백준 12865, 평범한 배낭

(냅색 문제)

DP[N][K]: N은 가방에 들어가는 물건 수, K는 가방에 들어갈 수 있는 무게

	1	2	3	4	5	6	7
6							
4							
3							
5							

Input

W: 가방 무게, V: 가방 가치

	0	1	2	3
W	6	4	3	5
V	13	8	6	12

Step 1

먼저 무게 6인 물건을 넣었을 때 가치를 채운다

	1	2	3	4	5	6	7
6	0	0	0	0	0	13	13
4							
3							
5							

Step 2

그 다음 무게가 4인 물건의 가치를 넣는다,

이때 dp[무게4][K-4] 값을 더한 후 dp[이전 무게][전체]와 비교하여 큰 수를 대입한다

	1	2	3	4	5	6	7
6	0	0	0	0	0	13	13
4	0	0	0	8	8	13	13
3							
5							

Step 3

그 다음 무게가 3인 물건의 가치를 넣는다,

이때 dp[무게3][K-3] 값을 더한 후 dp[이전 무게][전체]와 비교하여 큰 수를 대입한다

	1	2	3	4	5	6	7
6	0	0	0	0	0	13	13
4	0	0	0	8	8	13	13
3	0	0	6	8	8	13	14
5							

Step 4

마지막 무게가 5인 물건의 가치를 넣는다,

이때 dp[무게5][K-5] 값을 더한 후 dp[이전 무게][전체]와 비교하여 큰 수를 대입한다

	1	2	3	4	5	6	7
6	0	0	0	0	0	13	13
4	0	0	0	8	8	13	13
3	0	0	6	8	8	13	14
5	0	0	6	8	12	13	14

백준 12865, 평범한 배낭

(문제 이해 부족,
물건을 여러개 넣을 수 있는 것으로 이해)

Input W : 가방 무게, V : 가방 가치

	0	1	2	3
W	6	4	3	5
V	13	8	6	12

dp[k] : 가방에 무게 k 만큼 넣었을 때 최대가치

1	2	3	4	5	6	7
0	0	0	0	0	0	0

Step 1

k=1 : 가방의 모든 무게가 1보다 크므로 0을 대입
k=2 : 가방의 모든 무게가 2보다 크므로 0을 대입
k=3 : 모든 가방중 무게가 3인 가방만 넣을 수 있고, 가치는 6
이제 넣을 수 있는 가방 무게는 k-3=0,
무게가 0 이하인 가방이 없으므로 최종 가치 6을 대입

1	2	3	4	5	6	7
0	0	6	0	0	0	0

Step 2

k=4,
case1 : 무게가 3인 가방을 넣을 때 가치6 과
k-3=1 에서 dp[1]=0 이므로 최종 최대 가치는 6
case2 : 무게가 4인 가방을 넣을 때 가치는 8 이고
k-4=0 이므로 최종 최대 가치는 8

⇒ case 1 보다 case 2 경우가 가치가 크므로 8을 대입

1	2	3	4	5	6	7
0	0	6	8	0	0	0

Step 3

k=5,
case1 : 무게가 3인 가방을 넣을 때 가치6 과
k-3=2 에서 dp[2]=0 이므로 최종 최대 가치는 6
case2 : 무게가 4인 가방을 넣을 때 가치는 8 이고
k-4=1 에서 dp[1]=0 이므로 최종 최대 가치는 8
case3 : 무게가 5인 가방을 넣을 때 가치는 12 이고
k-5=0 이므로 최종 최대 가치는 12

⇒ case 3 경우가 가치가 제일 크므로 12 대입

1	2	3	4	5	6	7
0	0	6	8	12	0	0

Step 4

k=6,
case1 : 무게가 3인 가방을 넣을 때 가치6 과
k-3=3 에서 dp[3]=6 이므로 최종 최대 가치는 12
case2 : 무게가 4인 가방을 넣을 때 가치는 8 이고
k-4=2 에서 dp[2]=0 이므로 최종 최대 가치는 8
case3 : 무게가 5인 가방을 넣을 때 가치는 12 이고
k-5=1 에서 dp[1]=0 이므로 최종 최대 가치는 12
case4 : 무게가 6인 가방을 넣을 때 가치는 13 이고
k-6=0 이므로 최종 최대 가치는 13

⇒ case4 경우가 가치가 제일 크므로 13 대입

1	2	3	4	5	6	7
0	0	6	8	12	13	0

Step 5

k=7,
case1 : 무게가 3인 가방을 넣을 때 가치6 과
k-3=4 에서 dp[4]=8 이므로 최종 최대 가치는 14
case2 : 무게가 4인 가방을 넣을 때 가치는 8 이고
k-4=3 에서 dp[3]=6 이므로 최종 최대 가치는 14
case3 : 무게가 5인 가방을 넣을 때 가치는 12 이고
k-5=2 에서 dp[2]=0 이므로 최종 최대 가치는 12
case4 : 무게가 6인 가방을 넣을 때 가치는 13 이고
k-6=1 에서 dp[1]=0 이므로 최종 최대 가치는 13

⇒ case1,2 경우가 가치가 제일 크므로 14 대입

1	2	3	4	5	6	7
0	0	6	8	12	13	14

백준 2655, 가장 높은 탑 쌓기

step 1

입력 (S: 밑면 넓이, H: 높이, W: 무게, pos: 정렬 전 위치)

	0	1	2	3	4
S	25	4	9	16	1
H	3	4	2	2	5
W	4	6	3	5	2
pos	0	1	2	3	4

→ 밑면의 넓이를 내림차순으로 정렬

정렬 후 (앞으로 벽돌을 쌓을 때, 무게만 따지면 됨)

	0	1	2	3	4
S	25	16	9	4	1
H	3	2	2	4	5
W	4	5	3	6	2
pos	0	3	2	1	4

dp

maxH: 해당 벽돌을 제일 위에 두었을 때, 탑의 최대 높이

prev : 해당 벽돌 바로 밑에 있는 벽돌의 index

pos : 해당 벽돌의 원래 위치

	0	1	2	3	4
maxH	0	0	0	0	0
prev	-1	-1	-1	-1	-1
pos	0	0	0	0	0

step 2

	0	1	2	3	4
S	25	16	9	4	1
H	$\boxed{3}$	2	2	4	5
W	4	5	3	6	2
pos	$\boxed{0}$	3	2	1	4

	0	1	2	3	4
maxH	3	0	0	0	0
prev	-1	-1	-1	-1	-1
pos	0	0	0	0	0

0번 벽돌은 밑에 아무 것도 둘 수 없으므로 maxH에 3, pos에 0, prev에 -1대입

step 3

	0	1	2	3	4
S	25	16	9	4	1
H	3	$\boxed{2}$	2	4	5
W	4	5	3	6	2
pos	0	$\boxed{3}$	2	1	4

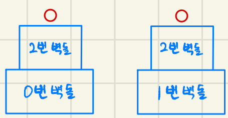
	0	1	2	3	4
maxH	3	2	0	0	0
prev	-1	-1	-1	-1	-1
pos	0	3	0	0	0

1번 벽돌은 0번보다 무게가 4가므로 0번을 밑에 둘수 없어 H를 바로 대입, 원래 위치 3을 pos에 대입

step 4

	0	1	2	3	4
S	25	16	9	4	1
H	3	2	$\boxed{2}$	4	5
W	4	5	3	6	2
pos	0	3	$\boxed{2}$	1	4

	0	1	2	3	4
maxH	$\boxed{3}$	2	5	0	0
prev	-1	-1	0	-1	-1
pos	0	3	2	0	0



2번 벽돌은 0과 1 벽돌 위에 올 수 있고

dp[0].maxH > dp[1].maxH 이므로

dp[2]. maxH에 $H_{(2)}$ + dp[0].maxH을 대입

pos에는 원래 위치 2를 대입

prev에는 바로 밑의 벽돌의 index 0을 대입

step 5 나머지 4행 결과

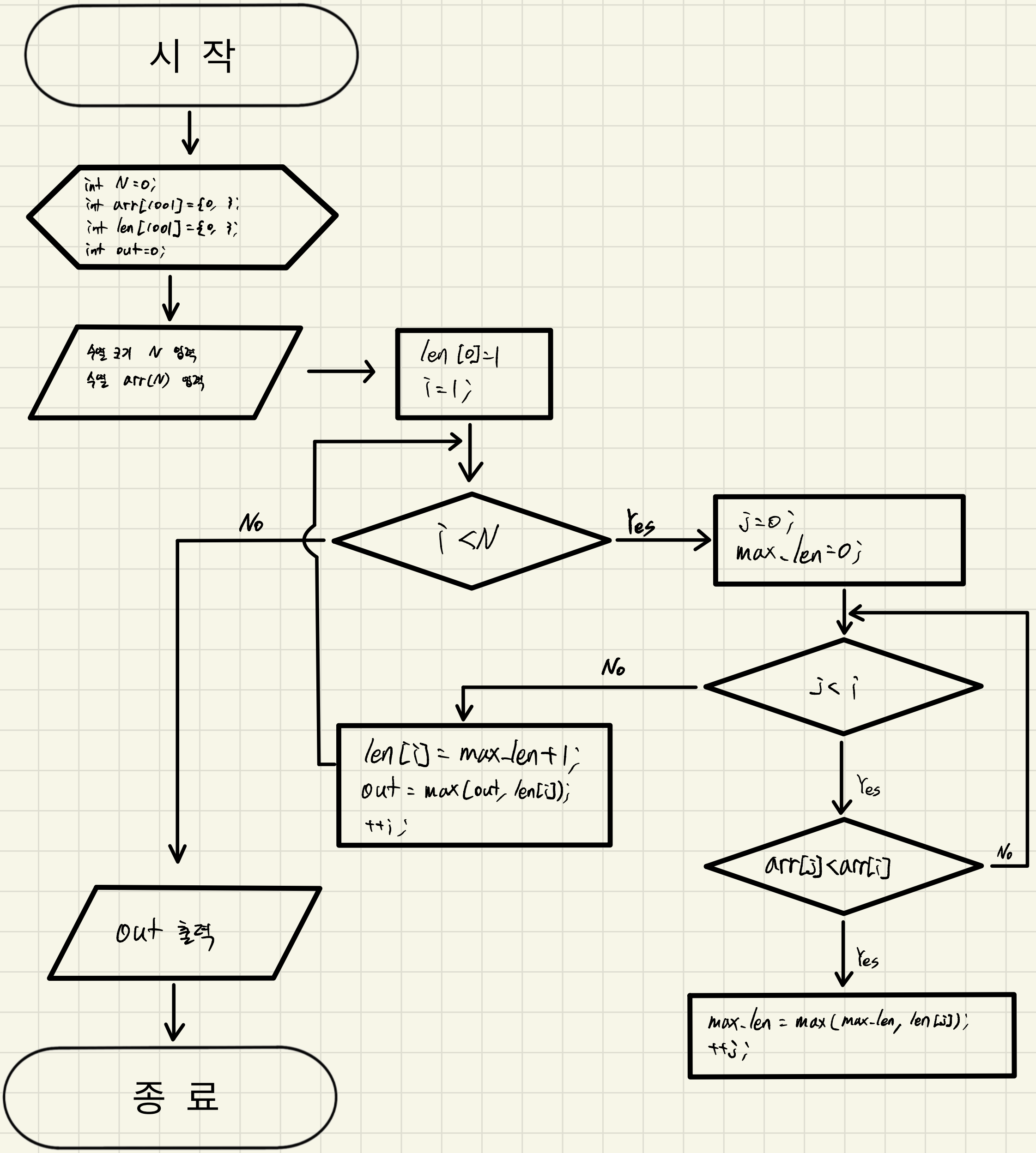
	0	1	2	3	4
maxH	3	2	5	4	10
prev	-1	-1	0	-1	-1
pos	0	3	2	1	4

step 6 출력

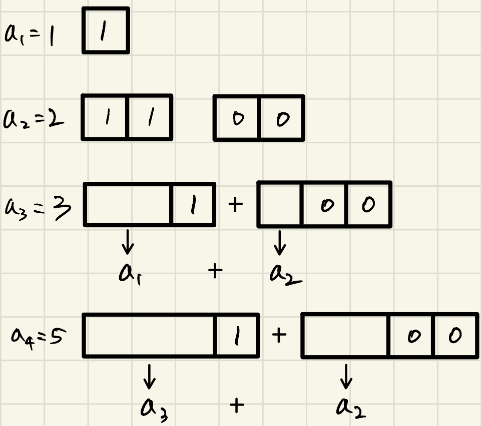
pseudo code

```
while (true)
{
    printf("%d", cur_dp.pos);
    if(cur_dp.prev==-1)
    {
        break;
    }
    cur_dp = dp[cur_dp.prev];
}
```

백준 11053, 가장 긴 증가하는 수열



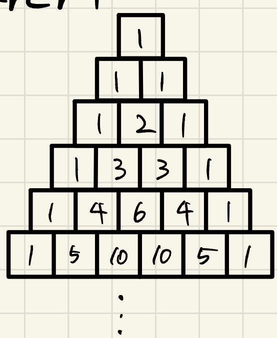
백준 1904, 01타일



점화식: $a_n = a_{n-1} + a_{n-2}$

백준 16395, 파스칼의 삼각형

조합공식 : $_nC_r = {}_{n-r}C_r + {}_{n-r}C_{r-1}$



pseudo code

```
arr[0][0]=1;
arr[1][0]=1;
arr[1][1]=1;
for (int i=2; i≤N; ++i)
{
    for(int j=0; j≤N; ++j)
    {
        if (j=0 || j=N)
            arr[i][j]=1;
        arr[i][j] = arr[i-1][j]+arr[i-1][j-1];
    }
}
```