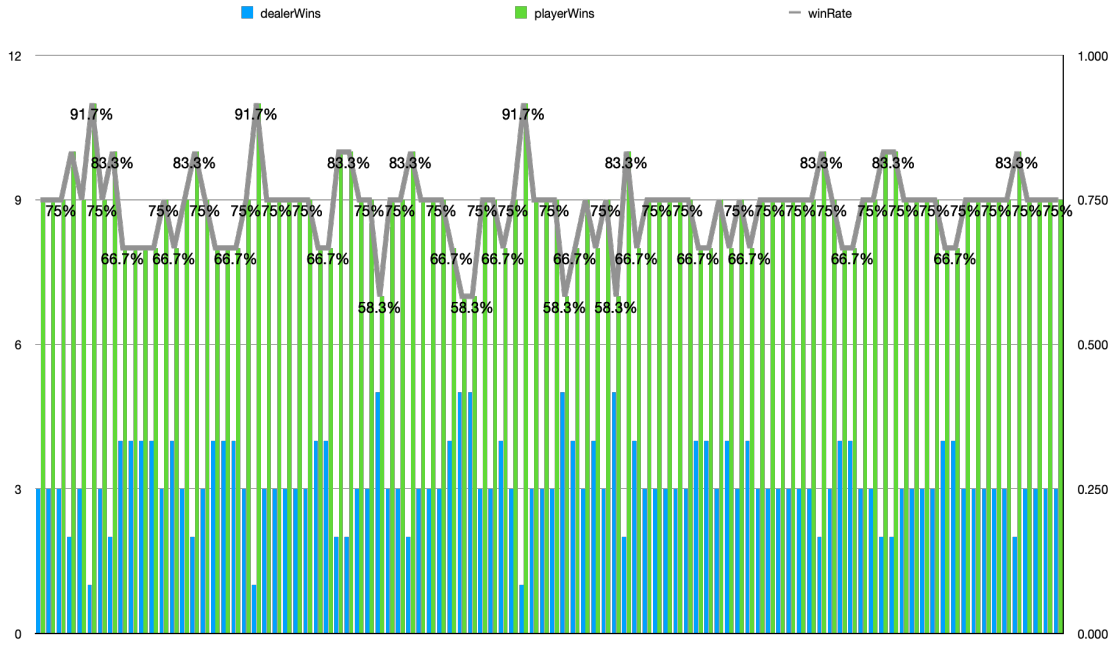
20201015 최유림

1) 나의 승률은 평균적으로 73%이다.



평균 승률을 계산하기 위해 게임을 100번 반복했고 이를 나타낸 그래프이다. 파란색 막대 그래프는 dealer가 이긴 횟수를 의미하고, 초록색 막대 그래프는 player아 이긴 횟수를 의미한다. 회색 꺾은선 그래프는 회차당 승률을 계산한 값을 표현한다. 승률은 딜러의 승리 횟수와 플레이어의 승리 횟수를 더한 값을 분모로 하고, 플레이어의 승리 횟수를 분자로 하는 확률을 사용한다. 해당 확률의 평균을 계산했을 때 73%의 확률로 승리하는 것을 확인 가능하다.

여기서 사용한 player의 전략은 deck을 hit하기long elapsedTimeInMillis = endTime - startTime; System.out.println("실행 시간(ms): " + elapsedTimeInMillis); 전에 미리 하나의 카드를 미리 보고, 21을 넘어서 bust되지 않는 수일 때만 hit하는 것이다. Bust하여 지는 횟수를 줄이기 위해 고안한 방식이다. Dealer의 전략은 stand-on-17을 따른다.

2) Chat GPT를 사용하여 친구의 프로그램을 대신 작성하였다.
   A. ChatGPT 제공 코드의 승률
      i. 평균 승률은 86%로 나의 프로그램보다 훨씬 높은 승률을 가진다.
      ii. 해당 승률은 1번 문항에서 구한 방식과 동일한 방식으로 계산하였다.
   B. ChatGPT 제공 코드와의 승률 차이 발생 이유
      i. 나의 전략은 카드 덱의 순서를 알고 있음을 이용하지만 이를 단편적으로 사용한다. 코드를 작성하여 딜러가 지는 경우를 살펴봤을 때, Bust가 발생하여 지는 경우가 많았다. 따라서 카드 덱의 순서를 미리 알면 다

음 카드의 value를 알 수 있기 때문에 이를 bust 방지에 사용하자는 전략을 사용한 것이다.

ii. ChatGPT 코드는 덱의 순서를 모두 고려하여 발생 가능한 모든 경우를 고려하여 점수를 계산한다. DP와 Memoization 방식을 사용하여 모든 경우를 탐색하고 메모에 작성해둠으로써 카드의 순서를 알고 있다는 점을 활용하고 있기 때문에 더 높은 승률을 보인다.

3) 해당 프로그램의 시간복잡도 계산

A. Subproblem 개수

i. O(n) : 남은 카드 개수 => 52개

B. Time per subproblem

i. dealerDP : O(n)

ii. playerDP: O(n)

iii. 총 O(n^2)

C. 총 시간 복잡도

i. O(n) * O(n^2) = O(n^3)

ii. 대략 52^3 = 140,608번의 연산 필요하다.

iii. 직접 걸린 시간을 확인해본 결과 5ms 소요된다.

iv. 예상한 시간보다 소요 시간 차이가 많았다.

1. 예상하는 이유로는 메모이제이션 도입을 통한 중복 연산 감소로 인한 것을 생각하고 있다.

4) 딜러의 전략보다 나은 전략

A. 나의 프로그램에서 사용한 Bust 감소 전략

지는 경우의 수를 살펴보았을 때, 상대방보다 점수가 낮은 경우와 Bust가 발생하는 경우가 있는데 상대방보다 점수가 낮은 경우를 해결하기 위해서는 딜러의 전략과 동일한 전략을 취하면 승리 확률을 높일 수 있다. 그리고 Bust 발생 횟수를 줄이게 되면 해당 횟수만큼 승리를 가져올 수 있는 확률이 늘어난다. 이 방법을 도입하기 위해서는 플레이어가 카드의 순서를 알고 있다는 전제 하에 가능하다. 다음 카드가 Bust를 발생하게 되면 hit하지 않고 넘어간다. 해당 전략을 취하면 73%의 승률을 갖게 된다. 이는 기존 방식보다 더 높은 승률을 가짐을 알 수 있기 때문에 더 나은 전략이라 할 수 있다.

B. ChatGPT가 제안한 모든 경우의 수 고려 방식

해당 방식은 DP를 극한까지 사용하여 모든 카드의 순서를 알고 있을 때 나올 수 있는 모든 경우의 수를 고려하는 것이다. 이는 더 좋은 승률을 확보하며 시간복잡도도 높지 않다. DP를 효과적으로 사용할 수 있는 방식이기에

딜러의 전략보다 훨씬 좋은 전략이라고 생각이 든다.