UCZELNIA BADAWCZA
INICJATYWA DOSKONAŁOŚCI

Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie

AGH University of Krakow

Projekt silnika szachowego z wykorzystaniem metod sztucznej inteligencji - 2

Bartłomiej Kózka
Promotor: Skoczylas Norbert, prof. dr hab. inż.

Plan prezentacji

Cel pracy

- **Głównym celem pracy** jest zaprojektowanie i implementacja silnika szachowego zdolnego do samodzielnej gry na poziomie **1000 ELO**, z wykorzystaniem klasycznych technik przeszukiwania drzewa gry (drzewa decyzyjnego).
- **Celem dodatkowym** jest porównanie skuteczności różnych funkcji oceny oraz metod optymalizacji procesu wyszukiwania ruchów. ✗

Metodyka

1. **Analiza teoretyczna i przegląd rozwiązań** (zapoznanie z istniejącymi silnikami) ✓
2. **Projekt architektury silnika** ✓
3. **Konfiguracja środowiska i pipeline CI/CD** ✓
4. **Implementacja i optymalizacja algorytmu przeszukiwania** ◆
5. **Funkcja oceny pozycji** (opracowanie heurystyki oceny) ✓
6. **Weryfikacja i ocena działania** (testy poprawności, porównanie wyników z innymi silnikami) ◆
7. **Wnioski i dalsze kierunki rozwoju** ◆

Metodyka – Reprezentacja planszy oraz Generacja ruchów ✓

Wybrana metoda reprezentacji planszy:

- **Piece-centric (bitboards)** – plansza traktowana jako skończony zbiór 64 pól, najczęściej reprezentowany przy użyciu 64-bitowych liczb całkowitych (każde pole odpowiada jednemu bitowi).

Wybrana metoda Generacji ruchów:

- **Iteracyjna (generatorowa)** – leniwe" generowanie ruchów i ich bieżąca ocena.

Metodyka – Reprezentacja planszy oraz Generacja ruchów ✓

Ocena poprawności i szybkości generacji ruchów – **testy perft**

rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1

Depth	Nodes	Captures	E.p.
0	1	0	0
1	20	0	0
2	400	0	0
3	8,902	34	0
4	197,281	1576	0
5	4,865,609	82,719	258

Wyniki – Reprezentacja planszy oraz Generacja ruchów ✓

Przykład wykonania perft testów

- **Głębokość**: 6
- **Liczba odwiedzonych node'ów**: 120 mln.
- **Czas wykonania**: 15 sekund

```
b1a3: 4856835
b1c3: 5708064
g1f3: 5723523
g1h3: 4877234
a2a3: 4463267
a2a4: 5363555
b2b3: 5310358
b2b4: 5293555
c2c3: 5417640
c2c4: 5866666
d2d3: 8073082
d2d4: 8879566
e2e3: 9726018
e2e4: 9771632
f2f3: 4404141
f2f4: 4890429
g2g3: 5346260
g2g4: 5239875
h2h3: 4463070
h2h4: 5385554
=== Perft Stats ===
  Nodes              : 119060324
  Captures           : 2896337
  Promotions         : 0
  Castles            : 0
  En-passant         : 5506
  Checks             : 836931
  Discovery checks   : 44
  Double checks      : 46
  Checkmates         : 355
  ==================
```

Metodyka – Algorytm przeszukiwania oraz Funkcja Oceny

Wybrana metoda przeszukiwania ruchów:

- **NegaMax framework**, Alpha–beta pruning

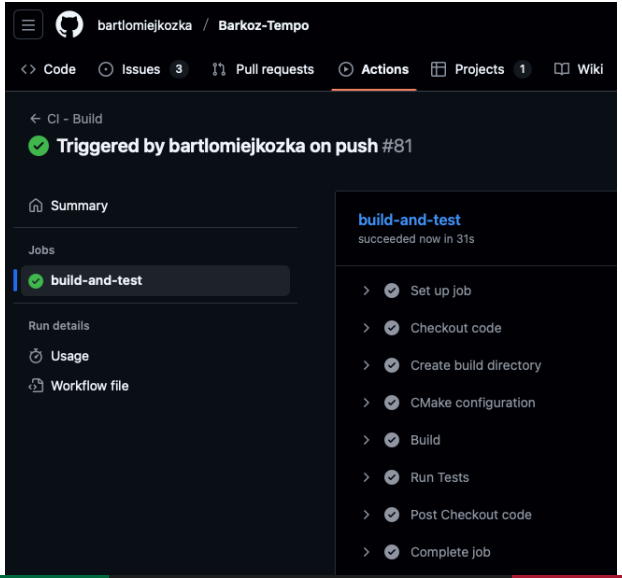
Wybrana funkcja oceny: (Hand-Crafted Evaluation)

```
f(p) = 200(K - K')
     + 9(Q - Q')
     + 5(R - R')
     + 3(B - B' + N - N')
     + 1(P - P')
     - 0.5(D - D' + S - S' + I - I')
     + 0.1(M - M') + ...

KQRBNP = liczba króli, hetmanów, żwie, ńgoców, skoczków i pionków
D, S, I = zdwojone, zablokowane i izolowane pionki
M = mobility (liczba legalnych ruchów)
```

Wyniki - CI/CD ✓

CI: builds and run tests



Pozostałe

- **Finalizacja implementacji alg. przeszukiwania**
- **Ocena wyników**
  - Analiza statycznej funkcji oceny - porównanie oceny danej pozycji z oceną innego silnika (stockfish)
  - Testy gry przeciw innym silnikom - oszacowanie finalnej siły silnika

Narzędzia

- język programowania: C++
- kompilacja: GCC, CMake
- środowisko: Visual Studio Code
- kontrola wersji: Git
- CI/CD: GitHub Actions
- testy jednostkowe: GoogleTest framework
- testy funkcjonalne: stockfish (perft), python-chess framework

---

- interfejs użytkownika: UCI (Universal Chess Interface) – możliwość integracji z lokalnymi lub internetowymi GUI (np. Arena, CuteChess, Lichess) ✓

Źródła

- Dokumnetacja wymienionych narzędzi (m.in. https://cppreference.com)
- Chess Programming Wiki – internetowa encyklopedia poświęcona programowaniu silników szachowych. (https://www.chessprogramming.org)

AGH

Dziękuję za uwagę.