Atol implementacija u Python 3 - Faza 1

Marko Petrović - 19317
Aleksandar Ristić - 19340
Nina Paunović - 19515

Link za projekat: https://github.com/ikugo-dev/vi-projekat
U okviru prve faze smo implementirali sve funkcije koje su neophodne za predstavljanje stanja table, unos početnih parametara igre, i unos poteza.
Nismo koristili eksterne biblioteke za implementaciju.

Sekvenca dogadjaja

Glavna (“__main__”) funkcija:

1. Traži se jedna od validnih veličina tabli od strane korisnika
2. Kreira se graf na osnovu zadate veličine
3. Pita se korisnik da li je protivnik drugo ljudsko biće ili kompjuter
4. Pita se korisnik da li želi da igra prvi
5. Ulazi se u glavni game loop (za sada je kraj nakon 999 poteza zbog toga što ne postoji provera da li je igra dostigla kraj)
6. Traži se unos za potez od strane korisnika (za sada se ne gleda da li je protivnik kompjuter ili ne zbog toga što nisu implementirane funkcije za obradu poteza od strane kompjutera)

Kreiranje table

Sama *create_starting_graph(size: int) -> dict[Point, Node]* funkcija predstavlja korake za kreiranje grafa.

1. *build_grid(size: int)* funkcija - generiše 2D matricu koja sadrži sva validna polja za igru.
2. *generate_points(size: int, grid: dict[str, list[int]])* funkcija - uzima tu 2D matricu i pretvaraju u listu tačaka koji su zapisani kao i na prezentaciji to jest jedno slovo jedan broj a ne dva broja kao u 2D listi
3. *build_graph(points: list[Point])* - Za svaku tačku u Listi tačaka inicijalizuje čvor koji treba tu da se nalazi i spaja ih sa svim susednim čvorovima

Prikazivanje table

Na sličan način kao što smo i kreirali tablu, mi inkrementalno pravimo 2D matricu od grafa koja će biti prikazana na ekranu.
Pozivam na funkciju *print_graph(graph: dict[Point, Node], graph_size: int)* mi definišemo neke globalne vrednosti koje zavise od veličine grafa (offset, visina grafa, broj kolona…) da bismo ih kasnije referencirali u ostalim funkcijama.

1. *create_table(graph: dict[Point, Node])* funkcija kreira praznu 2D matricu i zapisuje simbole iz našeg grafa u adekvatne pozicije
2. *create_island_table(table: list[list[str]])* funkcija uzima tabelu koju smo napravili iz *create_table* funkcije i dodaje joj ostrva koje se nalaze sa strane
3. *create_legend_table(island_table: list[list[str]])* funkcija dodaje simbole na tabelu sa ostrvima kako bi korisnik lakše mogao da uradi svoj potez

Nakon ovih poziva, *print_graph* funkcija može da odštampa konačnu 2D matricu, tj. tablu, na ekranu.

Struktura podataka:

Za predstavljanje table koristili smo strukturu podataka dictionary (ili hashmap u drugim jezicima) koji za ključ drži tačku (Point), a kao vrednost drži cvor (Node).
Point sadrži dva polja, slovo i broj, i obeležava poziciju kao što je zapisano i na prezentaciji.
Node ili čvor je tipična struktura podataka koje se koristi za implementaciju grafa.
Sadrži dva polja simbol (koji se koristi za grafičko predstavljanje tog čvora) i neighbors (koji predstavlja pokazivače na susedne čvorove).
Simbol je implementiran kao Enum koji sadrži unicode vrednosti emodžija.

Ova struktura podataka nam dozvoljava da:
a) Lakše pratimo validnost unosa korisnika (unos -> čvor)
b) Lakše implementiramo algoritme za pretraživanje koje radimo na ovom predmetu unutar čvorova