**Piotr Kica**

# Dokumentacja

- Program napisany został w języku Python bez wykorzystania bibliotek funkcyjnych (czyli max punktów to 9.5).
- Szczegóły implementacji danej funkcji zwarłem w kodzie z wykorzystaniem komentarzy i docstringów.
- Do stworzenia grafu wykorzystałem bibliotekę *networkx*, do rysowania bibliotekę *matplotlib*.
- W kodzie znajduje się gotowy plik main z przykładowym wywołaniem, oraz zostały przygotowane unittesty do większości funkcji.
- Potrzebne biblioteki: *networkx*, *matplotlib*, *numpy*
- Symbole transakcji są jednoliterowe

## 1. Relacja D, relacja I:

Do wyznaczenia relacji (nie)zależności służy funkcja

```
def D_I_relations(transactions: list) -> (set, set):
```

która zwraca parę (D, I).

**Przykład użycia:**

a = Transaction("a", "x = x + y")
b = Transaction("b", "y = y + 2z")
c = Transaction("c", "x = 3x + z")
d = Transaction("d", "z = y - z")
D, I = D_I_relations([a, b, c, d])

**Wynik:**

D = {('a', 'a'), ('a', 'b'), ('a', 'c'), ('b', 'a'), ('b', 'b'), ('b', 'd'), ('c', 'a'), ('c', 'c'), ('c', 'd'), ('d', 'b'), ('d', 'c'), ('d', 'd')}
I = {('c', 'b'), ('d', 'a'), ('b', 'c'), ('a', 'd')}

## 2. Postać Normalna Foaty na podstawie śladu [w]:

Do wyznaczenia FNF służy funkcja:

```
def FNF_word(word: str, D: set) -> dict:
```

**Przykład użycia:**

FNF_word("adhcbgfae", D)

**Wynik:**

FNF = {0: ['a', 'd', 'h'], 1: ['c', 'g'], 2: ['b', 'f'], 3: ['a', 'e']}

## 3. Minimalny graf zależności - Diekerta:

Do wyznaczenia grafu służy funkcja:

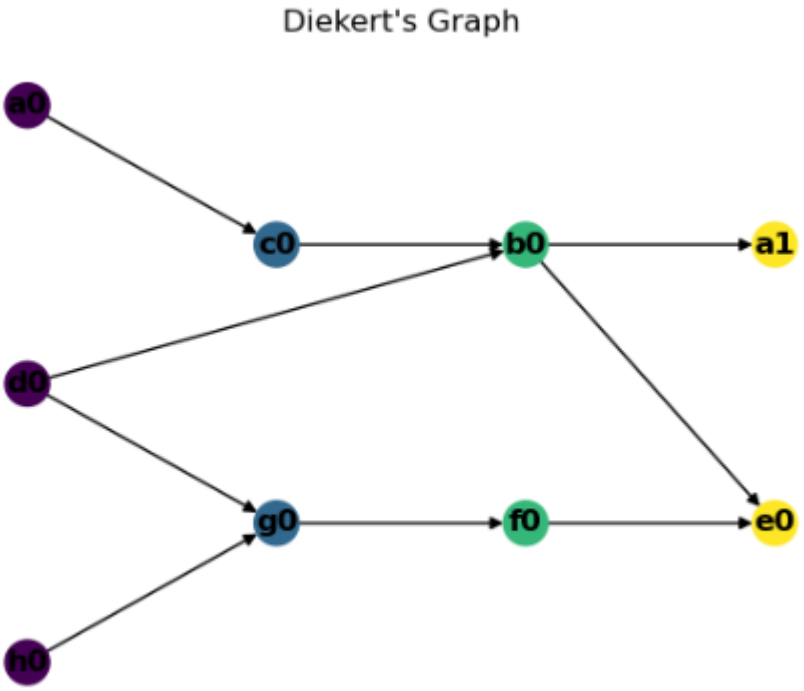
```
def diekertGraph(original_word: str, D: set) -> nx.DiGraph:
```

**Przykład użycia:**

G = diekertGraph("adhcbgfae", D)
draw_graph(G)

**Wynik:**

G to obiekt biblioteki networkx będący grafem acyklicznym skierowanym.




## 4. Postać Normalna Foaty na podstawie grafu Diekerta:

Do wyznaczenia FNF służy funkcja:

```
def FNF_graph(G: nx.DiGraph) -> dict:
```

**Przykład użycia:**

G = diekertGraph("adhcbgfae", D)
FNF = FNF_graph(G)

**Wynik:**

FNF = {0: ['a', 'd', 'h'], 1: ['c', 'g'], 2: ['b', 'f'], 3: ['a', 'e']}

**Pozostałe funkcje i klasy:**

1. `class Transaction:`
   Klasa reprezentująca pojedynczą transakcję.

   **Przykład wywołania:**
   Transaction("a", "x = x + y", alphabet)

2. `def draw_graph(G: nx.DiGraph):`
   Rysowanie grafu z wykorzystaniem networkx i matplotlib

   **Przykład wywołania:**
   G = diekertGraph("adhcbgfae", D)
   draw_graph(G)

3. `def from_file(filename: str) -> [set, list]:`
   Wczytanie danych z pliku.
   Przykładowy format pliku:

   a b c d e f g h

   a: x = 0
   b: x = x * y
   c: x = x + 2
   d: y = 1
   e: y = y - z
   f: z = z * v
   g: z = v + y
   h: v = z - v

   gdzie pierwsza linii to alfabet, a od trzeciej linii są wypisane produkcje.

   **Przykład wywołania:**
   alphabet, transactions = from_file("input.txt")

4. `def calc_and_save(transactions: list, word: str, D_I_FNF_filename: str, graph_filename: str):`
   Obliczenie relacji (nie)zależności, FNF, grafu Diekerta.
   Zapisanie ich do dwóch plików, graf ma osobny plik w formacie .dot.

   **Przykład wywołania:**
   calc_and_save(transactions, "adhcbgfae", "output.txt", "graph.dot")