

과제 1

사다리 타기

20232907 정현승

1. 문제 1

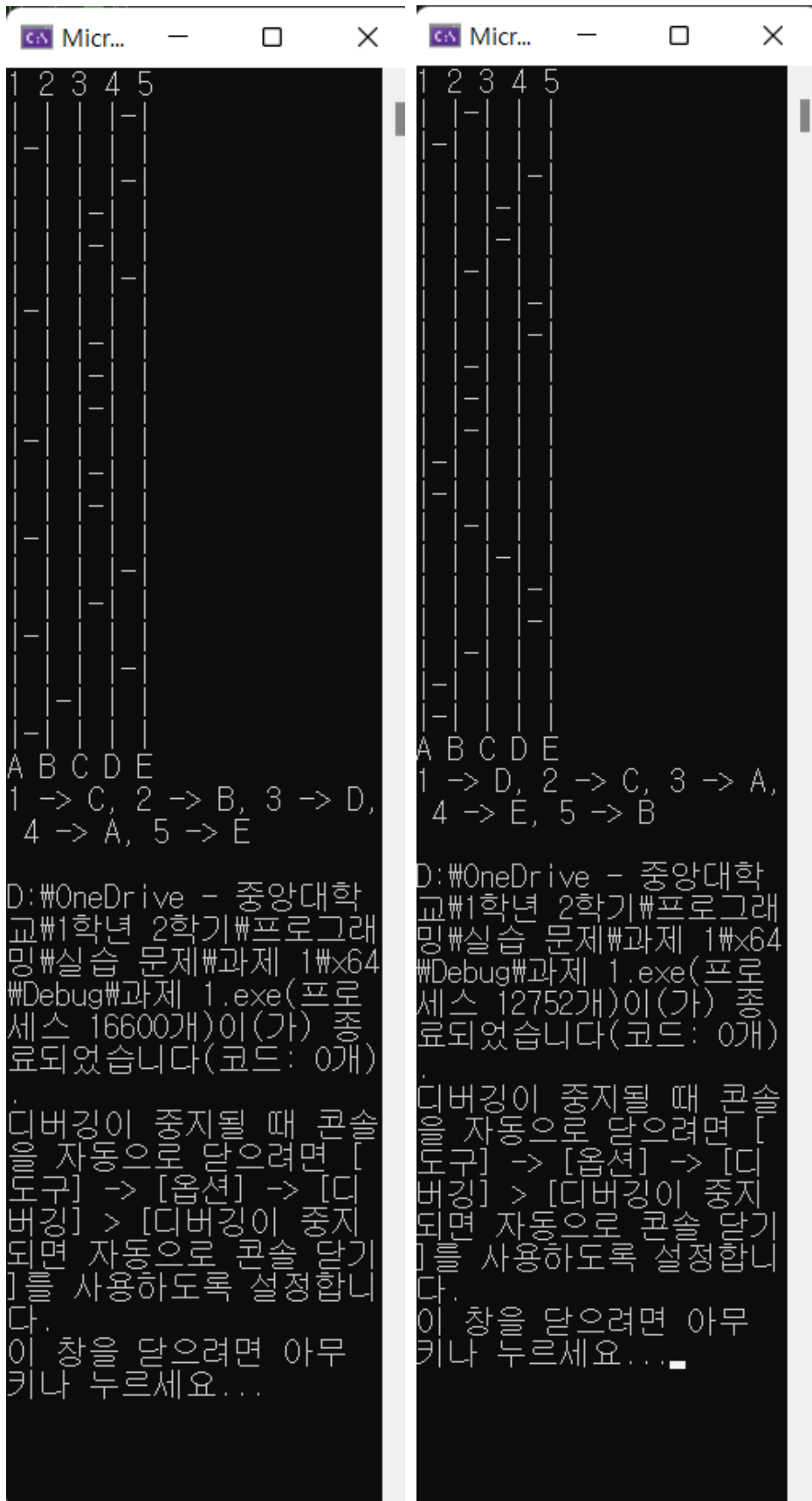
5명이 참여하는 사다리를 랜덤하게 고른다. 이때 가로선은 랜덤으로 매 판마다 바뀌어야 한다. 가로선은 15개 이상이어야 하며 한 곳에 10개 이상 너무 집중되거나 같은 줄에 2개의 가로선이 붙어 있을 수는 없다. 사다리 그림과 그 결과를 출력한다.

2. 사다리 가로선 문제 해결 방안

사다리의 세로 길이는 20줄이고, 참여자도 5명이고, 가로선은 최소 15개이고 같은 줄에 2개의 가로선이 붙을 수는 없다는 점을 해결하기 위해 간단하게 세로 한 칸 당 가로줄은 한 개만 그리는 것으로 해결한다. 이에 따라 가로줄이 같은 높이에 2개 이상이 놓이는 경우를 제거할 수 있으며(가로줄이 떨어져서 같은 높이에 그려지는 경우도 같이 제외된다.) 세로 한 칸 당 가로줄이 하나 그어지게 되면 가로선이 20개, 즉 세로 길이만큼 그어지게 되므로 가로선의 최소 개수도 채울 수 있다. 이러면 한 곳에 가로줄이 10개 이상 너무 집중되는 경우만 피하면 되는데, 이는 두 가지의 과정을 통해 구현했다. 먼저 가로줄의 총 개수가 20개라는 점을 이용해 두 인접한 세로선 사이의 가로선 개수 자체를 평균의 2배가 넘지 않도록 제한했다(가로선 총 개수가 20개인 경우, 가로선 위치를 뽑을 때 한 곳에 이미 9개가 있는데 이번에 그 곳을 뽑아 10번째가 되어 가로선의 평균의 2배가 되면, 가로선의 위치를 다시 뽑는다). 그 다음으로는 두 인접한 세로선 사이에 가로줄이 6개 이상 있으면서 그 사이에 다른 세로선과 연결된 가로선이 없는 상태에서 그 두 인접한 세로선 사이에 가로선이 뽑혀 7번 연속으로 다른 세로선과 연결되지 않고 가로선만 계속해서 나타나는 상태가 되면, 7번째 가로선을 왼쪽 또는 오른쪽으로 랜덤으로 옮겼다. 이때 가로선의 전체 개수와 연속되는 개수는 가로선을 뽑을 때마다 다시 계산하지 않고 가로선을 뽑을 때마다 하나씩 세서 저장했다. 이 두 과정을 통해서 가로선이 집중되는 경우를 피했다.

이를 적용하여 문제 1을 해결한 결과는 다음과 같다.

3. 문제 1의 결과



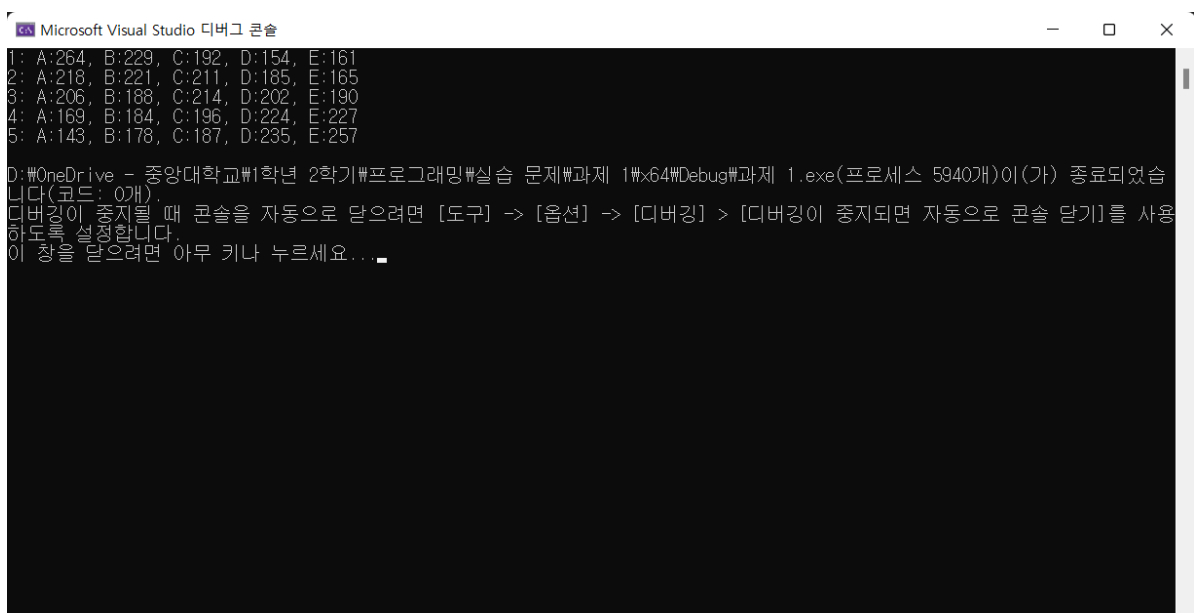
왼쪽부터

그림 1, 그림 2

4. 문제 2

우리가 만든 사다리가 공정한가 확인을 위해 이를 1000번 반복해 1, 2, 3, 4, 5번 각각 출발할 때 A, B, C, D, E에 각각 몇 번 도착하는지 통계를 내어 본다.

5. 문제 2의 결과



```
Microsoft Visual Studio 디버그 콘솔
1: A:264, B:229, C:192, D:154, E:161
2: A:218, B:221, C:211, D:185, E:165
3: A:206, B:188, C:214, D:202, E:190
4: A:169, B:184, C:196, D:224, E:227
5: A:143, B:178, C:187, D:235, E:257

D:\OneDrive - 중앙대학교\1학년 2학기\프로그래밍\실습 문제\과제 1\#x64\Debug\과제 1.exe(프로세스 5940개)이(가) 종료되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

그림 3

1번이 A에 도착할 확률이 다섯의 평균인 20%를 넘는 26.4%, 5번이 E에 도착할 확률이 25.7%를 기록하는 반면 1번에 D에 도착할 확률은 15.4%, 5번이 A에 도착할 확률은 14.3%로 우리가 만든 사다리는 그리 공정하지 않다는 점이 확인되었다. 특히 한 쪽 끝에서 출발하는 경우 반대쪽 끝에 도달할 확률이 매우 작고, 5개 줄 공통으로 시작했던 줄 그대로 끝날 확률이 가장 높다는 점도 확인할 수 있다.

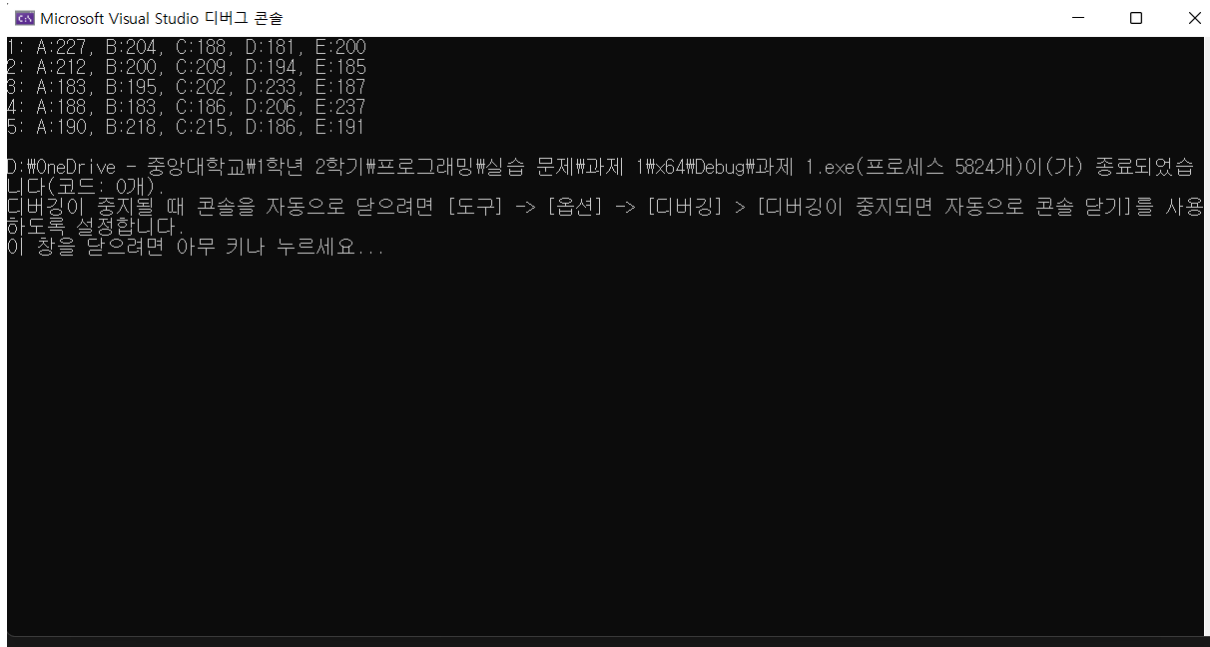
6. 문제 3

앞선 문제를 통해 사다리 게임이 공정하지 않다는 점을 확인하였다. 그렇다면 이 사다리 게임을 공정하도록 만들어 본다. 공정한 사다리게임을 구현하고 이를 문제 2처럼 1000번 반복해 그 결과를 출력한다.

7. 사다리 게임의 불공정성 해결 방안

위의 불공정성 문제는 1번이 A에 갈 때는 사다리를 타고 다른 줄에 갔다가 다시 같은 두 세로 줄 사이에 배치된 사다리를 타고 원위치로 오기만 하면 되어서 쉽지만, E에 갈 때는 각각 다른 세로 줄 사이에 있는 사다리를 총 4개 타야 하고 또 그 사이에 원위치로 되돌아가는 사다리를 타면 안 되므로 이런 일이 발생하는 것으로 추정된다. 이는 꼭 양 끝인 1번과 5번에서만 나타나는 것은 아니고 사다리에 참여하는 인원 수를 늘리면 사다리판 중간에서 출발하는 사람도 양 끝에 가기 어려울 것으로 예상되나 지금 사용하는 사다리는 딱 5명만 참여하므로 이번에는 이런 문제가 일어나지 않은 것으로 추정된다. 이를 해결하기 위해서는 줄 간의 이동이 더욱 활발해져야 할 것이다.

이 문제의 해결방법은 간단할 수 있다. 많은 사람이 참여하는 사다리가 아니라 딱 5명만 참여하는 사다리이므로 단순히 사다리의 세로 길이를 늘리는 것으로 해결할 수 있다. 줄의 이동이 활발해져야 하므로 가로줄을 늘리는 건데, 앞에서 세로 한 칸당 가로줄을 1개 그리도록 하였으므로 사다리의 세로 길이를 늘리면 해결되는 것이다. 앞에서 사다리를 프로그래밍할 때 사다리의 세로 길이에 숫자 20을 바로 넣은 게 아니라 코드 앞부분에 `#define Down_len 20`을 선언하고 세로 길이값에 `Down_len`을 넣어주었기 때문에 숫자 하나만을 바꾸는 것으로 사다리의 세로 길이를 늘릴 수 있다. 사다리 가로선도 한 줄에 사다리가 평균의 2배가 넘지 않도록 하는 방식으로 구현해 이와도 충돌이 발생하지 않는다.

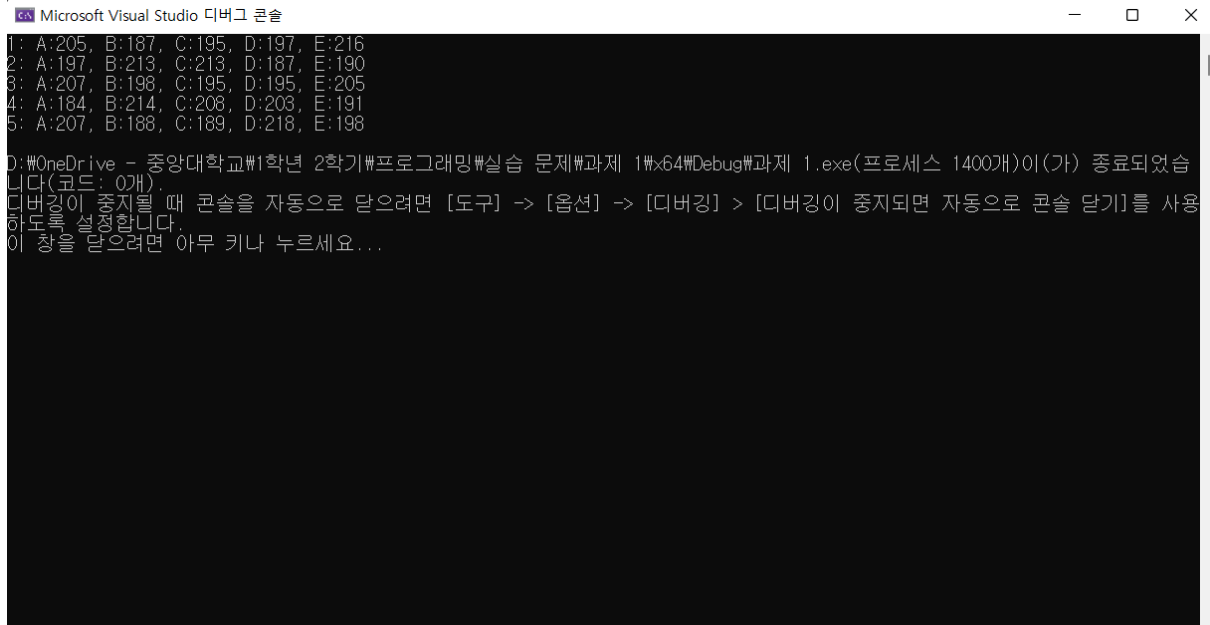


```
Microsoft Visual Studio 디버그 콘솔
1: A:227, B:204, C:188, D:181, E:200
2: A:212, B:200, C:209, D:194, E:185
3: A:183, B:195, C:202, D:233, E:187
4: A:188, B:183, C:186, D:206, E:237
5: A:190, B:218, C:215, D:186, E:191

D:\OneDrive - 중앙대학교\1학년 2학기\프로그래밍\실습 문제\과제 1\Debug\과제 1.exe(프로세스 5824개)이(가) 종료되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

그림 4

위는 `Down_len`을 40으로 변경하고 사다리를 1000번 반복한 결과이다. 평균이 20%인데 23.7%~ 18.1%의 범위를 보이니 나름 불공정성을 해결한 편이라고 할 수 있다. 다만 각각의 확률의 차이를 더욱 줄이기 위해 `Down_len` 값을 80으로 다시 변경하고 실행해보았다.



```
Microsoft Visual Studio 디버그 콘솔
1: A:205, B:187, C:195, D:197, E:216
2: A:197, B:213, C:213, D:187, E:190
3: A:207, B:198, C:195, D:195, E:205
4: A:184, B:214, C:208, D:203, E:191
5: A:207, B:188, C:189, D:218, E:198

D:\OneDrive - 중앙대학교\1학년 2학기\프로그래밍\실습 문제\과제 1\64\Debug\과제 1.exe(프로세스 1400개)이(가) 종료되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

그림 5

확률의 범위가 21.6%~18.4%로 더 줄었다. 이 정도의 차이는 독립시행으로 인한 결과물(오차)로 남길 수 있을 정도로 작은 수준으로 판단할 수 있다. 따라서 사다리 게임의 불공정성을 해결하는 방법은 사다리의 가로줄을 늘리는 것이라고 할 수 있다.

Down_len값이 80일 때 사다리 게임 1회의 결과는 다음과 같다.



왼쪽부터
그림 6, 그림 7

8. 소스코드

첨부파일 참조