BOJ

#14226. 이모티콘

https://www.acmicpc.net/problem/14226

25.06.16

📝 Problem

풀이시간: 1시간 10분
힌트: 10시 30분부터

14226번 | 제출 | 맞힌 사람 | 숏코딩 | 재채점 결과 | 채점 현황 | 내 제출 | 난이도 기여 | 강의 | 질문 게시판

이모티콘 성공

☆

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
2 초	512 MB	31337	11895	7974	34.542%

문제

영선이는 매우 기쁘기 때문에, 효빈이에게 스마일 이모티콘을 S개 보내려고 한다.

영선이는 이미 화면에 이모티콘 1개를 입력했다. 이제, 다음과 같은 3가지 연산만 사용해서 이모티콘을 S개 만들어 보려고 한다.

1. 화면에 있는 이모티콘을 모두 복사해서 클립보드에 저장한다.
2. 클립보드에 있는 모든 이모티콘을 화면에 붙여넣기 한다.
3. 화면에 있는 이모티콘 중 하나를 삭제한다.

모든 연산은 1초가 걸린다. 또, 클립보드에 이모티콘을 복사하면 이전에 클립보드에 있던 내용은 덮어쓰기가 된다. 클립보드가 비어있는 상태에는 붙여넣기를 할 수 없으며, 일부만 클립보드에 복사할 수는 없다. 또한, 클립보드에 있는 이모티콘 중 일부를 삭제할 수 없다. 화면에 이모티콘을 붙여넣기 하면, 클립보드에 있는 이모티콘의 개수가 화면에 추가된다.

영선이가 S개의 이모티콘을 화면에 만드는데 걸리는 시간의 최솟값을 구하는 프로그램을 작성하시오.

Hint

Step 1. 문제 분석

문제 요약

영선이는 효빈이에게 S개의 이모티콘을 보내려 한다.

영선이는 1초동안 다음 세 개의 작업을 할 수 있다.

- 화면에 있는 이모티콘을 모두 복사해서 클립보드에 저장한다.
- 클립보드에 있는 모든 이모티콘을 화면에 붙여넣기 한다.
- 화면에 있는 이모티콘 중 하나를 삭제한다.

목표는 **영선이가 S개의 이모티콘을 보내는데 걸리는 최소시간**을 구하는 것이다.

💡 Hint

Step 1. 문제 분석

💡 문제 유형

- 이모티콘을 보내는데 걸리는 시간의 최솟값 구하기 & 모든 연산이 1초 → **BFS**로 탐색 가능
  a. 연산의 종류마다 걸리는 시간이 상이했다면, 다익스트라같은 더욱 복잡한 방법으로 해결할 수 있었을 것
- 화면에 있는 이모티콘의 수와 클립보드에 있는 이모티콘의 수를 동시에 고려해야 함
- 2<=S<=1000의 범위 처리도 고려

Hint

Step 2. 접근 방식

💡 풀이 아이디어

- BFS탐색을 하면서 조건에 맞게 3개의 연산을 진행하고 현재 개수가 S와 같아진다면 BFS탐색을 종료하고 시간 출력.
- 계속 같은 구간을 반복하지 않도록 visited 배열 사용.
- 동일한 화면 이모티콘 수라도, 클립보드 내용이 다르면 **다른 경로**이므로 화면 이모티콘 개수와 클립보드 이모티콘 개수를 동시에 고려하여 visited배열을 아래와 같이 작성.
  visited[화면이모티콘 개수][클립보드 이모티콘 개수]

BFS 로직

1. (화면 이모티콘 개수, 클립보드 이모티콘 개수, 시간) 형식의 초기값 큐에 저장.
2. 세 연산에 대해 탐색한다.
   a. 화면의 이모티콘 복사 후 클립보드에 저장. 클립보드 이모티콘 개수 = 화면 이모티콘 개수
   b. 클립보드의 이모티콘 화면에 복사. 화면 이모티콘 개수 = 화면 이모티콘 개수 + 클립보드 이모티콘 개수
   c. 화면에서 이모티콘 하나 삭제. 화면 이모티콘 개수 -= 1
3. 방문처리 조건 탐색
   a. 새로 계산된 화면 이모티콘 개수와 클립보드 이모티콘 개수가 범위 밖 or 이미 방문한 곳이면 continue
   b. 방문처리

✨Solution 다른 사람 풀이

```python
from collections import deque

S = int(input())

queue = deque([[1, 0, 0]])  # 화면의 이모티콘 개수, 클립보드 이모티콘 개수, 연산 횟수

visited = [[False] * 1001 for _ in range(1001)]
visited[1][0] = True

while queue:

    screen, clipboard, cnt = queue.popleft()

    if screen == S:  # 만약 스크린의 개수와 S가 동일하다면
        print(cnt)  # 걸린 횟수를 출력 후
        break  # 탈출

    for i in range(3):  # 연산을 3번 수행한다.

        # 화면에 있는 이모티콘을 복사해서 클립보드에 저장
        if i == 0:
            new_clipboard, new_screen = screen, screen

        # 화면에 클립보드에 있는 이모티콘 들을 추가
        elif i == 1:
            new_screen, new_clipboard = screen + clipboard, clipboard

        # 화면에 있는 이모티콘 개수 한개 빼기
        else:
            new_screen, new_clipboard = screen - 1, clipboard

        # 만약 새로 계산된 이모티콘과 클립보드의 개수가 범위를 벗어나거나 이미 방문한 적이 있다면 continue
        if new_screen >= 1001 or new_screen < 0 or new_clipboard >= 1001 or new_clipboard < 0 or visited[new_screen][
            new_clipboard]:
```

✨Solution 내풀이

```python
from collections import deque
import sys

S = int(sys.stdin.readline().strip())

# visited[screen][clipboard]
visited = [[-1] * (S + 1) for _ in range(S + 1)]

queue = deque()
queue.append((1, 0))  # 화면: 1, 클립보드: 0
visited[1][0] = 0

while queue:
    screen, clipboard = queue.popleft()

    # 목표 이모티콘 수에 도달하면 종료
    if screen == S:
        print(visited[screen][clipboard])
        break

    # 1. 복사 (화면 -> 클립보드)
    if visited[screen][screen] == -1:
        visited[screen][screen] = visited[screen][clipboard] + 1
        queue.append((screen, screen))

    # 2. 붙여넣기 (클립보드 -> 화면)
    if clipboard != 0 and screen + clipboard <= S and visited[screen + clipboard][clipboard] == -1:
        visited[screen + clipboard][clipboard] = visited[screen][clipboard] + 1
        queue.append((screen + clipboard, clipboard))

    # 3. 삭제 (화면 -1)
    if screen - 1 >= 0 and visited[screen - 1][clipboard] == -1:
```

💻 Assignment

백준 #13549. 숨바꼭질 3

- 같은 원리 한 번 더 복습할 수 있음