WIELKI TURNIEJ ALGORYTMICZNY

CZYLI OPOWIEŚĆ O PASOŻYTACH, KONFORMISTACH I PRZODOWNIKACH PRACY

XVI edycja Matematyki dla Ciekawych Świata
Piotr Morawiecki, 18 czerwca

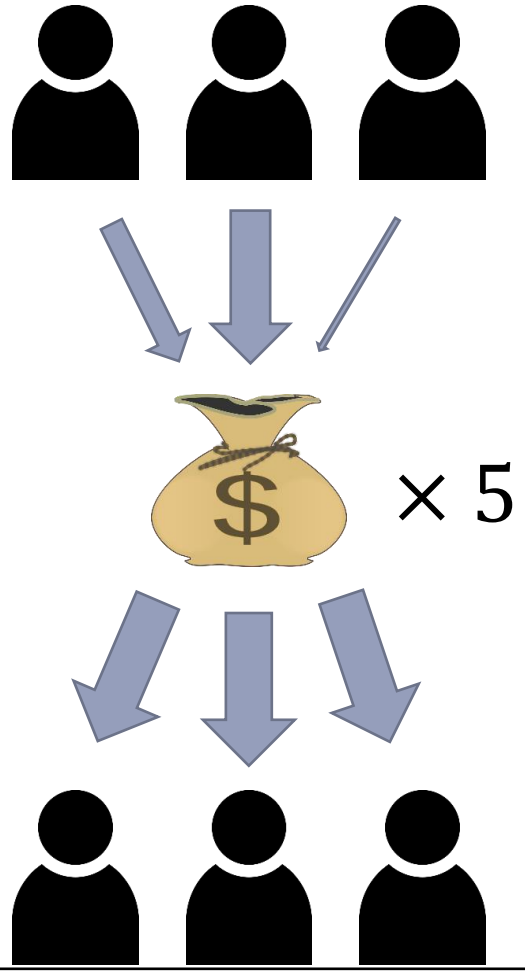
GRA W DOBRA PUBLICZNE

1. Każdy z graczy na początku dysponuje kwotą 10 złotych.

2. Jednocześnie każdy z graczy może przeznaczyć wybraną przez siebie kwotę do wspólnej puli.

3. Pieniądze w puli są mnożone przez 5 i rozdawane po równo wszystkim graczom.

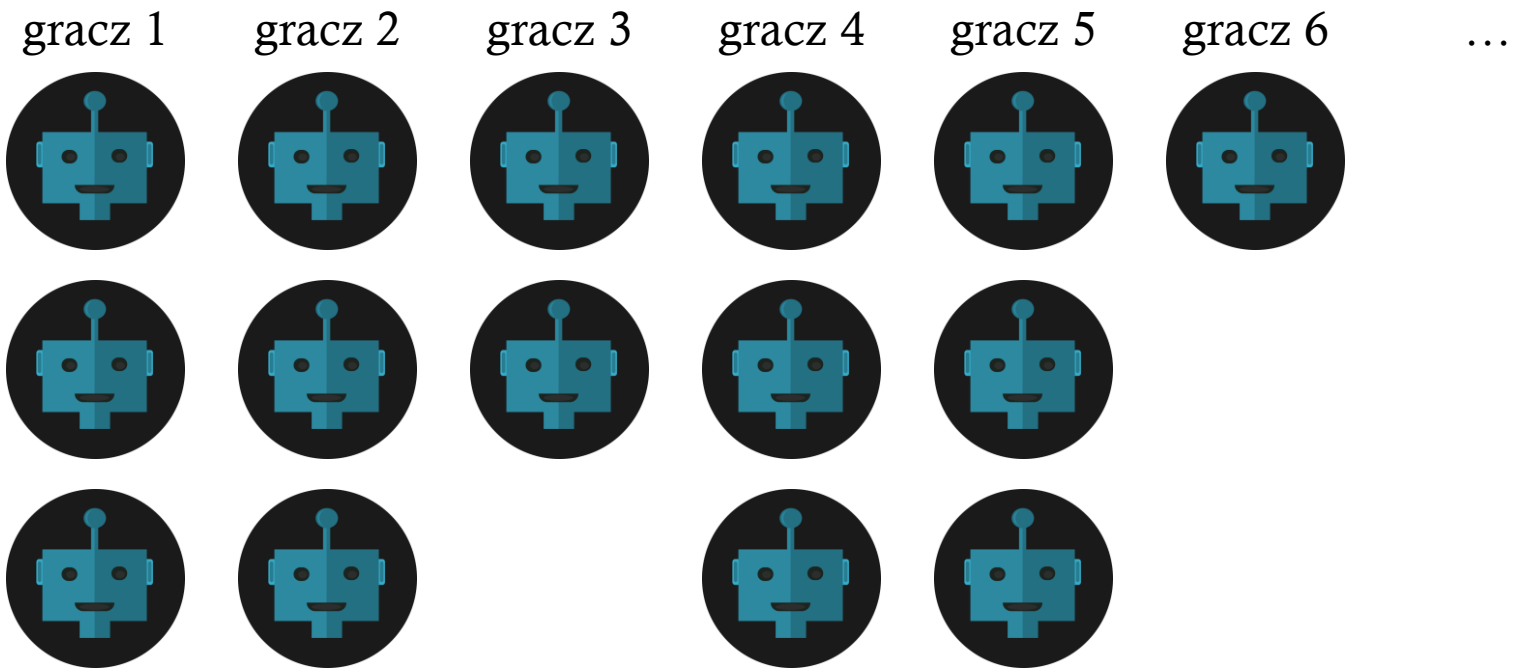
4. Celem graczy jest zakończenie gry z jak największą kwotą.



INTERATYWNA GRA

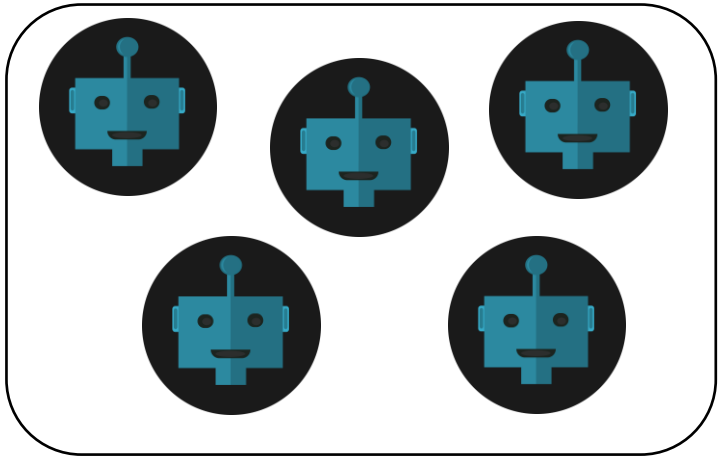
					
Tura 1	10	5	5	10	0
Tura 2	0	7	10	5	0
⋮	⋮	⋮	⋮	⋮	⋮
Tura *N*					

TURNIEJ ALGORYTMICZNY



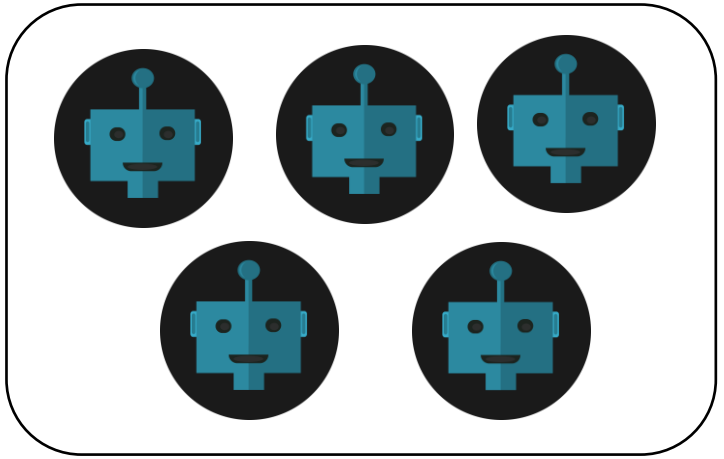
TURNIEJ ALGORYTMICZNY

Rozgrywka #1



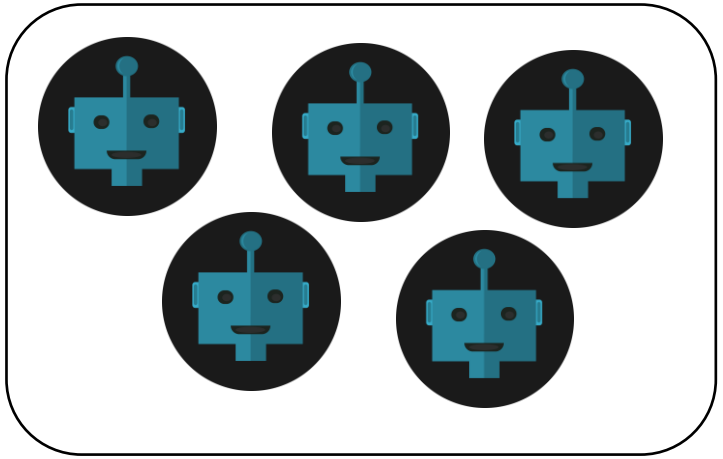
(100 tur)

Rozgrywka #2



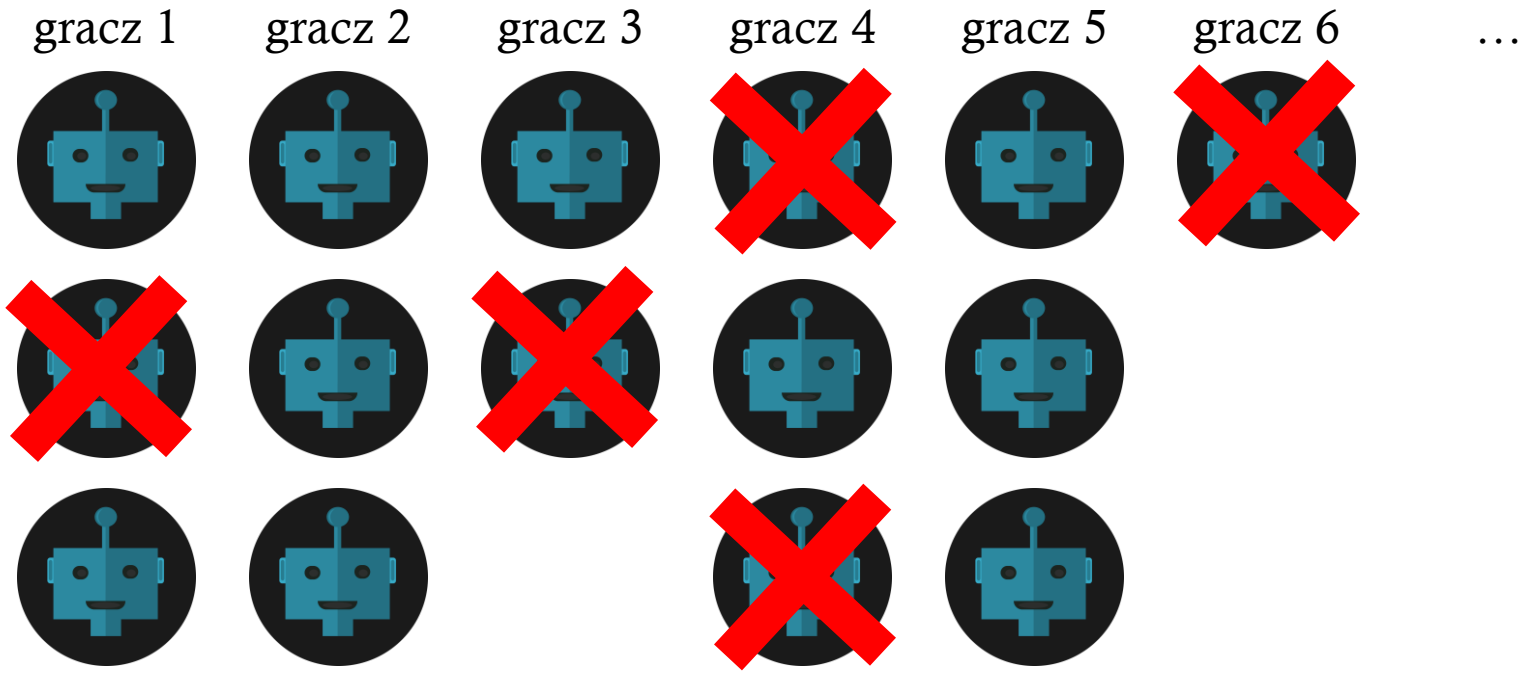
(100 tur)

Rozgrywka #3



(100 tur)

TURNIEJ ALGORYTMICZNY



Runda 1 (59 botów) → Runda 2 (50 botów) → Runda 3 (40 botów) → Runda 4 (30 botów) → Runda 5 (20 botów) → Runda 6 (10 botów)

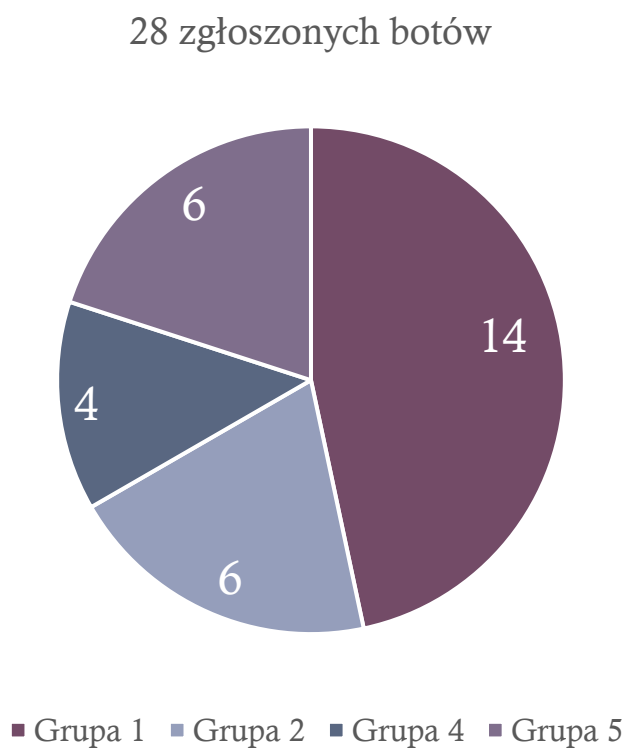
IMPLEMENTACJA STRATEGII BOTA

Lista zawierająca ostatnie akcje rywali

Lista zawierająca wszystkie akcje graczy w danej rundzie

```
def BotInTheMiddle_Julia[redacted]_5(ostatniRuchRywali, historiaAkcji):
 if len(ostatniRuchRywali) ==0:
  return 3
 else:
  srednia = sum(ostatniRuchRywali) / len(ostatniRuchRywali)
  return srednia
```

PODSUMOWANIE ZGŁOSZEŃ




```python
def Bot0 (ostatniRuchRywali, historiaAkcji):
 return 0
```

```python
def CapitalisticCommunist (ostatniRuchRywali, historiaAkcji):
 if len(ostatniRuchRywali) == 0:
  return 10
 else:
  if sum(ostatniRuchRywali)*2/5 < 12:
   return 0
  elif sum(ostatniRuchRywali)*2/5 < 16:
   return historiaAkcji[len(historiaAkcji) - 1][0] - 1
  else:
   return 10
```

```python
def AdaptacyjnyObserwator (ostatniRuchRywali, historiaAkcji):
 def srednia(lista):
  return sum(lista) / len(lista) if lista else 0.0

 def odchylenie_standardowe(lista):
  if len(lista) < 2:
   return 0.0
  avg = sum(lista) / len(lista)
  wariancja = 0.0
  for x in lista:
   wariancja += (x - avg) ** 2
  wariancja /= (len(lista) - 1)
  return wariancja ** 0.5

 if not historiaAkcji:
  return 7.0

 moja_ostatnia = historiaAkcji[-1][0]
 suma_wplat = sum(historiaAkcji[-1])
 liczba_rywali = len(ostatniRuchRywali)

 historia_skutecznosci = []
 for runda in historiaAkcji:
  moja_wplata = runda[0]
  suma = sum(runda)
  if moja_wplata > 0:
   skut = (0.4 * suma) / moja_wplata
   historia_skutecznosci.append(skut)
 if historia_skutecznosci:
  prog_skutecznosci = sum(historia_skutecznosci) / len(historia_skutecznosci)
 else:
  prog_skutecznosci = 1.0

 moja_wyplata = 0.4 * suma_wplat
 skutecznosc = moja_wyplata / (moja_ostatnia + 1e-6)

 if len(historiaAkcji) >= 2:
  trend = srednia(ostatniRuchRywali) - srednia(historiaAkcji[-2][1:])
 else:
  trend = 0.0

 rozrzut = odchylenie_standardowe(ostatniRuchRywali)

 srednia_rywali = srednia(ostatniRuchRywali)
 prog_mobilizacji = 3.0

 if skutecznosc >= prog_skutecznosci:
  if trend > 1.5:
   nowa_akcja = moja_ostatnia + 1.5
  else:
   nowa_akcja = moja_ostatnia

  if rozrzut > 3.0:
   nowa_akcja *= 0.85

 else:
  kara = (1.0 - skutecznosc) * 3.0
  kara += (rozrzut / 5.0)

  nowa_akcja = moja_ostatnia - kara + trend * 0.5

  if srednia_rywali < prog_mobilizacji:
   nowa_akcja *= 0.6

 if nowa_akcja < 0.0:
  nowa_akcja = 0.0
 elif nowa_akcja > 10.0:
  nowa_akcja = 10.0

 return nowa_akcja
```

BOTY DARMOZJADY

```
def Niszczyciel1_Janel[redacted]_1(ostatniRuchRywali, historiaAkcji):
  return 0


def Niszczyciel2_Janek[redacted]_1(ostatniRuchRywali, historiaAkcji):
  return 0


def Niszczyciel3_Janek[redacted]_1(ostatniRuchRywali, historiaAkcji):
  return 0
```

STRAŻNICY TEKSASU

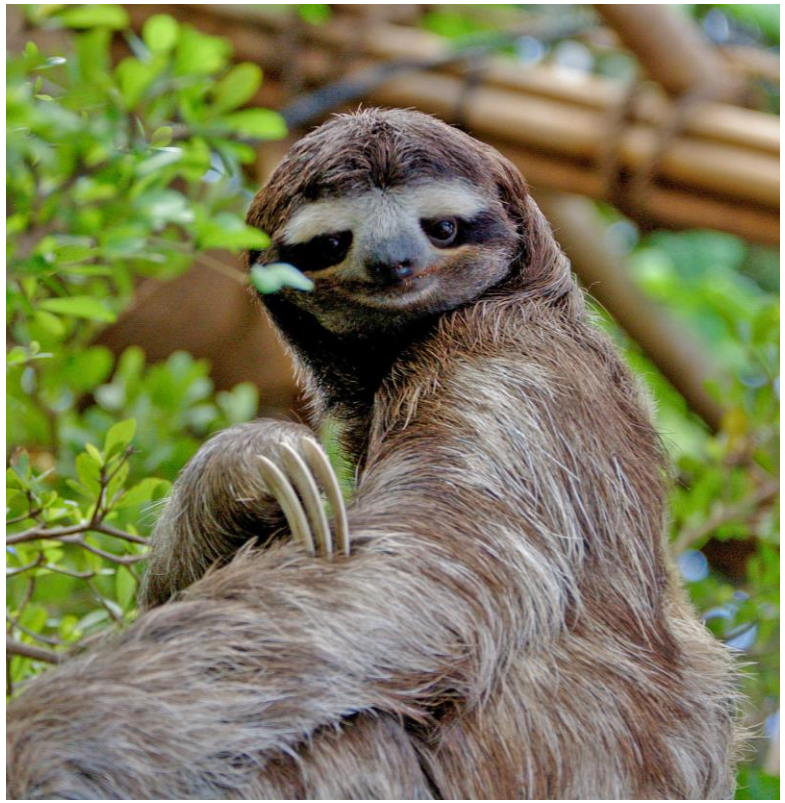
```
def GeometryDash_Witold[redacted]_1(ostatniRuchRywali, historiaAkcji):
  if len(ostatniRuchRywali) < 4:
    return 2
  else:
    iloczyn=1
    for i in range(4):
      iloczyn*=ostatniRuchRywali[i]
    return iloczyn**0.25

def LipidiLajn_Jonasz[redacted]_1(ostatniRuchRywali, historiaAkcji):
  if len(ostatniRuchRywali) == 0:
    return 0
  else:
    ostatniRuchRywali.sort()
    return ostatniRuchRywali[1]
```

KONFORMIŚCI

```
def Bot1_Krystiar[redacted]_1(ostatniRuchRywali, historiaAkcji):
  return 0 if len(ostatniRuchRywali) == 0 else statistics.median(ostatniRuchRywali)


def Mediocrity1_Mykhailo[redacted]_1(ostatniRuchRywali, historiaAkcji):
  if len(ostatniRuchRywali) == 0:
    return 5
  else:
    return sum([sum(i) / len(i) for i in historiaAkcji + [ostatniRuchRywali]]) /
      len(historiaAkcji + [ostatniRuchRywali])


def BotSredniak1_Daria[redacted]_4(opponentMoves, historiaAkcji):
  if len(opponentMoves)==0:
    return 5
  else:
    if 2*(sum(opponentMoves)+10)/(len(opponentMoves)+1) < 10:
      return 0
    else:
      return (sum(opponentMoves))/(len(opponentMoves))
```

LESERZY

```
def BotMinimalny_Karolina[redacted]_5(ostatniRuchRywali, historiaAkcji):
  if len(historiaAkcji) == 0:
    return 5
  else:
    return max(0, min(ostatniRuchRywali)-0.5)

def Dariusz_WitoldI[redacted]_5(ostatniRuchRywali, historiaAkcji):
  if len(ostatniRuchRywali) == 0:
    return 0
  elif (average(ostatniRuchRywali) - 1) >= 0:
    return average(ostatniRuchRywali) - 1
  else:
    return average(ostatniRuchRywali)
```

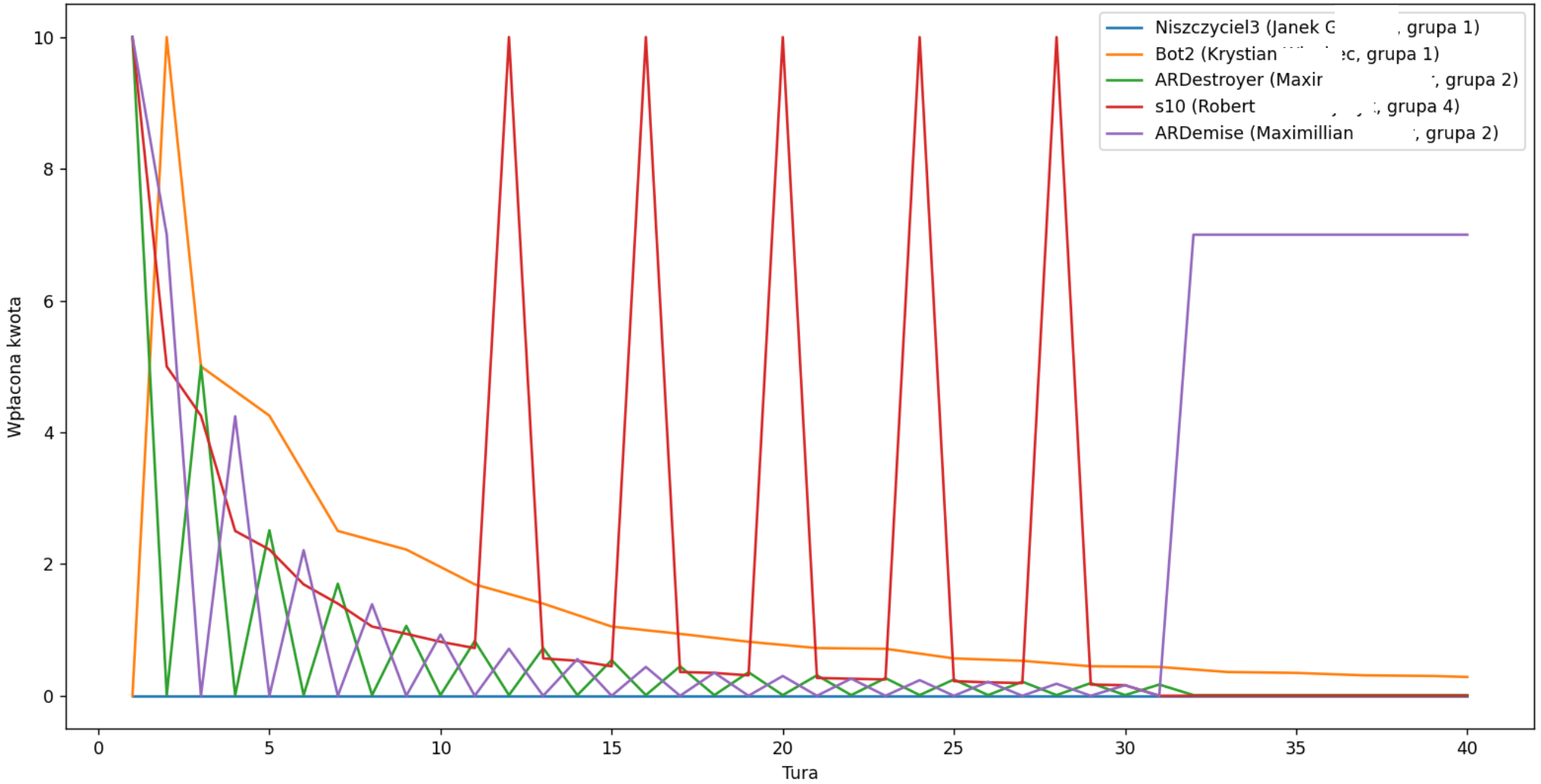


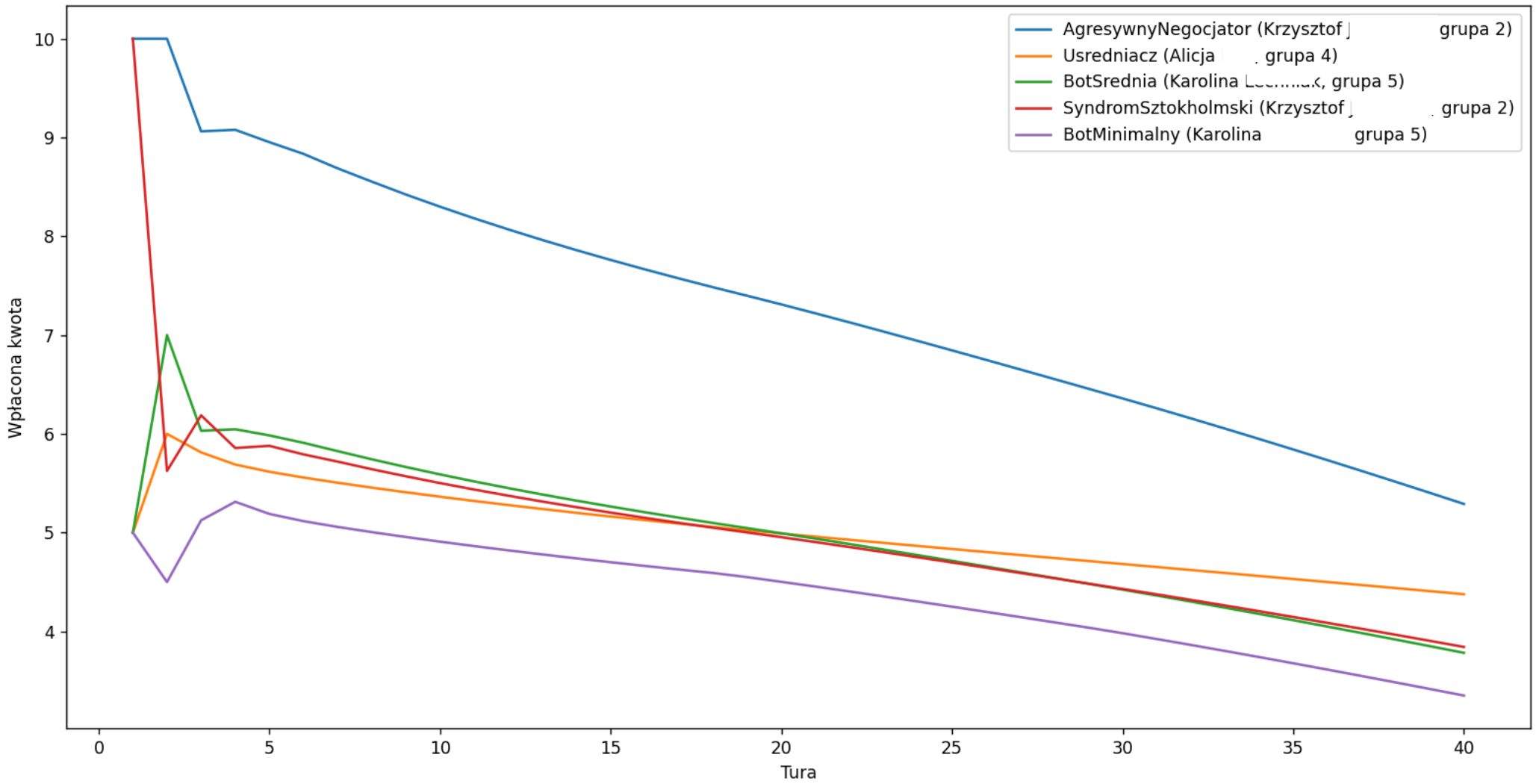
PRZODOWNICY PRACY

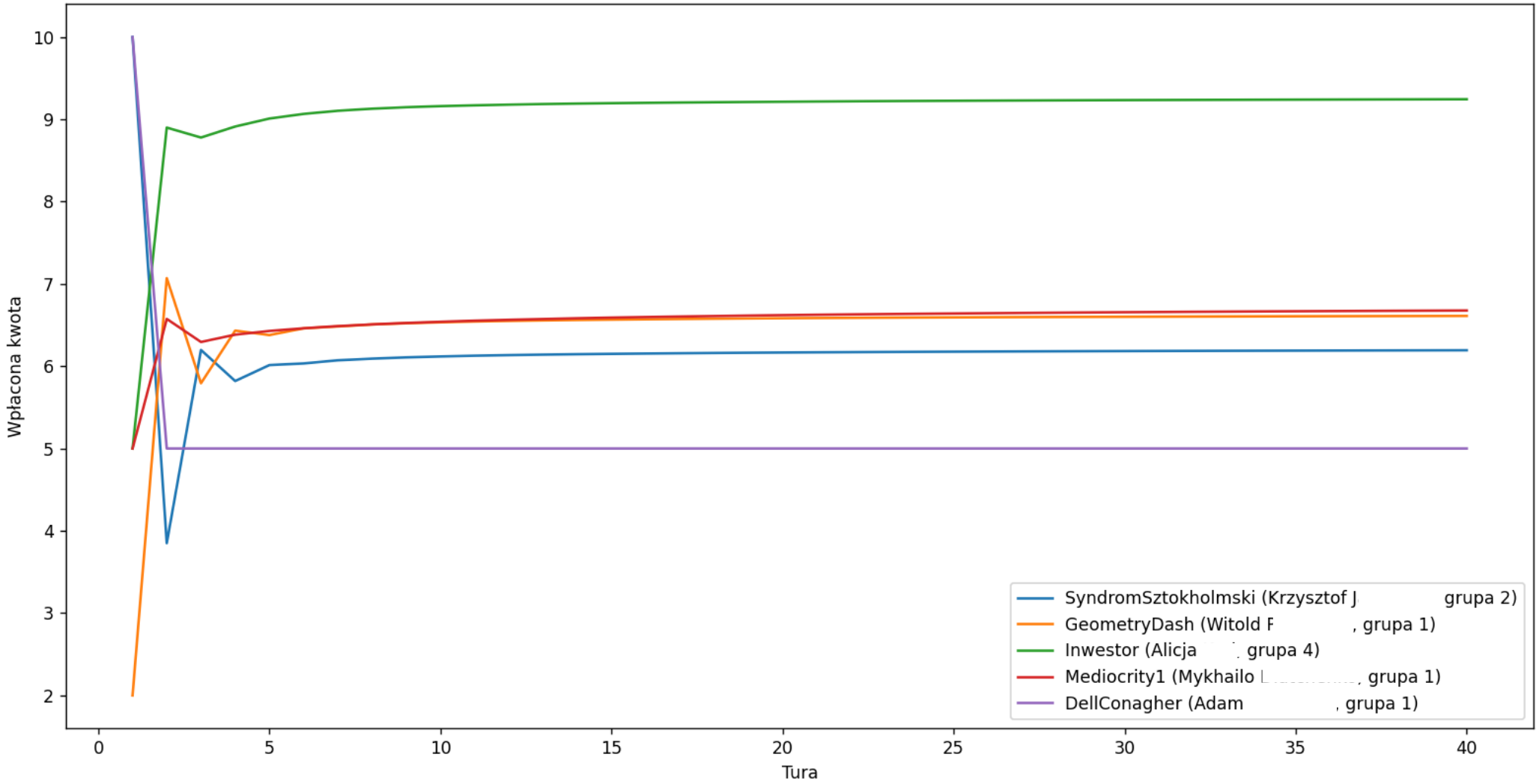

```
def Milioner_Mateusz[redacted]_5(ostatniRuchRywali, historiaAkcji):
 if len(ostatniRuchRywali) == 0:
  return 10
 else:
  return 10


def ARDemise_Maxi[redacted]_2(ostatniRuchRywali, historiaAkcji):
 if len(ostatniRuchRywali) == 0:
  return 10
 X = [0, 0, 0, 0]
 for i in range(4):
  X[i] = ostatniRuchRywali[i]

 X.sort()

 if X[1] - 0.01 < 0:
  return 7
 else:
  return X[1] - 0.01
```

AGENCI CHAOSU

```
def RandomUniform_Julia[redacted]_5(ostatniRuchRywali, historiaAkcji):
  x= random.uniform(0,10)
  if len(ostatniRuchRywali) == 0:
    if x>=5:
      return 5
    else:
      return 0
  else:
    if x<=3:
      return 3
    elif x<=6:
      return 6
    else:
      return 0
```


Niszczyciel3 (Janek G , grupa 1)
Bot2 (Krystian ec, grupa 1)
ARDestroyer (Maxir , grupa 2)
s10 (Robert , grupa 4)
ARDemise (Maximillian , grupa 2)
Wpłacona kwota
Tura
0
2
4
6
8
10
0
5
10
15
20
25
30
35
40


AgresywnyNegocjator (Krzysztof J grupa 2)
Usredniacz (Alicja . grupa 4)
BotSrednia (Karolina L, grupa 5)
SyndromSztokholmski (Krzysztof J . grupa 2)
BotMinimalny (Karolina grupa 5)
Wpłacona kwota
Tura
10
9
8
7
6
5
4
0
5
10
15
20
25
30
35
40


Wpłacona kwota
Tura
SyndromSztokholmski (Krzysztof J grupa 2)
GeometryDash (Witold F , grupa 1)
Inwestor (Alicja grupa 4)
Mediocrity1 (Mykhailo , grupa 1)
DellConagher (Adam , grupa 1)

ZWYCIĘZCA RUNDY 1 (KATEGORIA OPEN):

ROBERT

(GRUPA 4)

S10

```
def s10_Rober[redacted]_4(ostatniRuchRywali, historiaAkcji):
  L = len(ostatniRuchRywali)  # ilość przeciwników
  counter = len(historiaAkcji)
  strat = 1
  tol = 1.5

  if counter >= 3:
    temp = historiaAkcji[-1] + historiaAkcji[-2] + historiaAkcji[-3]
    check1 = True
    check2 = False
    av = sum(temp) / len(temp)
    if av >= 9:
      check1 = False
    elif av <= 6 and counter >= 30:
      check1 = False
      check2 = True
    else:
      for i in temp:
        if av - tol < i < av + tol:
          continue
        else:
          check1 = False
    if check1:
      strat = 0
    elif check2:
      strat = 2
    else:
      strat = 1

  if historiaAkcji == []:
    return 10

  if strat == 0:
    return 10
  if strat == 1:
    return sum(ostatniRuchRywali) / L
  if strat == 2:
    return 0
```

ZWYCIĘZCA RUNDY 2:

MAXIMILLIAN

(GRUPA 2)

ARRUIN

```
def ARRuin_Maximillian[redacted]_2(ostatniRuchRywali, historiaAkcji):
 if len(ostatniRuchRywali)==0:
  return 10
 X=[0,0,0,0]
 for i in range(4):
  X[i]=ostatniRuchRywali[i]
 def Babelkowe(X):
  n=len(X)
  for i in range(n-1):
   for j in range(0,n-i-1):
    if X[j]>X[j+1]:
     X[j], X[j+1]=X[j+1], X[j]
 Babelkowe(X)
 if X[1]+0.01>10:
  return 10
 else:
  return X[1] + 0.01
```

ZWYCIĘZCA RUNDY 3:

MIKOŁAJ

(GRUPA 1)

B2

```
def b2_Mikolaj[redacted]_1(past, hist):
  if len(hist) == 0: return 10
  h = hist[-(min(len(hist), 4)):]
  ah = 0
  for i in h:
    ah += sum(i) / len(i)
  a = ah / len(h)
  return a
```

ZWYCIĘZCA RUNDY 4:

ALICJA
(GRUPA 4)

INWESTOR

```
def Inwestor_Alicja[redacted]_4(ostatniRuchRywali, historiaAkcji):
 startValue = 5
 const = 0.5
 if len(ostatniRuchRywali) == 0: return startValue
 lastMoves = historiaAkcji[-1]
 playerLastMove = startValue
 for move in ostatniRuchRywali:
  if move not in lastMoves:
   playerLastMove = move
 profit = sum(historiaAkcji[-1]) * 2 / (len(ostatniRuchRywali) + 1) - playerLastMove
 valReturn = playerLastMove + profit * const
 if valReturn < 1:
  return 0
 return max(0, min(valReturn, 10))
```

ZWYCIĘZCA PÓŁFINAŁU:

ADAM

(GRUPA 1)

DELL CONAGHER

```
def DellConagher_Adam [redacted]_1(ostatniRuchRywali, historiaAkcji):
 friendliness = [0.5, 0.5, 0.5, 0.5]
 avg = []
 if len(ostatniRuchRywali) == 0:
  return 10
 else:
  for i in range(4):
   if ostatniRuchRywali[i] < 5:
    friendliness[i] -= 0.2
   else:
    friendliness[i] += 0.3

  for i in range(4):
   avg.append(ostatniRuchRywali[i]*friendliness[i])

  if sum(avg)/(sum(friendliness) + 0.1) < 3.5:
   return 0
  elif sum(avg)/(sum(friendliness) + 0.1) < 5:
   return 1
  elif sum(avg)/(sum(friendliness) + 0.1) < 7:
   return 5
  else:
   return 9
```

ZWYCIĘZCA FINAŁU:

KRYSTIAN
(GRUPA 4)

BULLY DETECTOR

```
def BullyDetector_Krystian[redacted]_4(opponentMoves, historyOfMoves):
 if not historyOfMoves:
  return 5.0

 numOpponents = len(opponentMoves)
 opponentHistories = [[] for _ in range(numOpponents)]

 for pastRound in historyOfMoves:
  for i in range(numOpponents):
   opponentHistories[i].append(pastRound[i])

 totalRounds = len(historyOfMoves)
 baseThreshold = 0.6
 minThreshold = 0.3
 dynamicThreshold = max(minThreshold, baseThreshold - (totalRounds * 0.01))

 totalSuspicion = 0.0

 for history in opponentHistories:
  if not history:
   continue
  lowContribRatio = sum(1 for move in history if move < 2) / len(history)
  if lowContribRatio > dynamicThreshold:
   totalSuspicion += 1.0
  else:
   totalSuspicion += lowContribRatio / dynamicThreshold

 suspicionLevel = min(1.0, totalSuspicion / numOpponents)
 contribution = (1.0 - suspicionLevel) * 5.0

 return contribution
```

DZIĘKUJĘ WSZYSTKIM ZAWODNIKOM!