〈 GBC_Algorithm PA2 〉

27기 최 하영

1. FACT

- 가장 많은 양의 포도주를 마실 수 있도록 하는 프로그램

1) 포도주 시식 시, 두 가지 규칙 있음
   i) 포도주 잔 선택하면 잔에 들어있는 포도주 모두 마셔야 하며, 마신 후 원위치
   ii) 연속으로 놓여 있는 3잔을 모두 마실 수 없음 → 최대 2잔 연속 가능

2) 1부터 n까지 번호가 붙어 있는 n개의 포도주 잔, 잔에 들어 있는 포도주의 양

3) 입력 : (첫째 줄) 포도주 잔의 개수 n $(1 \leq n \leq 10{,}000)$
   (두번째 줄부터 n번째 줄까지) 포도주 잔에 들어 있는 포도주의 양 $(0 \leq n \leq 1{,}000)$

4) 출력 : 최대로 마실 수 있는 포도주의 양 출력

2. Overviews

1) Dynamic Programming을 사용
2) 현재 위치를 나타낼 배열 필요
3) 포도주의 최대량을 담을 배열 필요
4) 현재 마신 잔이 0번 연속인지, 1번 연속인지, 2번 연속인지 구분해야 함
5) 포도주의 최대값을 배열에 누적 저장하여 비교하며 max 값을 구한다

3. Algorithm

1) 포도주 잔의 개수$(1 \leq n \leq 10{,}000)$를 입력 받는다.
2) n개의 포도주 잔에 채워질 포도주 양을 n번 입력 받는다.
3) 현재 잔의 포도주 양을 arr[i], n번째까지 포도주를 최대로 마신 양을 dp[n].
4) 현재 잔이 1~3번일 경우와 4번 이상일 경우로 나눈다.
   i) 잔이 1-3번일 경우,
      가) 잔이 3번일 경우, 3번+2번, 3번 1번, 2번+1번 잔의 포도주 양을 더한 것 중 어느 것이 max 값인지 구한다.
      나) 잔이 1, 2번일 경우, 현재 잔+현재 위치 전까지의 최대 마신 양이 max!
   ii) 잔이 4번 이상일 경우
      → 현재 잔을 마시지 않은 case와 현재 잔을 마셔서 1번 연속된 case, 현재

잔까지 2번 연속된 case중에서 max 값을 구한다.

Ex) 현재 4번 잔일 때,

가) 4번 잔을 마시지 않음 → 3번 잔까지의 최대 포도주 양 dp[i-1]

나) 4번 잔을 마심 (1번 연속) → 2번 잔까지의 최대 포도주 양 dp[i-2]
+ 현재 마신 잔의 포도주 양 arr[i]

다) 4,3번 잔을 마심 (2번 연속) → 1번 잔까지의 최대 포도주 양 dp[i-3]
+ 3번 잔의 포도주 양 arr[i-1]
+ 현재 마신 잔의 포도주 양 arr[i]

5) max() : x, y, z의 값을 서로 비교하여 최대 값을 반환한다.

6) max 값을 구해 출력해준다.

4. Time complexity

O(n) : for loop 돈다.

```
for(int i = 1; i <= n; i++){
    //1번~3번 잔
   if(i < 4){
        if(i == 3){
            //3+2, 3+1, 2+1 중 max
            dp[i] = Max(arr[i-1]+arr[i], arr[i-2]+arr[i-1], arr[i-2]+arr[i]);
        }
        else
            //max인 경우 -> 현재+현재까지의 최대 포도주 양
            dp[i] = arr[i]+dp[i-1];
   }
   //4번 잔 이상부터
   else{
       //4번일 경우, 3번까지의 최대 양/1번 연속:2번까지의 최대양+4번/2번 연속:1번까지의 최대양+3번+4번
       //위의 케이스 중 max값 선택
       dp[i] = Max(dp[i-1], arr[i]+dp[i-2], arr[i]+arr[i-1]+dp[i-3]);
   }
}
```