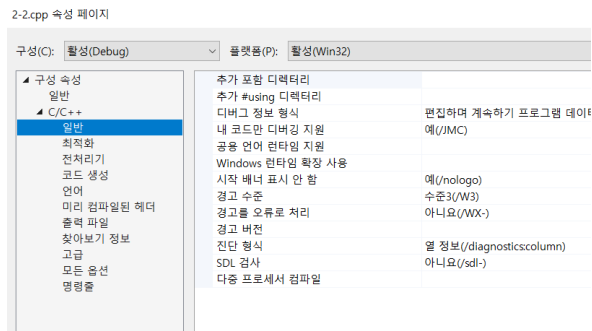


알고리즘 HW2 – hybrid quick sort / rod cutting

2016025469 서건식

컴파일환경: Windows10, Visual Studio 2019

- fprintf, fscanf, fopen 등의 파일 입출력 함수를 사용하였기 때문에



SDL 검사 기능을 '아니요'로 설정 후 컴파일 하였습니다.

2-1:

인풋은 input2-1.txt이고, 아웃풋은 output2-1.txt 입니다.

같은 폴더에 존재한 경우 읽고, 쓰도록 하였고 쓸 때 파일이 없는 경우엔 새로 생성합니다.

Description of function:

```
13 void swap(int* a, int* b) {
14     int temp = *a;
15     *a = *b;
16     *b = temp;
17 }
18
19 int partition(int* arr, int p, int r)
20 {
21     int x = arr[r];
22
23     int i = p - 1, j;
24
25     for (j = p; j < r ; j++) {
26         if (arr[j] <= x) {
27             i = i + 1;
28
29             swap(&arr[i], &arr[j]);
30         }
31     }
32     swap(&arr[i+1], &arr[j]);
33
34     return i + 1;
35 }
36
37
```

- Partition 함수는 강의자료의 수도코드를 참고하여 만들었습니다.
 - 기본적으로 가장 끝의 값을 pivot으로 만들며, r은 후에 randomized 함수를 통해서

랜덤으로 설정된 값과 끝의 값을 swap하여 randomized quicksort를 구현하였습니다.

```
39 int randomized(int* arr, int p, int r) {
40     int a = rand() % (r-p+1) + p;
41     int b = rand() % (r - p + 1) + p;
42     int c = rand() % (r - p + 1) + p;
43     int midindex;
44
45     if (arr[a] >= arr[b])
46         if (arr[b] >= arr[c])
47             midindex = b;
48     else if (arr[a] <= arr[c])
49         midindex = a;
50     else
51         midindex = c;
52     else if (arr[a] > arr[c])
53         midindex = a;
54     else if (arr[b] > arr[c])
55         midindex = c;
56     else
57         midindex = b;
58
59     swap(&arr[r], &arr[midindex]);
60
61     return partition(arr, p, r);
62 }
63
64 void quick_sort(int* arr, int p, int r) {
65     if( p < r ){
66         int q = randomized(arr, p, r);
67         quick_sort(arr, p, q - 1);
68         quick_sort(arr, q+1, r);
69     }
70 }
71
```

- Randomized 함수입니다.

- a,b,c에 각각 p~r까지의 난수를 받습니다. 여기서, 받은 값들은 index입니다. 따라서 뒤에 if문은 index로 접근한 배열의 값을 비교한 다음에, midindex에 그 배열의 값중 중간값의 index를 저장합니다.
- 그 후 끝의 값과 중간값의 자리를 바꿔준 후에, partition 함수를 return 합니다.

- QuickSort

- Quicksort 함수는 수도코드를 보고 참고하였습니다.

```

72 void insertion_sort(int* arr, int n) {
73     int i, j, key;
74
75     for (i = 1; i < n; i++)
76     {
77         key = arr[i];
78
79         for (j = i - 1; j >= 0; j--)
80         {
81             if (arr[j] > key)
82             {
83                 arr[j + 1] = arr[j];
84             }
85             else
86             {
87                 break;
88             }
89         }
90         arr[j + 1] = key;
91     }
92 }
93
94
95

```

● Insertion_sort

- Insertion sort는 인풋의 개수가 10이하일 때만 작동합니다. 수도코드를 보고 참고하였습니다.

```

97 int main() {
98     srand((unsigned)time(NULL)); // 매번 새로운 random value 를 받기위한 구문
99
100     FILE* file = fopen("input2-1.txt", "r");
101     FILE* outfile = fopen("output2-1.txt", "w");
102
103     int* arr = new int[200000]; // 총 20만개의 array 동적할당
104
105     if (file == NULL) {
106         return 0;
107     }
108
109     while (!feof(file)) { // 배열에 값 저장
110         fscanf(file, "%d ", &arr[length]);
111
112         if (arr[length] < 0 || arr[length] > 10000) { //0 이상 10000미만의 값을 가지게 하는 조건
113             return 0;
114         }
115
116         length++;
117     }
118

```

● Main

- srand함수는 컴파일 할 때 마다 새로운 random value를 받기위해 사용하였습니다.

- 파일 입출력을 위해 FILE 구조체 변수를 선언하였습니다.
- 배열의 크기는 20만개로 동적할당 하였습니다.
- 배열에 값을 저장할 때는 파일 끝까지 검사하면서, arr에 넣어줍니다. length는 전역 변수로 0으로 초기화 되어있고, 값을 넣을 때 마다 증가시킵니다. 다만 넣었을 때 과제의 조건에 명시된 인풋값의 범위를 벗어나게 되면 프로그램을 종료합니다.

```

120 if (3 <= length && length <= 10) { // 원소가 3이상 10이하일때 insertion sort
121     clock_t start, end;
122     float delay;
123     start = clock();
124     insertion_sort(arr, length); // 시간을 잴 함수
125     end = clock();
126     delay = (float)(end - start) / CLOCKS_PER_SEC;
127
128     for (int i = 0; i < length; i++)
129     {
130         fprintf(outfile, "%d ", arr[i]);
131     }
132     fprintf(outfile, "%s", "\n");
133     fprintf(outfile, "%.3f s", delay);
134
135 }
136
137 else if (length < 3 || length > 200000) { //원소의 갯수 조건을 벗어날 때 프로그램 종료
138     return 0;
139 }
140 else
141 {
142     clock_t start, end;
143     float delay;
144     start = clock();
145     quick_sort(arr, 0, length - 1); // 시간을 잴 함수
146     end = clock();
147     delay = (float)(end - start) / CLOCKS_PER_SEC; // 나머지는 퀵소트로 돌린다.
148
149     for (int i = 0; i < length; i++)
150     {
151         fprintf(outfile, "%d ", arr[i]);
152     }
153
154     fprintf(outfile, "%s", "\n");
155     fprintf(outfile, "%.3f s", delay);
156
157 }
158
159
160
161 fclose(file);
162 fclose(outfile);
163
164 arr = NULL;
165 delete[] arr;
166
167
168 }

```

- 들어온 원소의 값이 과제에 명시된 조건에 맞춰, 3이상 10이하일 때는 insertion sort를 실행합니다. 여기서 그 함수의 시간을 측정합니다. 출력은 먼저 정렬된 arr의 길

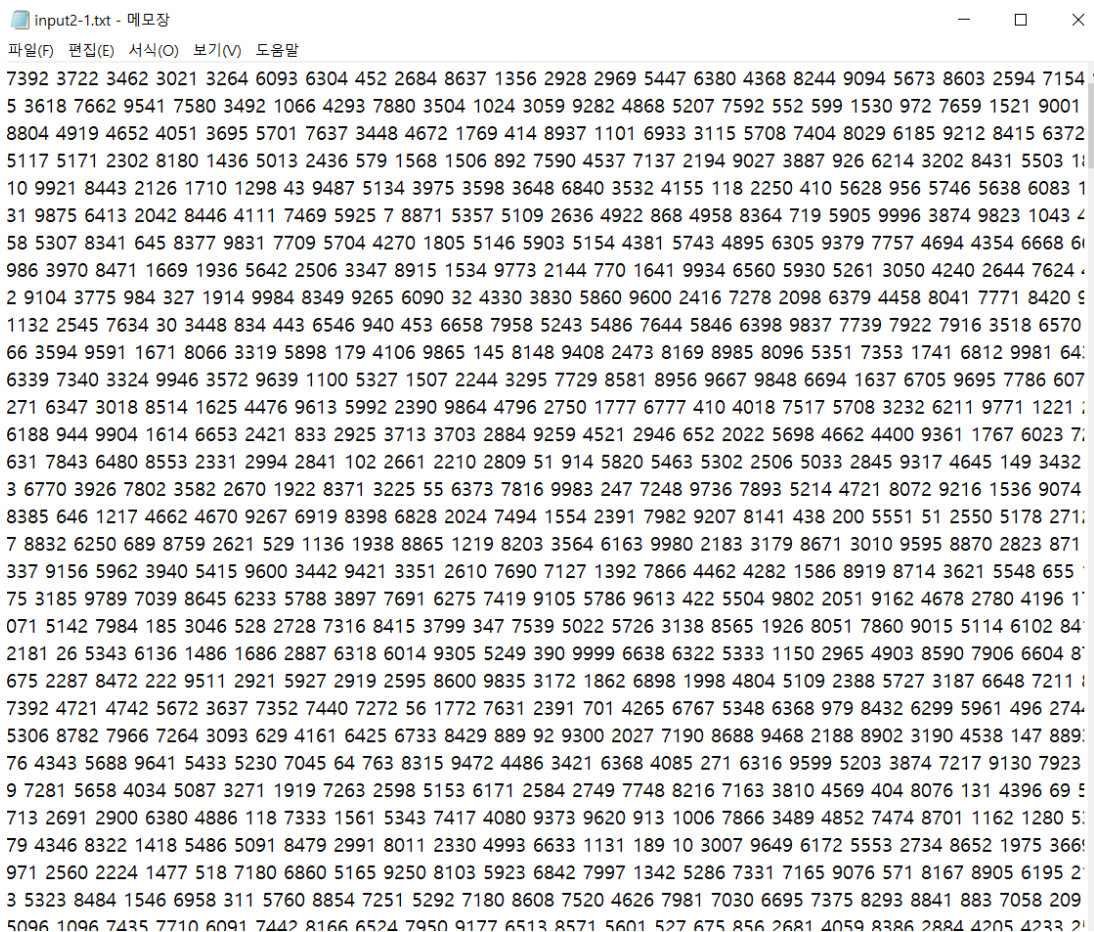
이만큼 txt파일에 쓰고 난 후에, 개행을 입력하고 delay를 출력합니다. 단, 소수점 3 자리 까지 출력합니다.

- 원소의 개수 조건을 벗어날 때는 프로그램을 종료합니다.

- 나머지의 경우에는 quicksort를 시행합니다.

- 제 환경에서는 exe파일과 input파일이 같은 폴더에 있을 때 양식에 맞춰 input을 넣어주면 output파일에 제대로 결과가 쓰여지는 것을 확인하였습니다. 10만개의 input을 넣었을 때의 결과입니다.

- input



input2-1.txt - 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말

7392 3722 3462 3021 3264 6093 6304 452 2684 8637 1356 2928 2969 5447 6380 4368 8244 9094 5673 8603 2594 7154
5 3618 7662 9541 7580 3492 1066 4293 7880 3504 1024 3059 9282 4868 5207 7592 552 599 1530 972 7659 1521 9001
8804 4919 4652 4051 3695 5701 7637 3448 4672 1769 414 8937 1101 6933 3115 5708 7404 8029 6185 9212 8415 6372
5117 5171 2302 8180 1436 5013 2436 579 1568 1506 892 7590 4537 7137 2194 9027 3887 926 6214 3202 8431 5503 11
10 9921 8443 2126 1710 1298 43 9487 5134 3975 3598 3648 6840 3532 4155 118 2250 410 5628 956 5746 5638 6083 1
31 9875 6413 2042 8446 4111 7469 5925 7 8871 5357 5109 2636 4922 868 4958 8364 719 5905 9996 3874 9823 1043 4
58 5307 8341 645 8377 9831 7709 5704 4270 1805 5146 5903 5154 4381 5743 4895 6305 9379 7757 4694 4354 6668 6
986 3970 8471 1669 1936 5642 2506 3347 8915 1534 9773 2144 770 1641 9934 6560 5930 5261 3050 4240 2644 7624
2 9104 3775 984 327 1914 9984 8349 9265 6090 32 4330 3830 5860 9600 2416 7278 2098 6379 4458 8041 7771 8420 9
1132 2545 7634 30 3448 834 443 6546 940 453 6658 7958 5243 5486 7644 5846 6398 9837 7739 7922 7916 3518 6570
66 3594 9591 1671 8066 3319 5898 179 4106 9865 145 8148 9408 2473 8169 8985 8096 5351 7353 1741 6812 9981 64
6339 7340 3324 9946 3572 9639 1100 5327 1507 2244 3295 7729 8581 8956 9667 9848 6694 1637 6705 9695 7786 607
271 6347 3018 8514 1625 4476 9613 5992 2390 9864 4796 2750 1777 6777 410 4018 7517 5708 3232 6211 9771 1221
6188 944 9904 1614 6653 2421 833 2925 3713 3703 2884 9259 4521 2946 652 2022 5698 4662 4400 9361 1767 6023 7
631 7843 6480 8553 2331 2994 2841 102 2661 2210 2809 51 914 5820 5463 5302 2506 5033 2845 9317 4645 149 3432
3 6770 3926 7802 3582 2670 1922 8371 3225 55 6373 7816 9983 247 7248 9736 7893 5214 4721 8072 9216 1536 9074
8385 646 1217 4662 4670 9267 6919 8398 6828 2024 7494 1554 2391 7982 9207 8141 438 200 5551 51 2550 5178 271
7 8832 6250 689 8759 2621 529 1136 1938 8865 1219 8203 3564 6163 9980 2183 3179 8671 3010 9595 8870 2823 871
337 9156 5962 3940 5415 9600 3442 9421 3351 2610 7690 7127 1392 7866 4462 4282 1586 8919 8714 3621 5548 655
75 3185 9789 7039 8645 6233 5788 3897 7691 6275 7419 9105 5786 9613 422 5504 9802 2051 9162 4678 2780 4196 1
071 5142 7984 185 3046 528 2728 7316 8415 3799 347 7539 5022 5726 3138 8565 1926 8051 7860 9015 5114 6102 84
2181 26 5343 6136 1486 1686 2887 6318 6014 9305 5249 390 9999 6638 6322 5333 1150 2965 4903 8590 7906 6604 8
675 2287 8472 222 9511 2921 5927 2919 2595 8600 9835 3172 1862 6898 1998 4804 5109 2388 5727 3187 6648 7211
7392 4721 4742 5672 3637 7352 7440 7272 56 1772 7631 2391 701 4265 6767 5348 6368 979 8432 6299 5961 496 274
5306 8782 7966 7264 3093 629 4161 6425 6733 8429 889 92 9300 2027 7190 8688 9468 2188 8902 3190 4538 147 889
76 4343 5688 9641 5433 5230 7045 64 763 8315 9472 4486 3421 6368 4085 271 6316 9599 5203 3874 7217 9130 7923
9 7281 5658 4034 5087 3271 1919 7263 2598 5153 6171 2584 2749 7748 8216 7163 3810 4569 404 8076 131 4396 69 5
713 2691 2900 6380 4886 118 7333 1561 5343 7417 4080 9373 9620 913 1006 7866 3489 4852 7474 8701 1162 1280 5
79 4346 8322 1418 5486 5091 8479 2991 8011 2330 4993 6633 1131 189 10 3007 9649 6172 5553 2734 8652 1975 366
971 2560 2224 1477 518 7180 6860 5165 9250 8103 5923 6842 7997 1342 5286 7331 7165 9076 571 8167 8905 6195 2
3 5323 8484 1546 6958 311 5760 8854 7251 5292 7180 8608 7520 4626 7981 7030 6695 7375 8293 8841 883 7058 209
5096 1096 7435 7710 6091 7442 8166 6524 7950 9177 6513 8571 5601 527 675 856 2681 4059 8386 2884 4205 4233 21

- output

[illegible]

시간은 제대로 출력 됩니다.

2-2:

인풋은 input2-2.txt이며, 같은 폴더에 있어야 합니다. 출력은 stdout입니다.

```

0
7 void bottom_up_cutrod(int* r, int* s, int* p, int n) {
8     int q;
9
10    for (int i = 0; i < n; i++)
11    {
12        r[i] = 0;
13        s[i] = 0;
14    }
15
16
17    for (int j = 1; j <= n; j++)
18    {
19        q = INT_MIN;
20
21        for (int i = 1; i <= j ; i++)
22        {
23            if (q < p[i] + r[j - i])
24            {
25                q = p[i] + r[j - i];
26                s[j] = i;
27            }
28        }
29        r[j] = q;
30    }
31
32
33
34 }
35

```

- extended-bottom_up_cut_rod

- 강의자료의 수도코드를 참고하였습니다.
- r과 s 배열은 각각 i번째 의 최대 이익값, 최대 이익값 일때의 막대의 길이입니다.
- main함수에서 선언하여 call by reference로 구현하였습니다.
- INT_MIN은 헤더파일 limit.h에서 가져온 값이며, Int값중 가장 작은 값을 저장한 상수입니다.

```


39
40 int main() {
41     int n; // Rod 의 길이
42     int i = 1;
43     int* p = new int[101];
44     p[0] = NULL;
45     FILE* file = fopen("input2-2.txt", "r");
46
47     if (file == NULL) {
48         return 0;
49     }
50
51     fscanf(file, "%d\n", &n);
52     int* r = new int[n];
53     int* s = new int[n];
54
55
56     while (!feof(file)) { // 배열에 값 저장
57         fscanf(file, "%d ", &p[i]);
58         i++;
59     }
60
61     bottom_up_cutrod(r, s, p, n);
62     cout << r[n] << endl;
63
64     while (n > 0)
65     {
66         cout << s[n] << " ";
67         n = n - s[n];
68     }
69
70     fclose(file);
71     p = NULL; r = NULL; s = NULL;
72     delete[] p;
73     delete[] r;
74     delete[] s;
75
76     return 0;
77 }
78

```

- i는 1부터 시작합니다. P 배열(길이 별 값 저장)은 과제에서 100개까지 명시하였기 때문에 101개까지 할당했습니다. (i가 1부터 시작하기 때문) p[0]엔 NULL을 넣었습니다.
- 처음 값은 n에 저장합니다. 저장한 n을 바탕으로 r과 s 배열을 선언합니다.
- 파일의 끝까지 p배열에 값을 저장합니다.
- r[n]으로 최대 이익 값을 출력하고, 현재 길이에서 최대 이익 값일 때의 막대의 길이를 뺀 값을 n이 0이 될 때 까지 출력해줍니다.

- exe파일에서는 출력을 stdout으로 해서 제 환경에서는 바로 꺼지는 문제가 있습니다.
Visual studio 환경에서 직접 컴파일 하면 값이 나옵니다.

- Input

 input2-2.txt - 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

8

1 5 8 9 10 17 17 20

- Output

```
Microsoft Visual Studio 디버그 콘솔
22
2 6
C:\Users\iqeq2\OneDrive\바탕 화면\알고리즘\이 창을 닫으려면 아무 키나 누르세요.
```