TP3

Sokoban 3: *Ahora es personal*

El objetivo de este TP es agregarle dos funcionalidades a nuestro viejo y querido Sokoban:

- **Deshacer**
- **Pistas**

Deshacer

Presionando una tecla determinada (configurable mediante `teclas.txt`) se deshace el último movimiento realizado. Al igual que la funcionalidad "deshacer" de los programas de uso común, debe ser posible deshacer reiteradas veces hasta llegar al estado inicial del nivel.

Pistas

Presionando una tecla determinada (configurable mediante `teclas.txt`), el efecto dependerá de dos casos posibles:

- No hay pistas disponibles: en este caso el juego intentará encontrar una solución al nivel, utilizando el algoritmo de **backtracking** que se describe a continuación. Si se encuentra la solución, la sucesión de acciones correspondiente será considerada como las pistas disponibles.
- Hay pistas disponibles: Se desencola una pista y se efectúa esa acción, como si la hubiera hecho el jugador.

Si el jugador efectúa cualquier otra acción que no sea la acción de "pista", se debe descartar las pistas disponibles, ya que no podemos asegurar que sean válidas para el estado actual.

Backtracking

Fuente: Wikipedia

Backtracking es un método para encontrar todas las soluciones (o algunas) a problemas computacionales, que consiste en plantear soluciones candidatas e ir descartando (*backtrack*) las que se determina que no son soluciones.

Para resolver un nivel de Sokoban utilizando backtracking, lo más fácil es plantear el algoritmo en forma recursiva:

```
algoritmo buscar_solucion(estado_inicial):
    visitados := un conjunto vacío de estados
    devolver backtrack(estado_inicial, visitados)

algoritmo backtrack(estado, visitados):
    agregar(visitados, estado)                              [1]
    si juego_ganado(estado):
        # ¡encontramos la solución!
        devolver Verdadero, []
    Para toda acción posible `a` desde el estado:
        nuevo_estado := mover(estado, a)
        si pertenece(visitados, nuevo_estado):              [2]
            continuar
        solución_encontrada, acciones := backtrack(nuevo_estado, visitados)
        si solución_encontrada:
            devolver Verdadero, concatenar([a], acciones)
    devolver Falso, ∅
```

Notar que `visitados` debe permitir almacenar un conjunto de estados `[1]`, y buscar en forma eficiente si un estado particular está o no en el conjunto `[2]`. Para que el algoritmo de backtracking sea eficiente, es un requerimiento que `pertenece` sea más rápido que una búsqueda lineal.

Una forma de lograr eso es que `visitados` sea un diccionario (`dict`) o un conjunto (`set`) de Python. Para ello es necesario que el estado sea inmutable, o bien que haya una forma de obtener una representación única e inmutable a partir de un estado. Supongamos que tenemos una función `h` que recibe un estado y devuelve una representación inmutable del mismo, entonces `[1]` y `[2]` pueden ser cambiados a:

```
agregar(visitados, h(estado))                           [1]
si pertenece(visitados, h(nuevo_estado)):               [2]
```

Algunas alternativas para la representación inmutable son:

- Una cadena con la representación del nivel en el formato de `niveles.txt`.
- Una tupla con la posición del jugador y todas las posiciones de las cajas.

Nota: aun logrando que `pertenece` sea eficiente, el algoritmo de backtracking podrá resolver únicamente los niveles más simples en un tiempo razonable. Por ejemplo, los primeros 4 niveles del set de niveles proporcionado en el TP2 pueden ser resueltos en alrededor de 1 minuto por backtracking. El nivel 5, que a simple vista parece simple, es computacionalmente más complejo ya que permite al jugador más libertad de movimiento. El algoritmo de backtracking puede no terminar en un tiempo razonable; eso es normal y esperable.

Además, si se implementa el algoritmo de backtracking en forma recursiva, para algunos niveles es probable que se llegue a superar el límite de recursión del intérprete Python. No es un requisito plantear una solución para este problema.

Entrega

La entrega del trabajo (a través del formulario de entregas) debe incluir todos los archivos necesarios para ejecutar el juego comprimidos en un archivo `zip`.