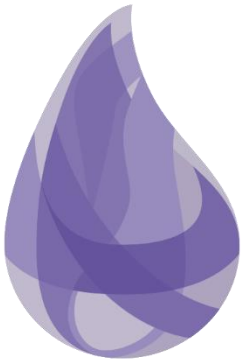
ELIXIR



DEKLARATÍV PROGRAMOZÁS I. NHF

KEMPING

Jankó András

NHVU6N

2023.10.07

1. Követelmények elemzése

A házi feladatban a híres Tents Puzzle problémát kellett megoldani, amit pontosabban ismertetett a házi feladat leírása a hivatalos tárgyoldalon. A feladat leírásában definiált bemenetre kell megmondani az összes lehetséges megoldást az adott rejtvényre. Az általam megvalósított megoldásban felsoroljuk az összes lehetséges kitöltést, majd kiszűrjük közülük a helyteleneket, és visszatérünk az összes helyes megoldással, DFS-t alkalmazva. A szűrés közben megpróbáljuk eldobni azokat a variációkat, amelyek biztosan nem vezetnek majd megoldásra, ezzel csökkentve a program számítási igényét. Először megnézi a program, ha egy sátor lerakása, a pályán belül esik, majd az oszlop, és sorokon jelzett számokat veszi figyelembe. Ha ezek a kényszerek nem teljesülnek akkor a DFS adott fa ágán nem halad tovább. Az előbbiek után következnek a szomszédos sátrak kényszerei, és végezetül pedig megnézzük, hogy egy helyre ne helyezzünk több objektumot.

2. Tervezés, az egyes eljárások, függvények specifikálása

```
@spec satrak(pd::puzzle_desc) :: tss::[tent_dirs]
```

tss: a pd feladványleíróval megadott feladvány összes

megoldásának listája, tetszőleges sorrendben.

```
def satrak({tents_count_rows, tents_count_cols, trees})
```

```
@spec dfs(map, trees, [field], [integer], [integer], [integer],
[integer], integer) :: [tent_dirs]
```

DFS-t alkalmazva megnézzük minden fára, hogy érvényes-e az adott fára a megoldás.

Ha érvényes továbbmegyünk egy fával, ha nem akkor nem foglalkozunk többet a megoldással.

Ha már nincs több fa visszatérünk egy üres megoldással:

```
defp dfs(_, _, _, _, _, _, _, 0),
defp dfs(all_tents, trees, previous_tents, target_tents_count_rows,
target_tents_count_cols, current_target_count_rows,
current_target_count_cols, i)
```

```
@spec validate_tent([integer], [integer], [integer], [integer],
[field], integer, field) :: list
```

Az alábbi függvényekkel megnézzük, hogy nem helyezünk egymás mellé-e sátrakat.

target_tents_count_rows: Elvárt sátrak száma soronként[], target_tents_count_cols: Elvárt sátrak száma oszloponként[]

new_target_count_rows: Jelenlegi sátrak száma soronként[], new_target_count_cols: Jelenlegi sátrak száma oszloponként[]

previous_tents: Eddig lerakott sátrak[{}], x0/y0: Jelenlegi sátor koordinátái

Ha maga a sátor invalid:

```
defp validate_tent(_, _, _, _, _, _, :invalid),
```

Ha maga a sátor invalid, és az utolsó:

```
defp validate_tent(_, _, _, _, _, 1, {:invalid, _}),
```

Ha bármilyen mennyiségű sátorban valamelyik invalid:

```
defp validate_tent(_, _, _, _, _, _, {:invalid, _}),
```

Utolsó sátor esete:

```
defp validate_tent(target_tents_count_rows, target_tents_count_cols,
current_target_count_rows, current_target_count_cols, previous_tents,
1, {x0, y0})
```

```
@spec check_conditions([integer], [integer], [integer], [integer],
[{integer, integer}], integer, integer, boolean)
  :: {:invalid | [integer], [integer]}
```

Ez a függvény nézi meg jó helyre raktuk-e a sátrat. (Nem romlott-e el a megoldás)

target_tents_count_rows: Elvárt sátrak száma soronként[], target_tents_count_cols: Elvárt sátrak száma oszloponként[]

new_target_count_rows: Jelenlegi sátrak száma soronként[], new_target_count_cols: Jelenlegi sátrak száma oszloponként[]

previous_tents: Eddig lerakott sátrak[{}], x/y: Jelenlegi sátor koordinátái

is_last_tent: Utolsó sátrat jelzi

```
defp check_conditions(target_tents_count_rows,
target_tents_count_cols, new_target_count_rows, new_target_count_cols,
previous_tents, x, y, is_last_tent)
```

```
@spec are_neighbours(field, field) :: boolean
```

Megvizsgáljuk 2 mező szomszédos vagy egybeesik-e.

Igazzal térünk vissza ha igaz az előző, és hamissal ha nem.

```
defp are_neighbours({x1, y1}, {x2, y2})
```

```
@spec gen_tent_coordinates(map, field, integer, integer, trees) :: map
```

Bejárjuk mind a 4 égtáj felé a kempinget, és megnézzük az adott fához merre rakható sátor.

prev: eddig megtalált helyek, tree: vizsgált fa, n/m: pálya dimenziói, all_trees: összes fa listája

A függvény végén kiegészítjük az eddig megtalált helyek Map-jét.

```
defp gen_tent_coordinates(prev, tree, n, m, all_trees) do
```

```
@spec gen_tent_coord_for_direction(field, atom, integer, integer,
trees) :: field | :invalid
```

Legeneráljuk a sátor koordinátáját, egy irány alapján (invalid-ot dobunk ha kiesne a pályáról).

{x,y}: fa koordinátái, d: sátor iránya, n/m: pálya dimenziói, all_trees: összes fa listája

```
defp gen_tent_coord_for_direction({x, y}, direction, n, m, all_trees)
```

3. Tesztelési megfontolások

A tárgyoldalon lévő tesztesetek használtam, amiket beolvastam az 1. KHF-ban lévő függvényemmel, amik mindegyike rendben lefutott, majd az ETS-be feltöltve is 10/10 helyes megoldással tért vissza a program.

4. Kipróbálási tapasztalatok

A programot lokálisan futtattam, és VS Code-ot használtam a fejlesztéshez.