



#12851. 숨바꼭질 2

<https://www.acmicpc.net/problem/12851>

25.06.09



Problem

#12851. 숨바꼭질2

- ~ 22:25 | 문제 풀기
- ~ 22:30 | 힌트 공개
- ~ 22:55 | 2차 풀기
- ~ 23:00 | 풀이공개

12851번 제출 맞힌 사람 슯코딩 재채점 결과 채점 현황 내 제출 난이도 기여 강의를 질문 게시판

숨바꼭질 2

성공

4 골드 IV



시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
2 초	512 MB	71449	20583	14256	26.176%

문제

수빈이는 동생과 숨바꼭질을 하고 있다. 수빈이는 현재 점 $N(0 \leq N \leq 100,000)$ 에 있고, 동생은 점 $K(0 \leq K \leq 100,000)$ 에 있다. 수빈이는 걷거나 순간이동을 할 수 있다. 만약, 수빈이의 위치가 X 일 때 걷는다면 1초 후에 $X-1$ 또는 $X+1$ 로 이동하게 된다. 순간이동을 하는 경우에는 1초 후에 $2 \times X$ 의 위치로 이동하게 된다.

수빈이와 동생의 위치가 주어졌을 때, 수빈이가 동생을 찾을 수 있는 가장 빠른 시간이 몇 초 후인지 그리고, 가장 빠른 시간으로 찾는 방법이 몇 가지인지 구하는 프로그램을 작성하시오.

입력

첫 번째 줄에 수빈이가 있는 위치 N 과 동생이 있는 위치 K 가 주어진다. N 과 K 는 정수이다.

출력

첫째 줄에 수빈이가 동생을 찾는 가장 빠른 시간을 출력한다.

둘째 줄에는 가장 빠른 시간으로 수빈이가 동생을 찾는 방법의 수를 출력한다.

예제 입력 1

5 17

예제 출력 1

4
2



Hint

Step 1. 문제 분석



문제 요약

수빈이는 현재 위치 N 에 있고, 동생은 K 에 있다.

수빈이는 다음 3가지 방법으로 이동할 수 있다.

- $X - 1$ (1초 소요)
- $X + 1$ (1초 소요)
- $X * 2$ (1초 소요)

목표는 수빈이가 동생을 찾는 가장 빠른 시간과, 그 시간 안에 동생을 찾는 방법의 수를 구하는 것이다.



Hint

Step 1. 문제 분석



문제 유형

- 최단 시간 → **BFS**로 탐색
- "경로 수"까지 구해야 하므로, **같은 노드에 여러 경로로 도달할 수 있음**을 고려해야 한다.
- 중복 방문 방지 + 경로 수 누적 = **dist**, **ways** 와 같은 배열 사용



Hint

Step 2. 접근 방식



풀이 아이디어

- `dist[x]`: x 지점까지 도달하는 데 걸리는 최단 시간 (0 ~ 100000)
- `ways[x]`: x 지점에 최단 시간으로 도달하는 방법의 수

BFS 로직

1. 시작점 `N`을 큐에 넣고 시작한다.
2. 이동 가능한 세 가지 위치(`X-1`, `X+1`, `2*X`)에 대해 탐색한다.
3. 처음 방문하는 위치는 `시간 저장 + 경로 수 복사`
4. 이미 방문한 위치라도, `같은 최단 시간이라면` `경로 수 누적`



Hint

Step 2. 접근 방식



풀이 아이디어

1. 아직 방문하지 않은 위치라면 (`dist[nx] == -1`)

- 지금 방문하는 게 **최단 거리**임!
- 그래서:
 - `dist[nx] = dist + 1`: 현재까지 걸린 시간 + 1초
 - `ways[nx] = ways[x]`: 이 시간에 처음 도달한 거니까, **이전 위치에서 오는 방법 수만큼 그대로 가져옴**
 - `queue.append(nx)`: 큐에 추가해서 이후 탐색

👉 최초 도달 + 경로 1세트만 존재

2. 이미 방문했지만, 같은 시간에 다시 도달한 경우 (`dist[nx] == dist + 1`)

- "또 다른 경로"로 같은 시간에 도착한 것
→ 이 경우에도 카운트해줘야 함
그래서 `ways[nx] += ways[x]`: 기존 방법 수 + 새로운 경로 수

👉 최단 시간은 유지되지만, 경로는 추가적으로 누적



Hint

Step 3. 주의할 점



주의할 점: 메모리 초과 & 중복 상태 관리

- 같은 위치를 여러 번 방문할 수 있다.
 - 단, 같은 시간에 도달했을 경우에만 추가 탐색이 유효함.
 - 방문 배열 없이 큐에 (위치, 시간)을 계속 추가하면
 $5 \rightarrow 10 \rightarrow 5 \rightarrow 10 \rightarrow \dots$ 처럼 무한 반복 가능 → 메모리 초과 발생
- 탐색 시 반드시 방문 시간(dist / visited) 체크 필요



해결 포인트

- $\text{dist}[x] / \text{visited}[x]$ 는 → x 위치에 처음 도달한 시간(최단 시간)을 저장
- $\text{ways}[x]$ 는 → x 위치에 도달할 수 있는 방법 수를 누적 저장
- 탐색 도중 이미 방문한 위치라도
 - 현재 시간 + 1 == dist[next] 인 경우엔
 - $\text{ways}[\text{next}] += \text{ways}[\text{current}]$ (다른 경로로의 동일 시간 도달 허용)
- 이동 범위는 $0 \leq x \leq 100000$ 범위 내로 제한

Solution

1. dist, ways 2개의 배열로 관리

1. `MAX = 100000`으로 문제에서 주어진 최대 범위를 설정한다.
2. `N, K`를 입력받아 수빈이의 위치와 동생의 위치를 저장한다.
3. `dist`: 각 위치까지 도달하는 **최소 시간**을 저장 (초기값 -1)
4. `ways`: 각 위치까지 **최소 시간으로 도달하는 방법의 수**를 저장 (초기값 0)
5. `queue`를 생성하고 시작점 `N`을 넣는다.
 - `dist[N] = 0`: 시작점은 0초에 도달
 - `ways[N] = 1`: 시작 위치에는 한 가지 방법으로 도달
6. **BFS 탐색 시작** (큐가 빌 때까지 반복):
 - `x = queue.popleft()`: 현재 위치를 꺼낸다.
 - 다음 이동 가능한 위치들을 계산: `x - 1, x + 1, x * 2`
 - 범위 조건 검사: $0 \leq nx \leq 100000$ 인 경우만 유효
 - 처음 방문하는 위치(`nx`): `dist[nx] == -1`
`dist[nx] = dist[x] + 1`: 현재 위치에서 한 번 더 이동한 시간
`ways[nx] = ways[x]`: 이전 위치까지 도달하는 모든 방법 수를 그대로 가져옴
`queue.append(nx)`로 큐에 추가
 - 이미 방문한 위치지만 동일 시간에 도달 가능한 경우:
`dist[nx] == dist[x] + 1`이면,
`ways[nx] += ways[x]`: 추가로 또 다른 경로가 생긴 것이므로 누적
7. BFS 종료 후:
 - `print(dist[K])`: 동생 위치에 도달하는 **최소 시간**
 - `print(ways[K])`: 해당 시간으로 동생에게 도달할 수 있는 **방법 수**

```
import sys
from collections import deque

input = sys.stdin.readline

MAX = 100000
N, K = map(int, input().split())
dist = [-1] * (MAX + 1)
ways = [0] * (MAX + 1)

queue = deque([N])
dist[N] = 0
ways[N] = 1

while queue:
    x = queue.popleft()
    for nx in (x - 1, x + 1, x * 2):
        # 범위 내
        if 0 <= nx <= MAX:
            # 아직 방문하지 않은 위치
            if dist[nx] == -1:
                dist[nx] = dist[x] + 1 # 현재 걸린 시간 +1 초
                ways[nx] = ways[x] # 이전 위치에서 오는 방법의 수
                queue.append(nx)
            # 이미 방문했지만, 같은 시간에 다시 도달한 경우
            elif dist[nx] == dist[x] + 1:
                ways[nx] += ways[x] # 기존 방법 수 + 새로운 경로 수

    # 가져오기

print(dist[K])
print(ways[K])
```


✨ Solution



다른 풀이

2. 배열 하나에 한 번에 관리하기

```
MAX = 100_000
from collections import deque

N, K = map(int, input().split())
# info[i] = [최단 시간, 경로 수]
info = [[-1, 0] for _ in range(MAX+1)]
info[N] = [0, 1]

q = deque([N])
while q:
    x = q.popleft()
    d, w = info[x]
    for nx in (x-1, x+1, x*2):
        if 0 <= nx <= MAX:
            # 첫 방문
            if info[nx][0] == -1:
                info[nx][0] = d + 1
                info[nx][1] = w
                q.append(nx)
            # 같은 최단 시간에 다시 방문
            elif info[nx][0] == d + 1:
                info[nx][1] += w

print(info[K][0])
print(info[K][1])
```

✧ Solution



다른 풀이

3. visited 배열 + “레벨별 카운팅” 트릭 (ways 배열 없이)

```
import sys
from collections import deque
input = sys.stdin.readline

N, K = map(int, input().split())
if N == K:
    print(0)
    print(1)
    sys.exit()

MAX = 100_000
visited = [False] * (MAX + 1)
visited[N] = True
```

```
q = deque([N])
time = 0
ans = 0
found = False

while q and not found:
    time += 1
    level_size = len(q)
    # 이번 레벨에서 새로 방문할 노드들 기록 (방문 표시는 레벨 끝에)
    next_level = []

    for _ in range(level_size):
        x = q.popleft()
        for nx in (x - 1, x + 1, x * 2):
            if 0 <= nx <= MAX:
                if nx == K:
                    # 동생 위치를 발견할 때마다 카운트
                    ans += 1
                    found = True
                # 아직 한 번도 다음 레벨에서 enqueue되지 않은 노드만
                elif not visited[nx]:
                    q.append(nx)
                    next_level.append(nx)

    # 이번 레벨의 탐색이 끝난 뒤, 새로 enqueue된 노드들에 한꺼번에 방문
    # 표시
    for node in next_level:
        visited[node] = True

# 답 출력
print(time)
print(ans)
```

Solution



다른 풀이

- [\[백준\] 12851. 숨바꼭질 2 - Python](#)
- [\[백준\] 12851. 숨바꼭질 2 \(python\)](#)

```
import sys
from collections import deque

N, K = map(int, sys.stdin.readline().split())
queue = deque()
queue.append(N)
way = [0] * 100001 # 최대 크기
cnt, result = 0, 0
while queue:
    a = queue.popleft()
    temp = way[a]
    if a == K: # 둘이 만났을 때
        result = temp # 결과
        cnt += 1 # 방문 횟수 +1
        continue

    for i in [a - 1, a + 1, a * 2]:
        if 0 <= i < 100001 and (way[i] == 0 or way[i] == way[a] + 1):
            # 범위 안에있고 방문하지 않았거나, 다음 방문이 이전 방문+1이면
            way[i] = way[a] + 1
            queue.append(i)

print(result)
print(cnt)
```

```
from sys import stdin

input = stdin.readline
from collections import deque
```

```
# 수빈 위치, 동생 위치
N, K = map(int, input().split())
```

```
MAX_SIZE = 100001
```

```
que = deque()
que.append(N)
visited = [-1] * MAX_SIZE
visited[N] = 0
cnt = 0
```

```
while que:
    # 현 위치
    current = que.popleft()
    # 도착
    if current == K:
        cnt += 1
    # 이동
    for next in [current * 2, current + 1, current - 1]:
        # 범위 내
        if 0 <= next < MAX_SIZE:
            # 첫방문 혹은 방문 시간이 같은 경우가 이미 있음(가장 빠른 시간 방법의 수를 위해)
            if visited[next] == -1 or visited[next] >= visited[current] + 1:
                visited[next] = visited[current] + 1
                que.append(next)
```

```
print(visited[K])
print(cnt)
```



Assignment

백준 #1697. 숨바꼭질

- 같은 원리 한 번 더 복습할 수 있음

1697번

제출

맞힌 사람

소요임

재제정 결과

채점 현황

내 제출

강의

질문 게시판

숨바꼭질

다국어

☆ 한국어

실버 I

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
2 초	128 MB	290376	87575	55791	26.577%

문제

수빈이는 동생과 숨바꼭질을 하고 있다. 수빈이는 현재 점 $N(0 \leq N \leq 100,000)$ 에 있고, 동생은 점 $K(0 \leq K \leq 100,000)$ 에 있다. 수빈이는 걸거나 순간이동을 할 수 있다. 만약, 수빈이의 위치가 X 일 때 걷는다면 1초 후에 $X-1$ 또는 $X+1$ 로 이동하게 된다. 순간이동을 하는 경우에는 1초 후에 $2 \times X$ 의 위치로 이동하게 된다.

수빈이와 동생의 위치가 주어졌을 때, 수빈이가 동생을 찾을 수 있는 가장 빠른 시간이 몇 초 후인지 구하는 프로그램을 작성하시오.

입력

첫 번째 줄에 수빈이가 있는 위치 N 과 동생이 있는 위치 K 가 주어진다. N 과 K 는 정수이다.

출력

수빈이가 동생을 찾는 가장 빠른 시간을 출력한다.

예제 입력 1 복사

예제 출력 1 복사

5 17

4