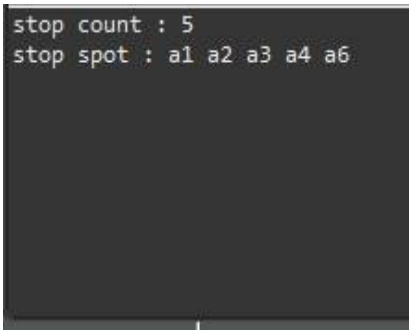
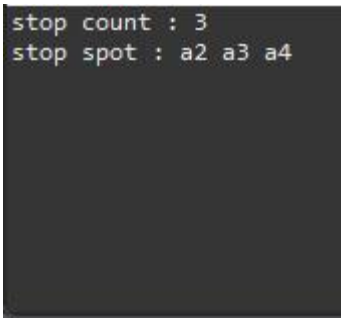
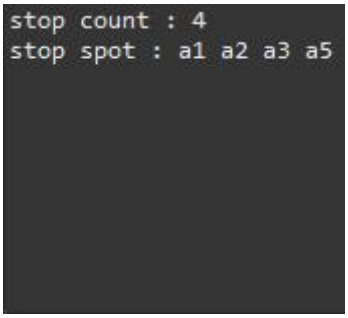
# Algorithm
# - greedy Algorithm -

| | |
|---|---|
| **제출일자** | 2020.10.13 |
| **과제번호** | 06 |
| **이　　름** | 강인한 |
| **학　　번** | 201701969 |

# 실행결과




optimal substructure :
if (최근 stop 장소부터 다음 가야 할 위치까지 거리의 합 > 최대 이동 가능 거리)
then 현재의 위치가 stop 장소가 됨
if (최근 stop 장소부터 다음 가야 할 위치까지 거리의 합 =< 최대 이동 가능 거리)
then 현재의 위치가 stop 장소가 되지 않고 다음 stop 위치까지 이동 후 판별

recursive equation :
현재 위치에서 가장 멀리 갈 수 있는 곳을 arr[i]라고 하면
arr[i]=
if (최근 stop 장소부터 다음 가야 할 위치까지 거리의 합 =< 최대 이동 가능 거리)
arr[i+1]
else arr[i]

greedy choice property : 현재 위치에서 갈 수 있는 최대 이동 거리까지 가게 된다면 최적해에 반드시 포함되게 된다.
d1을 출발지 d6을 도착지라고 하면 d1에서 최선의 선택이 최적해에 포함될 수밖에 없다. 만약 d1에서 출발해 d2에 stop spot이 된다고 하면 이것 대신 다른 것이 들어온다고 해도 이것보다 더 많이 갈 수 있는 방법은 없다. 즉 d1에서 d2까지 한번에 가는 것은 이 거리에서 최적해라고 볼 수 있다. 점화식을 이용해 진행해 나간다면 최적해는 d1에서 출발해 d2에서 stop spot이 되는 방법을 포함할 수밖에 없다.

풀이과정 : optimal substructure에서 말했듯이 최근 stop장소부터 이번에 가야할 다음 위치까지의 총합이 최대 거리보다 크다면 현재 위치에 멈춰 물을 충전하고 그렇지 않을 경우는 다음 거리까지 이동 가능하므로 이동하여 그때 정하게 된다.

코드 설명 : 출발점부터 도착지까지 while문을 통해 반복하고 내부의 while 문을 통해 현재 위치에서 최대 이동 거리까지 이동한다. 각각의 이동 거리 들을 sum에 합치고 최대 이동 거리와의 비교를 통해 멈출지 멈추지 않을지를 결정한다.

느낀 점 : 개인적으로 greedy algorithm과 이번 과제의 연관성을 이해하는 것이 조금 어려웠던 것 같습니다. 코드작성에서는 어려운 부분이 없었던 것 같습니다.