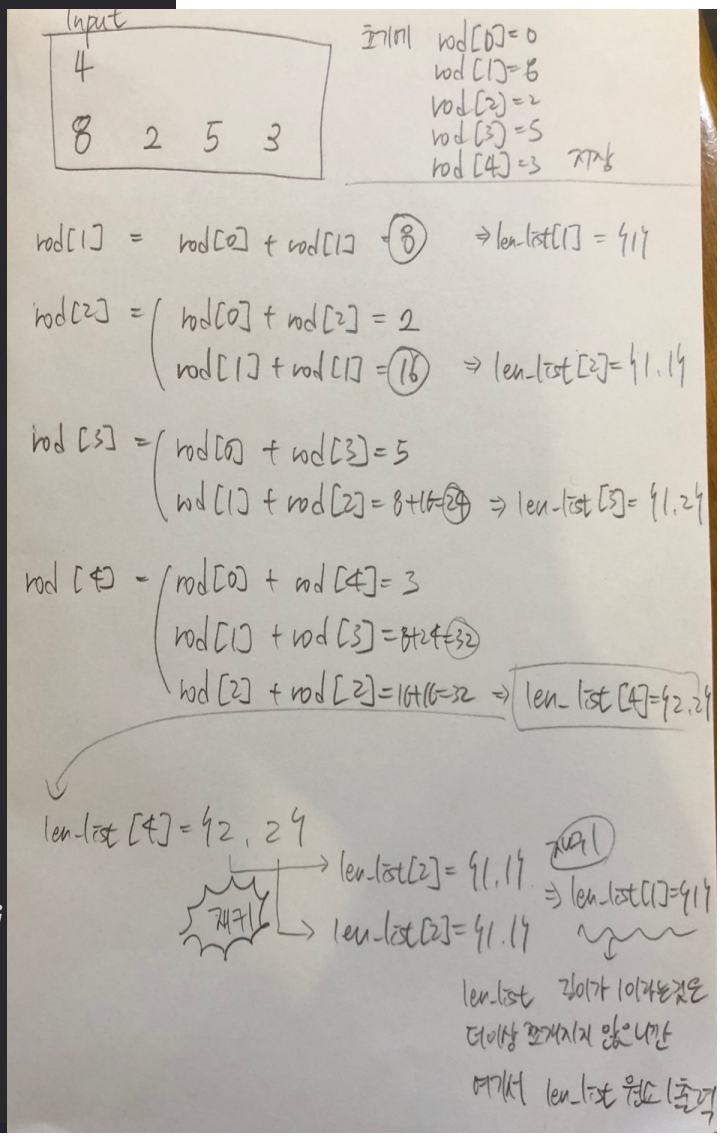
2-2 : rod cutting

2016025187 김도은

함수

1. 다이나믹프로그래밍을 이용하여 답을 구한 후, 트래킹을 통해 언제 최대가 되는지 구한다

```
14  void tracking(vector <vector <int> > v, int len_chunk){
15      if (v[len_chunk].size() == 2){
16          int first_chunk_len = v[len_chunk][0];
17          int second_chunk_len = v[len_chunk][1];
18
19          if (v[first_chunk_len].size() == 1)
20              printf("%d ", first_chunk_len);
21          if (v[second_chunk_len].size() == 1)
22              printf("%d ", second_chunk_len);
23
24          if (v[first_chunk_len].size() == 2)
25              tracking(v, first_chunk_len);
26          if (v[second_chunk_len].size() == 2)
27              tracking(v, second_chunk_len);
28      }
29      printf("\n");
30      return;
31  }
33  void cutRod(int A[], int n){
34      int i, j;
35
36      vector <vector<int> > len_list;
37      len_list.resize(n+1);
38
39      int rod[n+1];
40      rod[0] = 0;
41      for (i = 1; i <= n ; i++){
42          rod[i] = A[i];
43      }
44
45      for(i = 1; i <= n; i++){
46          int maxRod = INT_MIN;
47          int first_chunk_len = 0;
48          int second_chunk_len = 0;
49          for (j = 0; j <= i/2; j++){
50              if (maxRod < rod[j] + rod[i-j]){
51                  maxRod = rod[j] + rod[i-j];
52                  first_chunk_len = j;
53                  second_chunk_len = i - j;
54              }
55          }
56          if (first_chunk_len != 0)
57              len_list[i].push_back(first_chunk_len);
58          if (second_chunk_len != 0)
59              len_list[i].push_back(second_chunk_len);
60          rod[i] = maxRod;
61      }
62
63      printf("%d\n", rod[n]);
64      tracking(len_list, n);
65  }
```



Output

1. 인풋이 4 / 3 3 3 일 때

```c
int main(void) {
    FILE *input = fopen("input2-2.txt", "r");

    int i, rod;
    int array[105];

    fscanf(input, "%d", &rod);
    for(i = 1; i <= rod; i++){
        fscanf(input, "%d ", &array[i]);
    }

    cutRod(array, rod);

    fclose(input);

    return 0;
}
```

input2-2.txt

```
4
10 3 3 3
```

```
40
1 1 1 1

Program ended with exit code: 0
```

2. 인풋이 5 / 1 5 8 9 10 일 때

```c
int main(void) {
    FILE *input = fopen("input2-2.txt", "r");

    int i, rod;
    int array[105];

    fscanf(input, "%d", &rod);
    for(i = 1; i <= rod; i++){
        fscanf(input, "%d ", &array[i]);
    }

    cutRod(array, rod);

    fclose(input);

    return 0;
}
```

input2-2.txt

```
5
1 5 8 9 10
```

```
13
2 3
Program ended with exit code: 0
```

G++ 환경

```
~/Documents/2-2/알고/hw/project2/2-2_/2-2_
❯ g++ -o 2-2.exe 2-2.cpp

~/Documents/2-2/알고/hw/project2/2-2_/2-2_
❯ ./2-2.exe
13
2 3
```