

객체 지향 프로그래밍 ASSN1

20180285 이신범

무은재학부

nm160

나는 이 프로그래밍 과제를
다른 사람의 부적절한 도움 없이
수행하였습니다.
이신범

나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다.

(1) 개요

어싸인 1은 기존의 C 언어를 능숙하게 사용하여 제시된 과제를 해결할 수 있는지를 판단하는 기초적인 프로그래밍 과제였다고 생각한다.

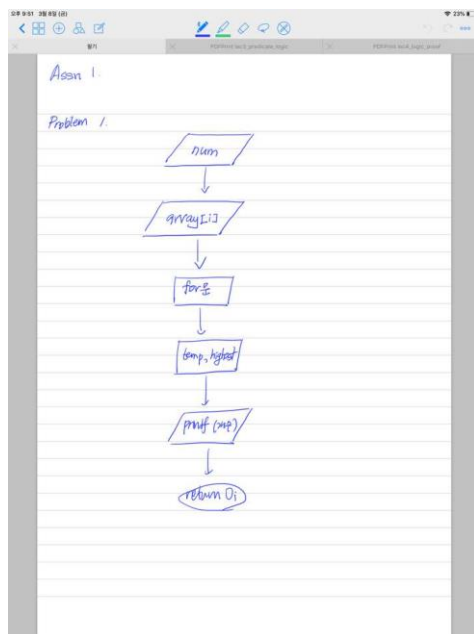
1번 문제에서는 단순한 배열을 이용하여 나무막대의 개수를 `sum ++` 하며 구했다.

2번 문제에서는 number sequence에 대해서 longest increasing subsequence 를 찾기 위해서 <algorithm> 헤더 파일에 포함되어 있는 `lower_bound` 함수를 이용했다.

3번 문제에서는 connected component 를 찾기 위해서 BFS 를 이용했다. 이 문제는 약간 작년 프로그래밍과 문제해결 과제의 assn3 이었던 '애니팡 만들기' 와 비슷한 느낌을 받았다. (서로 연결되어 있는 덩어리를 컴퓨터가 인식할 수 있도록 해야 하기 때문이다.)

(2) 알고리즘 및 설명

Prob 1)



입력 : `num` (나무 막대의 개수), `array[i]` (나무 막대의 높이가 저장된 배열)

출력 : `sum` (왼쪽에서 봤을 때 보이는 나무막대 + 오른쪽에서 봤을 때)

```

Microsoft Visual Studio 디버그 콘솔
7
4 6 5 8 6 7 1
6
C:\Users\POSTECH20180285\source\repos\data_assn\Debug\data_assn.exe(37628 프로세스)이(가) 0 코드로 인해 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요.

```

왼쪽에서 보았을 때는 4 6 8 이 보이고

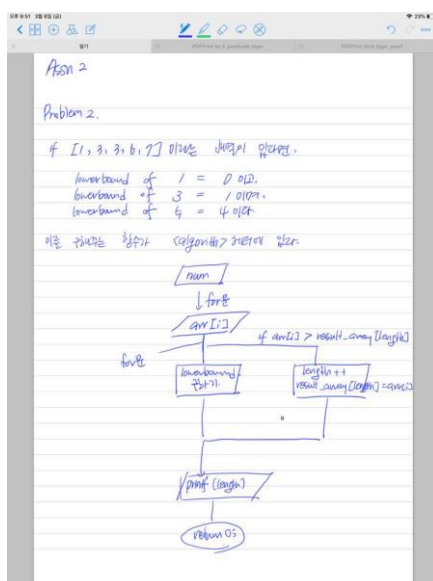
오른쪽에서 보았을 때는 1 7 8 이 보이므로, 두 개를 더하면 총 6이 나와야 한다.

<알고리즘>

highest 라는 변수를 두고, 배열의 원소들을 비교한다.

만약 highest 보다 배열의 원소가 더 크다면, 그 나무막대는 앞에있는 나무막대보다 길다는 것이므로 보일 것이다. 따라서 `sum ++` 를 해주고 `highest = array[i]` 가 된다는 것을 이용하면 된다.

Prob 2)



입력 : num (나무 막대의 개수), arr[i] (나무 막대의 길이 값)

출력 : length (가장 증가 배열의 길이)

100만개 이상의 배열을 이용해 longest increasing subsequence 를 빠르게 구하기 위해서는 시간복잡도가 $(N \log N)$ 인 알고리즘을 이용해야 한다.

이 알고리즘은 배열 $L[i]$ 를 이용한다.

$L[i]$ = 길이가 i 인 가장 증가 배열을 만들었을 때 마지막 원소 중 가장 작은 값

따라서, length of L 이 LIS 의 길이이다.

만약, $arr[i]$ 가 L 의 마지막 원소보다 크다면, L 의 뒤에 $arr[i]$ 를 추가하고 $length++$ 을 해 준다.

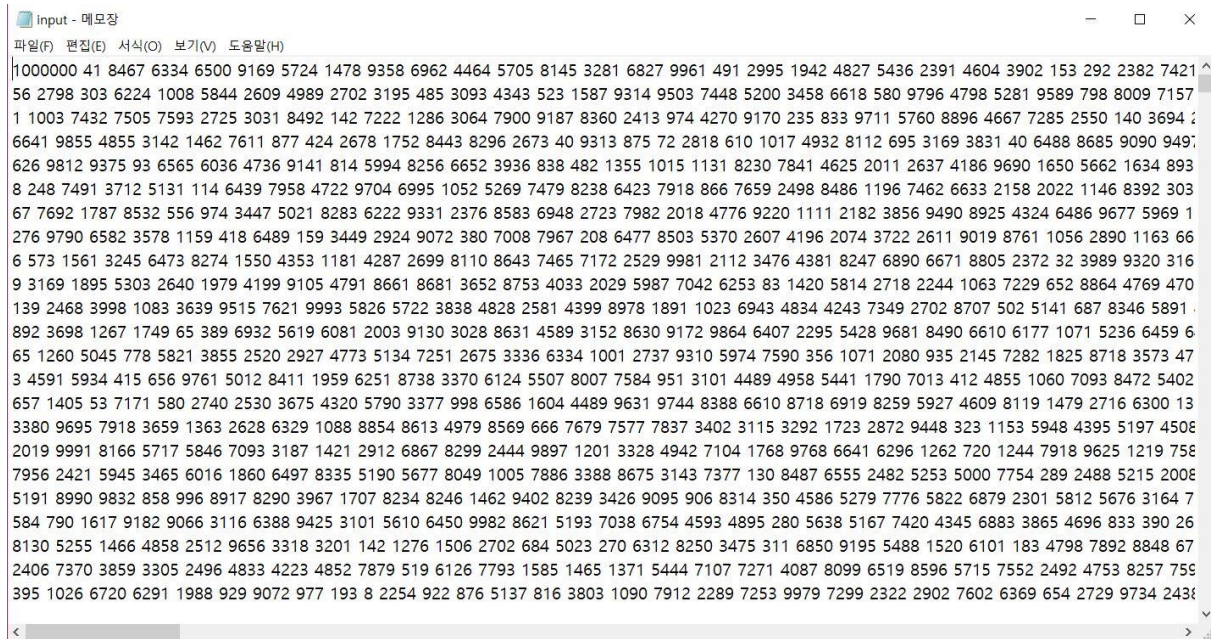
만약, $arr[i]$ 가 L 의 마지막 원소보다 작다면, $arr[i]$ 의 lower bound 를 찾아서 그자리를 $arr[i]$ 로 바꿔준다.

Lower bound 에 대한 설명은 위의 그림에 첨부해 놓았으니 명시하지 않겠다.

Lower bound를 구해주는 함수는 <algorithm>에 lower_bound 라는 이름으로 존재한다.

따라서, $lower_bound(arr, arr+length, 값)$ 의 형식을 이용해 lower_bound 를 구한다.

100만개의 나무막대가 주어졌을 때, 값을 10초 안에 구할 수 있는지를 알아보기 위해 파일입출력을 이용해 100만개의 난수를 생성하고 프로그램을 실행해보았을 때의 결과이다.

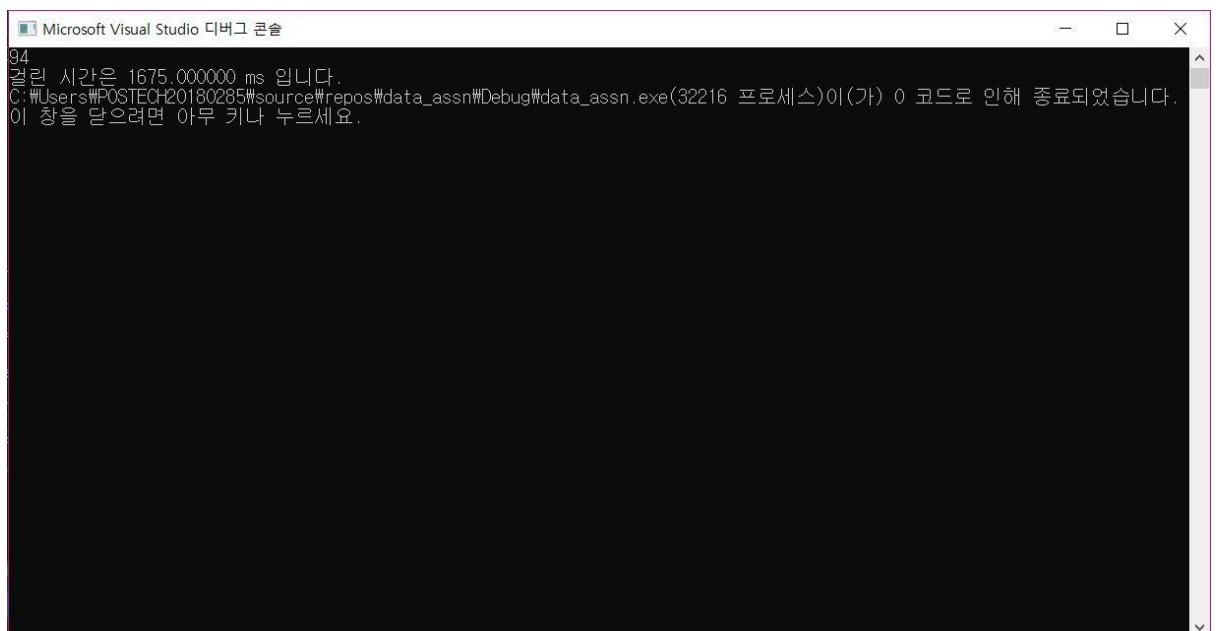


input - 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
1000000 41 8467 6334 6500 9169 5724 1478 9358 6962 4464 5705 8145 3281 6827 9961 491 2995 1942 4827 5436 2391 4604 3902 153 292 2382 7421
56 2798 303 6224 1008 5844 2609 4989 2702 3195 485 3093 4343 523 1587 9314 9503 7448 5200 3458 6618 580 9796 4798 5281 9589 798 8009 7157
1 1003 7432 7505 7593 2725 3031 8492 142 7222 1286 3064 7900 9187 8360 2413 974 4270 9170 235 833 9711 5760 8896 4667 7285 2550 140 3694
6641 9855 4855 3142 1462 7611 877 424 2678 1752 8443 8296 2673 40 9313 875 72 2818 610 1017 4932 8112 695 3169 3831 40 6488 8685 9090 949
626 9812 9375 93 6565 6036 4736 9141 814 5994 8256 6652 3936 838 482 1355 1015 1131 8230 7841 4625 2011 2637 4186 9690 1650 5662 1634 893
8 248 7491 3712 5131 114 6439 7958 4722 9704 6995 1052 5269 7479 8238 6423 7918 866 7659 2498 8486 1196 7462 6633 2158 2022 1146 8392 303
67 7692 1787 8532 556 974 3447 5021 8283 6222 9331 2376 8583 6948 2723 7982 2018 4776 9220 1111 2182 3856 9490 8925 4324 6486 9677 5969 1
276 9790 6582 3578 1159 418 6489 159 3449 2924 9072 380 7008 7967 208 6477 8503 5370 2607 4196 2074 3722 2611 9019 8761 1056 2890 1163 66
6 573 1561 3245 6473 8274 1550 4353 1181 4287 2699 8110 8643 7465 7172 2529 9981 2112 3476 4381 8247 6890 6671 8805 2372 32 3989 9320 316
9 3169 1895 5303 2640 1979 4199 9105 4791 8661 8681 3652 8753 4033 2029 5987 7042 6253 83 1420 5814 2718 2244 1063 7229 652 8864 4769 470
139 2468 3998 1083 3639 9515 7621 9993 5826 5722 3838 4828 2581 4399 8978 1891 1023 6943 4834 4243 7349 2702 8707 502 5141 687 8346 5891
892 3698 1267 1749 65 389 6932 5619 6081 2003 9130 3028 8631 4589 3152 8630 9172 9864 6407 2295 5428 9681 8490 6610 6177 1071 5236 6459 6
65 1260 5045 778 5821 3855 2520 2927 4773 5134 7251 2675 3336 6334 1001 2737 9310 5974 7590 356 1071 2080 935 2145 7282 1825 8718 3573 47
3 4591 5934 415 656 9761 5012 8411 1959 6251 8738 3370 6124 5507 8007 7584 951 3101 4489 4958 5441 1790 7013 412 4855 1060 7093 8472 5402
657 1405 53 7171 580 2740 2530 3675 4320 5790 3377 998 6586 1604 4489 9631 9744 8388 6610 8718 6919 8259 5927 4609 8119 1479 2716 6300 13
3380 9695 7918 3659 1363 2628 6329 1088 8854 8613 4979 8569 666 7679 7577 7837 3402 3115 3292 1723 2872 9448 323 1153 5948 4395 5197 4508
2019 9991 8166 5717 5846 7093 3187 1421 2912 6867 8299 2444 9897 1201 3328 4942 7104 1768 9768 6641 6296 1262 720 1244 7918 9625 1219 758
7956 2421 5945 3465 6016 1860 6497 8335 5190 5677 8049 1005 7886 3388 8675 3143 7377 130 8487 6555 2482 5253 5000 7754 289 2488 5215 2008
5191 8990 9832 858 996 8917 8290 3967 1707 8234 8246 1462 9402 8239 3426 9095 906 8314 350 4586 5279 7776 5822 6879 2301 5812 5676 3164 7
584 790 1617 9182 9066 3116 6388 9425 3101 5610 6450 9982 8621 5193 7038 6754 4593 4895 280 5638 5167 7420 4345 6883 3865 4696 833 390 26
8130 5255 1466 4858 2512 9656 3318 3201 142 1276 1506 2702 684 5023 270 6312 8250 3475 311 6850 9195 5488 1520 6101 183 4798 7892 8848 67
2406 7370 3859 3305 2496 4833 4223 4852 7879 519 6126 7793 1585 1465 1371 5444 7107 7271 4087 8099 6519 8596 5715 7552 2492 4753 8257 758
395 1026 6720 6291 1988 929 9072 977 193 8 2254 922 876 5137 816 3803 1090 7912 2289 7253 9979 7299 2322 2902 7602 6369 654 2729 9734 2438
```

Input.txt 파일을 읽어서 프로그램을 시행해보았다.



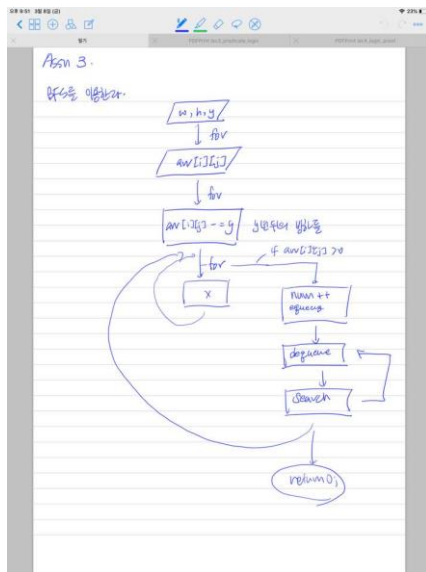
Microsoft Visual Studio 디버그 콘솔

```
94
결린 시간은 1675.000000 ms 입니다.
C:\Users\POSTECH20180285\source\repos\data_assn\Debug\data_assn.exe(32216 프로세스)이(가) 0 코드로 인해 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요.
```

1675ms (1.675초) 만에 프로그램이 올바르게 시행되었다 !

[나무막대의 길이는 0 ~ 10000 으로 했던 것으로 기억한다]

Prob 3)



입력 : W (가로) , H(세로) , Y(연 수)

출력 : num (빙하의 개수)

BFS 는 그래프를 탐색하는 방법 중 하나이다.

“너비 우선 탐색” 이라는 이름을 갖는데, 모든 정점을 한번만 방문하되, 그 깊이가 같은 정점을 우선적으로 탐색하는 과정이다.

<https://mygumi.tistory.com/102?category=677288>

이 블로그에 포함되어 있는 그림을 보면, DFS 와 BFS 의 차이를 한 눈에 알 수 있다.

너비 우선 탐색 (BFS) 은 다음의 차례를 따른다.

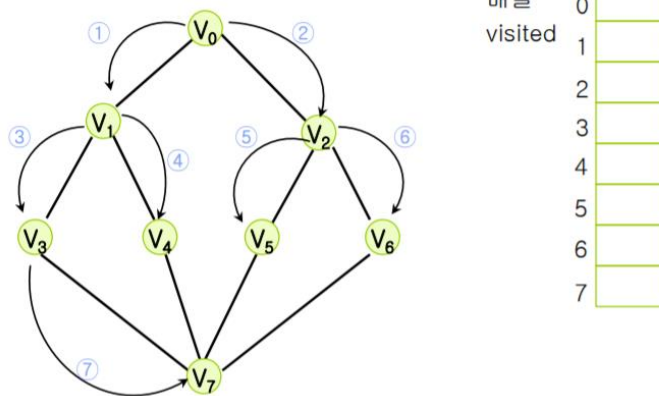
1. 하나의 노드를 택한다
2. 노드를 방문해 필요한 작업을 한 뒤 연결된 다음 노드를 찾아 큐에 저장한다. 더 이상 방문할 노드가 없으면 3단계로 간다.

3. 큐의 맨 앞 노드를 빼내 2단계를 반복한다. (큐가 빌 때 까지 계속한다.)

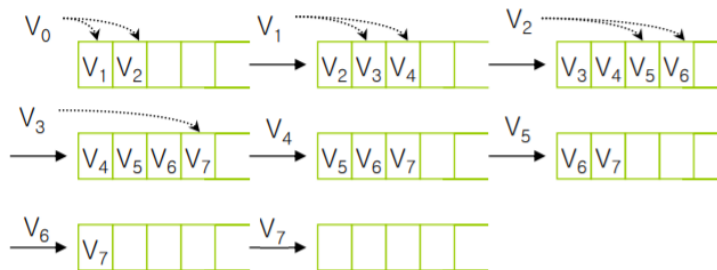
다음의 그림을 본다면 이해가 빠를 것이다. BFS는 '큐'를 이용한다.

너비우선탐색(breadth first search) 예제

너비우선탐색의 예



너비 우선 탐색의 예 - 큐의 모양



위의 방법을 이용해 queue, dequeuer, isEmpty, search 함수를 구현했다.

프로그램이 올바르게 작동하는지, 빠르게 작동하는지 알아보기 위해 파일입출력을 통해 임의의 W H Y 와 임의의 빙하값을 가진 Text.txt 파일을 통해 프로그램을 시행해보았다.

```
Text - 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
603 980 37
41 67 34 0 69 24 78 58 62 64 5 45 81 27 61 91 95 42 27 36 91 4 2 53 92 82 21 16 18 95 47 26 71 38 69 12 67 99 35 94 3 11 22 33 73 64 41 11 53 68
81 44 96 22 29 61 35 50 73 66 44 59 92 39 53 24 54 10 45 49 86 13 74 22 68 18 87 5 58 91 2 25 77 14 14 24 34 74 72 59 33 70 87 97 18 77 73 70 63
75 59 25 21 70 26 34 5 83 50 98 79 1 93 34 37 34 56 93 76 5 62 48 81 0 13 41 55 55 42 62 11 77 24 78 52 43 96 73 40 13 75 72 18 10 17 32 12 95 65
51 41 15 79 10 98 73 88 77 32 56 89 13 8 41 90 23 63 28 84 78 0 71 85 74 71 33 67 53 95 68 25 76 29 50 98 9 93 86 80 16 49 67 28 79 64 21 5 26 16
72 68 58 31 68 26 10 22 74 79 10 52 82 91 95 64 74 64 2 55 60 74 72 21 22 47 77 89 5 95 94 50 43 54 81 12 72 39 28 12 62 67 8 15 8 23 59 34 4 86
73 22 48 51 86 44 46 77 17 29 16 74 91 69 12 46 93 91 15 49 57 40 52 36 51 87 26 62 55 83 94 80 97 65 65 13 61 78 78 78 40 11 47 45 70 75 89 50
13 68 51 16 79 68 40 31 33 79 63 59 53 36 95 65 74 20 35 80 76 55 25 71 8 59 56 2 32 5 40 75 62 85 62 80 36 97 2 8 80 40 76 58 93 40 46 74 73 97
9 23 26 3 48 51 70 36 58 67 56 5 31 62 19 75 56 31 95 44 91 3 88 15 50 19 72 91 83 33 75 42 69 0 54 78 85 57 33 93 90 3 50 33 22 71 11 90 38 41 5
60 52 58 33 55 65 26 1 34 59 93 48 73 92 93 33 37 3 90 11 94 68 27 98 89 48 79 47 49 31 69 75 98 49 36 99 23 43 31 54 61 34 85 67 16 93 4 66 53 8
43 73 66 51 75 28 32 61 69 3 29 7 92 51 53 92 70 19 72 7 94 5 40 83 62 9 7 38 60 71 71 54 47 14 23 29 89 66 87 56 32 70 39 66 22 83 60 53 5 91 59
33 9 57 17 34 53 35 58 93 43 93 66 51 40 94 37 74 12 97 51 95 17 92 92 37 30 13 96 10 89 28 9 86 44 40 82 69 20 93 92 62 14 27 7 94 5 13 11 13 93
17 46 93 87 21 12 67 99 44 97 1 28 42 4 68 68 41 96 62 20 44 18 25 19 84 74 84 59 49 14 95 3 32 51 37 14 63 51 19 90 17 83 18 84 52 86 69 48 37 5
51 29 37 33 0 64 83 57 68 10 68 8 41 32 37 84 99 1 69 83 56 92 40 87 12 10 20 35 63 15 97 77 74 34 53 33 78 86 24 34 75 49 12 84 24 92 4 22 55 13
23 32 56 96 10 76 92 50 90 95 52 50 15 36 61 60 84 29 34 50 82 6 36 84 90 17 82 66 16 88 25 1 10 50 82 21 93 38 54 93 95 80 38 67 20 45 83 65 96
84 23 70 12 50 75 11 50 95 88 20 1 83 98 92 48 84 45 87 62 43 58 59 71 16 72 32 41 83 88 15 76 16 98 39 83 5 97 46 69 75 17 87 20 30 42 35 89 73
4 15 44 36 69 78 76 58 52 46 37 15 88 16 88 82 74 33 15 56 56 62 16 24 29 42 84 72 49 96 26 82 2 75 16 65 52 59 55 17 44 54 43 24 99 22 21 33 13
52 69 62 70 11 35 90 11 64 50 60 45 67 54 87 17 65 42 88 73 15 79 37 88 69 86 30 9 40 21 19 81 26 20 11 54 93 84 57 99 36 29 50 57 5 65 34 49 94
4 19 79 95 58 7 45 74 38 56 20 32 66 22 57 53 71 55 89 25 29 28 39 27 94 90 48 54 17 92 30 18 48 56 90 82 43 39 1 93 27 69 93 33 31 92 35 26 42 8
11 67 82 83 87 93 50 79 68 74 84 29 35 60 5 11 92 24 50 16 60 1 59 9 23 67 0 25 56 40 48 55 7 49 91 60 82 36 19 70 79 25 19 9 1 31 24 7 25 1 83 6
9 82 89 37 99 35 79 54 52 99 50 32 45 93 71 49 90 4 77 5 19 50 28 28 22 68 85 74 84 51 84 37 15 89 81 74 59 50 46 60 27 33 21 61 18 46 50 63 61 4
69 38 4 76 12 91 93 9 20 97 62 28 0 59 42 40 89 94 42 20 86 46 64 86 71 43 66 74 45 88 49 27 54 4 81 46 57 85 14 50 32 73 43 12 32 83 39 68 89 83
92 61 69 75 39 96 81 71 68 13 66 30 6 9 90 59 84 46 59 3 6 91 93 96 83 50 57 35 12 67 69 65 3 73 91 16 49 15 18 43 88 97 11 91 74 63 25 7 70 21 2
```

다음의 Text.txt 파일을 통해 프로그램을 시행해보았다.

```
Microsoft Visual Studio 디버그 콘솔
603 980 37
방한의 개수는 11921 개입니다.
걸린 시간은 706.000000 ms 입니다.
C:\Users\POSTECH20180285\source\repos\data_assn\Debug\data_assn.exe (45588 프로세스)이(가) 0 코드로 인해 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요.
```

706ms (0.706초) 만에 프로그램이 올바르게 시행되었다.

```
Microsoft Visual Studio 디버그 콘솔
5 5 4
1 5 4 8 1
2 3 7 9 2
1 5 1 2 6
4 5 7 0 4
3 8 3 6 5
6
C:\Users\POSTECH20180285\source\repos\data_assn\Debug\data_assn.exe (38296 프로세스)이(가) 0 코드로 인해 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요.
```

Pdf 파일에 있는 예제도 올바르게 시행되었다. (답 6을 정확히 출력)

<토론>

Prob 1. 을 할 때 처음에 왼쪽에서 구한 나무막대의 개수를 구하고 오른쪽에서 구한 나무막대의 개수를 구했는데, 둘을 더하지 않고 왼쪽에서 구한 나무막대의 개수만 출력해서 값이 이상하게 나왔었다.

나중에는 그냥 둘 다 `sum++` 을 이용하게 한 뒤 `sum` 을 출력하게 프로그래밍했다.

Prob 2. LIS 를 구현할 때, `lower_bound` 를 구하기 위해 탐색방법을 코딩해야하나? 라는 생각을 했었다. `C++` 을 이전에 배운적이 없었기 때문에 `C++` 에 존재하는 헤더파일과 함수를 잘 몰랐기 때문이다.

하지만 구글에서 `lower_bound` 를 구하는 방법을 검색하던 도중 `<algorithm>` 헤더파일에 `lower_bound` 라는 함수가 존재한다는 것을 알고 이를 사용함으로써 코딩을 간단하게 할 수 있었다.

그리고, 아쉬웠던 점은 이 방법대로 LIS를 구하면 LIS의 길이는 알 수 있어도 LIS를 구성하는 원소는 알 수 없다는 것이었는데,

이는 또다른 배열 `P`를 만들고, `p[i]`에 수열의 `i`번째 원소가 `L` 배열 내에서 위치하는 인덱스를 저장하는 처리를 따로 함으로써 해결할 수 있다는 것을 알았다.

Prob 3. 인접한 빙하를 컴퓨터가 어떻게 인식하게 할 지 생각하는데 꽤 오랜 시간이 걸렸다.

작년에 `assn3` 에서 '애니팡'을 만드는 프로그래밍 숙제가 나왔었는데, 애니팡도 3개이상 인접한 배열을 컴퓨터가 인식하게 만들어야 한다는 점이 `prob3`와 비슷하다는 생각을 했다.

'큐'를 이용한 BFS 방법을 인터넷을 통해 알아낸 다음, `enqueue` 와 `dequeue` 를 직접 구현했다.

또한 `search` 함수를 통해 `for` 문을 돌리던 중 녹지않은 빙하가 있으면 그 빙하를 중심으로 상 -> 좌 -> 하 -> 우를 탐색해서 녹지않은 연결된 빙하가 있는지 탐색하도록 만들었다.

이 `search` 과정을 할 때 함수에다가 배열의 주솟값 뿐만 아니라 배열의 인덱스까지 넘겨줘야 한다는 점이 어려웠다.

왜냐하면, 배열의 $x > 0$ 인지, $y > 0$ 인지 등을 검사하기 위해서, 그리고 다음 search 함수에 `arr[x+1][y]` 를 넘겨주기 위해서는 배열의 인덱스를 주고받을 필요가 있었고,

Deque를 이용해 배열의 주솟값을 주고받기 위해서는 search 함수가 배열의 주솟값을 주고받을 수 있어야 했다.

처음에는 배열의 주솟값만 search 함수에 넘겨주었는데, 이렇게 하면 그 배열의 상 하 좌 우 를 인식하기 위해서 직접 포인터의 사칙연산을 해주어야 한다는 점이 매우 불편하다는 것을 깨닫고 코드를 전반적으로 수정하는 과정을 거쳤다.

<결론>

새롭게 깨달은 점보다는, 이전에 배웠던 내용들 중 까먹은 내용들을 환기하는 데 많은 도움이 되었던 과제라고 생각한다.

C 언어를 1학년 때 처음 배운 이후로 겨울방학동안 계절학기를 수강하느라 프로그래밍 공부를 할 시간이 따로 없었다.

이번 과제를 하면서 작년에 배웠던 프로그래밍과 문제해결 과목의 ppt 를 처음부터 끝까지 다시 보며 복습하고, 인터넷에서 C++ 과 관련된 여러 자료들을 찾아보며 과제를 완수할 수 있었다.

<개선방향>

C++ 에 포함되어 있는 여러 함수를 안다면, 코딩을 간단하게 해서 실행 속도를 높일 수 있을 것 같다.

<참고문헌>

<http://dblab.duksung.ac.kr/ds/pdf/Chap10.pdf> 제 10장 그래프

<https://hijuworld.tistory.com/1> 프로그램의 속도를 알기위해 clock 함수를 찾아보았다.

<https://gmlwjd9405.github.io/2018/08/15/algorithm-bfs.html> BFS에 대한 설명

<https://mygumi.tistory.com/102?category=677288> BFS에 대한 설명

<https://mygumi.tistory.com/191> BFS에 대한 설명

<https://seungkwan.tistory.com/8> LIS에 대한 설명

<https://blockdmask.tistory.com/168> lower_bound 함수에 대한 설명

작년 프로그래밍과 문제해결 LAB12 ppt 와 과제에 포함된 queue 구현 방법 참고하였음