프로그래밍 과제 1 - 사다리 타기

20222663 권기영

1. 사다리를 1000개 그리고, 각각의 확률 분석하기. - original_ladder.c

1) void create_ladder() - 사다리 만들기

총 4개의 세로줄에 대해서, 16개의 가로줄을 랜덤으로 배치한다. 이 때, [16][4] 크기의 배열을 사용하고, 한 행에 하나의 가로줄만 놓는다.

2) int move_ladder(int start) - 사다리 타고 내려가기

start번째 출발점에서 시작해서 16개의 행을 차례대로 내려가면서, 위치의 바로 옆에 가로줄이 있으면 가로줄 쪽으로 한 칸 이동한다. 모두 내려가고 난 뒤 도착점을 반환한다.

3) void count_result() - 도착지에 대한 횟수 세기

1000개의 다른 사다리를 만들고, 5개의 도착지에 대한 도착 횟수를 센다.

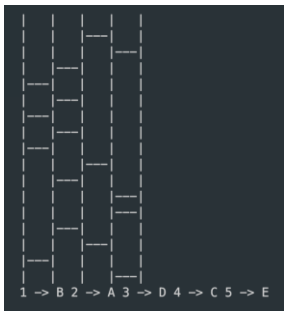
4) void print_result() - 도착지에 대한 결과 출력

1000개의 사다리에 대해서 도착지의 결과를 10번 반복해서 출력한다. 도착지 A, B, C, D, E의 도착 횟수와, 총 도착 횟수를 출력한다.

5) void print_ladder() - 사다리 출력

사다리를 출력하고, 각 출발지에 대해 move_ladder를 실행해 출발지와 도착지를 출력한다.

<실행 결과>



```
cd "/Users/minsiki2/Desktop/kiyoung/" && gcc original_ladder.c -o original_ladder && "/Users/minsiki2/Desktop/kiyoung/"original_ladder
➜  kiyoung cd "/Users/minsiki2/Desktop/kiyoung/" && gcc original_ladder.c -o original_ladder && "/Users/minsiki2/Desktop/kiyoung/"o
riginal_ladder
1 : A (292) B (257) C (185) D (135) E (131)  1000
2 : A (249) B (257) C (225) D (146) E (123)  1000
3 : A (247) B (256) C (212) D (148) E (137)  1000
4 : A (270) B (244) C (223) D (131) E (132)  1000
5 : A (261) B (216) C (191) D (193) E (139)  1000
6 : A (265) B (230) C (219) D (160) E (126)  1000
7 : A (287) B (216) C (207) D (152) E (138)  1000
8 : A (292) B (230) C (182) D (159) E (137)  1000
9 : A (284) B (225) C (198) D (159) E (134)  1000
10 : A (280) B (226) C (212) D (160) E (122)  1000
➜  kiyoung
```

실행에 대해서 분석한 결과, A에서 E로 갈 수록 확률이 낮아졌다. 1000번 시행을 10번 반복했는데도 불구하고 동일한 결과가 나오는 것을 보면, 분명히 경향성이 있고, 공정하지 않은 방법임을 확인할 수 있다.

2. 사다리가 공정한 결과가 나오지 않는 이유

사다리 타기가 공정하지 않은 이유는, 사다리의 세로선들이 직접적으로 연결되어 있지 않기 때문이다. 예를 들어, 1번 세로줄에서 2번 세로줄으로 가기 위해서는 하나의 가로줄만 타면 되지만, 5번 세로줄으로 가기 위해서는 1번 -> 2번 -> 3번 -> 4번 -> 5번, 총 4개의 가로줄을 타야 한다. 그런데 가로줄을 한 번 타는 경우에 대해서 일반적으로 2가지 경우(왼쪽, 오른쪽)가 있으므로, 오른쪽을 총 4번 선택해야 5번 세로줄로 갈 수 있다.

또한 이는 가로줄의 개수가 적을수록 불공정하다.

3. 공정한 사다리를 만드는 방법

1) 가로줄의 개수를 늘린다. - fair1_ladder.c

다른 모든 조건은 유지한 채, 가로줄의 개수를 16 -> 500개로 늘렸다.

<실행 결과>

```
➜  kiyoung cd "/Users/minsiki2/Desktop/kiyoung/" && gcc fair1_ladder.c -o fair1_ladder && "/Users/minsiki2/Desktop/kiyoung/"fair1_l
adder
1 : A (192) B (185) C (205) D (220) E (198)  1000
2 : A (208) B (196) C (220) D (183) E (193)  1000
3 : A (205) B (192) C (206) D (179) E (218)  1000
4 : A (204) B (200) C (202) D (202) E (192)  1000
5 : A (214) B (172) C (192) D (227) E (195)  1000
6 : A (204) B (196) C (205) D (210) E (185)  1000
7 : A (190) B (193) C (205) D (212) E (200)  1000
8 : A (219) B (175) C (200) D (199) E (207)  1000
9 : A (219) B (208) C (193) D (197) E (183)  1000
10 : A (198) B (212) C (203) D (173) E (214)  1000
➜  kiyoung
```

이전에 구현한 사다리 보다 공정성이 확연히 증가한 모습을 볼 수 있다.

2) 결과를 미리 정해 놓은 사다리를 만들어, 원하는 확률로 맞춘다. - fair2_ladder.c

(1) int create_1000_ladder() - 원하는 사다리 1000개를 만드는 함수

출발점 1 2 3 4 5에 대응하는 도착점이 ABCDE, BCDEA, CDEAB, DEABC, EABCD 인 사다리를 각각 200개 씩 총 1000개를 만든다.

create_ladder함수를 통해 16개의 가로선을 가지는 사다리를 만든 뒤, 도착점을 검사해서 조건에 맞는 도착점을 가지는 사다리만 선택하고, 조건에 맞지 않을 경우 새로 사다리를 만든다.

이 과정을 위의 5가지 도착점 조건에 대해서 200번씩 반복한다.

(2) int is_ladder(int n, int dest) - 사다리가 조건에 맞는지 검사하는 함수

create_1000_ladder 함수 내에서 사다리를 선택할 때 조건에 맞는지 검사하는 함수이다. 5개의 출발점에 대해서 move_ladder 함수를 사용하여 도착점을 구하고, 도착점이 조건과 하나라도 맞지 않으면 0을 반환하고, 모두 맞으면 1을 반환한다.

(3) count_result() - 1번째 출발점에서 출발한 사다리의 도착점의 횟수 세기

위에서 구현한 사다리 타기와 동일한 방식으로 도착점의 횟수를 세는데, 뽑은 사다리를 표시하는 visited 배열을 만들어 뽑지 않은 사다리만 뽑는다. 1000개의 사다리에 대해서 무작위로 하나를 뽑은 뒤, 뽑지 않은 사다리라면 사다리를 타고, 뽑은 사다리라면 다시 뽑기를 반복한다.

(4) 가로줄의 개수 선정

가로줄을 16개로 선정했는데, 그 이유는 가로줄이 짝수 개여야만 하기 때문이다.

출발점 1 2 3 4 5에 대응하는 도착점이 A B C D E인 조건이 있는데, 이는 출발점과 도착점이 동일한 경우이다. 출발점과 도착점이 동일하기 위해서는 갔다가 다시 돌아오는, 즉 짝수개의 쌍이 맞춰져야만 한다.

따라서 가로줄을 놓을 수 있는 4개의 영역에 대해서, 모두 짝수개의 가로줄이 있어야 한다. 따라서 짝수 + 짝수 + 짝수 + 짝수 = 짝수, 총 가로줄은 짝수가 되어야 한다.

<실행 결과>

모든 도착지점 전부 동일하게 200번이 나왔음을 알 수 있다. 애초에 표본 사다리 자체가 1000개이고, 각 도착지점 당 200개씩 할당되어 있기 때문에, 1000 개를 전부 뽑는다면 200개씩 나오게 된다.