Kryptografia i kryptoanaliza

Laboratorium 3

Michał Łaskawski

Zadanie 1

Zrealizować program implementujący podstawieniowy algorytm szyfrowania.

1. Wybrać dłuższy fragment tekstu w języku angielskim (np. akapit składający się z kilkunastu zdań).
2. Usunąć z niego wszystkie znaki niebędące literami (ograniczenie do 26 liter alfabetu łacińskiego).
3. Zaszyfrować tekst używając wybranego w sposób losowy klucza (tablicy podstawień): permutacji $\hat{\pi}$.

Zadanie 2

Dokonać kryptoanalizy heurystycznej na zaimplementowany w ramach pierwszego zadania monoalfabetyczny kryptosystem podstawieniowy. Założenia ataku są następujące:

1. Znany jest szyfrogram.
2. Wiadomo jaki kryptosystem użyty został do zaszyfrowania wiadomości.
3. Należy odzyskać klucz i tekst jawny.

Do realizacji zadania kryptoanalizy heurystycznej, należy użyć algorytmu *Metropolis-Hastings*. Algorytm ten umożliwia realizację procedury poszukiwanie klucza (czyli mapowania znaków), który najlepiej dopasowuje się do rozkładu prawdopodobieństwa tekstu jawnego. Przebieg tego procesu wygląda następująco:

1. Wybieramy reprezentację klucza jako permutację $\hat{\pi}$:
   - Zakładamy losowy klucz początkowy, czyli permutację znaków klucza.
2. Definiujemy funkcję oceny klucza (funkcja celu / funkcja wiarygodności) $\mathrm{pl}(\hat{\pi})$:
   - Funkcja ta jest miarą tego jak bardzo odszyfrowany tekst przypomina tekst naturalny.
   - Na potrzeby zadania, funkcja ta zdefiniowana jest w następujący sposób: $\mathrm{pl}\left(\hat{\pi}\right) = \prod_{i,j} \left(M_{i,j}\right)^{\hat{M}_{i,j}}$. gdzie:
     - M to macierz bigramów utworzona na bazie tekstu referencyjnego, natomiast $M_{i,j}$ to liczba wystąpień pary (i, j) w tekście referencyjnym.
     - $\hat{M}$ to macierz bigramów utworzona na bazie szyfrogramu, natomiast $\hat{M}_{i,j}$ to liczba wystąpień pary (i, j) w szyfrogramie.
   - Uwaga zdefiniowaną funkcję należy rozpatrywać w kategorii prawdopodobieństwa.
3. Losujemy nową permutację klucza $\hat{\pi}'$:
   - Zadanie to realizowane jest poprzez losową zamianę dwóch znaków z permutacji (klucza) $\hat{\pi}$.
4. Definiujemy kryterium akceptacji $\rho(\hat{\pi}, \hat{\pi}')$:
   - Algorytm *Metropolis-Hastings* akceptuje nowy klucz z pewnym prawdopodobieństwem, które zależy od stosunku funkcji oceny dla nowego i starego klucza. Jeśli nowy klucz $\hat{\pi}'$ prowadzi do lepszego dopasowania, to akceptujemy go jako X_{t+1}, jeśli nie, to zostajemy przy starym $\hat{\pi}$ kluczu.
   - Dla rozważanego przypadku, kryterium akceptacji można zdefiniować w następujący sposób: $\rho(\hat{\pi}, \hat{\pi}') = \frac{\mathrm{pl}(\hat{\pi}')}{\mathrm{pl}(\pi)}$.
   - Dla rozważanego algorytmu, należy wylosować liczbę u należącą do rozkładu jednostajnego na przedziale $[0, 1]$ a następnie dokonać porównania: $u \leq \rho(\hat{\pi}, \hat{\pi}')$. Jeżeli warunek ten jest spełniony to akceptujemy nowy klucz (permutację), jeżeli nie to zostajemy przy starym kluczu.

5. Iteracja procesu:
    - Proces ten jest powtarzany, tworząc łańcuch kluczy $\{X_t : t = 0, \ldots, T\}$, które przybliżają optymalne rozwiązanie.
    - W miarę postępu procesu iteracyjnego, algorytm koncentruje się na obszarach rozwiązań, które lepiej odtwarzają tekst jawny.

Algorytm *Metropolis-Hastings* dla rozważanego problemu, przyjmuje następującą postać:

Algorithm 1 MH

$t \leftarrow 0$
$X_0 \leftarrow \hat{\pi}_0$
for $t = 1, \ldots, T$ **do**
    dla $X_t \leftarrow \hat{\pi}$
    wygeneruj $i, j \sim U(\{1, 2, \ldots, 26\})$ ▷ $\sim$ znaczy ma rozkład
    wygeneruj $\hat{\pi}'$ ▷ zamieniając znaki na pozycjach i oraz j w kluczu $\hat{\pi}$
    $\rho(\hat{\pi}, \hat{\pi}') \leftarrow \frac{\text{pl}(\hat{\pi}')}{\text{pl}(\pi)}$ ▷ ρ - prawdopodobieństwo akceptacji
    wygeneruj $u \sim U([0, 1])$
    if $u \leq \rho(\hat{\pi}, \hat{\pi}')$ **then**
        $X_{t+1} \leftarrow \hat{\pi}'$
    else
        $X_{t+1} \leftarrow \hat{\pi}$
    end if
end for

Uwagi:

- Wyznaczenie funkcji pl, może prowadzić do przekroczenia zakresu numerycznego. Aby uniknąć problemów związanych z precyzją numeryczną, można zastosować logarytmowanie funkcji pl.

$$\log \text{pl}\left(\hat{\pi}\right) = \sum_{i,j} \hat{M}_{i,j} \cdot \log M_{i,j}$$

- Uwaga, przekształcenie to w konsekwencji poprowadzi do innego sposobu wyznaczenia współczynnika akceptacji poprzez obliczenie wartości funkcji wykładniczej, której argumentem jest różnica pomiędzy $\text{pl}(\hat{\pi}')$ a $\text{pl}(\hat{\pi})$.

  Zamiast obliczać: $\rho(\hat{\pi}, \hat{\pi}') = \frac{\text{pl}(\hat{\pi}')}{\text{pl}(\pi)}$ należy obliczyć:

$$\rho\left(\hat{\pi}, \hat{\pi}'\right) = \exp\left[\log \text{pl}\left(\hat{\pi}'\right) - \log \text{pl}\left(\hat{\pi}\right)\right]$$

  Należy również zauważyć, iż współczynnik akceptacji nie powinien być większy do jedności.

- Jest to powszechny sposób stabilizacji obliczeń w algorytmach probabilistycznych.

Zadanie 3

Dokonać kryptoanalizy heurystycznej na zaimplementowany w ramach pierwszego zadania monoalfabetyczny kryptosystem podstawieniowy. Założenia ataku są takie jak w zadaniu 2. Do ataku wykorzystać algorytm optymalizacji *Symulowanego Wyżarzania* (ang. *Simulated Annealing*).

Algorytm symulowanego wyżarzania

Symulowane wyżarzanie to metoda optymalizacji inspirowana procesem fizycznym zwanym wyżarzaniem, który jest stosowany w metalurgii i krystalografii. Proces ten polega na podgrzewaniu materiału do wysokiej temperatury, a następnie stopniowym chłodzeniu w celu osiągnięcia stanu minimalnej energii. Algorytm

symulowanego wyżarzania wykorzystuje tę koncepcję, aby znaleźć rozwiązanie problemu optymalizacyjnego, które minimalizuje (lub maksymalizuje) pewną funkcję celu.

Ogólny opis algorytmu

1. **Inicjalizacja**:
   - Algorytm zaczyna od pewnego początkowego rozwiązania (np. losowego lub opartego na pewnej heurystyce).
   - Ustalana jest początkowa temperatura, która kontroluje prawdopodobieństwo akceptacji gorszych rozwiązań.
2. **Główna pętla**:
   - W każdej iteracji algorytm generuje nowe rozwiązanie poprzez niewielką modyfikację bieżącego rozwiązania (np. zamiana dwóch elementów w mapowaniu liter).
   - Obliczana jest różnica wyników między nowym a aktualnym rozwiązaniem.
     - Jeśli nowe rozwiązanie jest lepsze (ma wyższy wynik), to zostaje zaakceptowane.
     - Jeśli nowe rozwiązanie jest gorsze, to może ono zostać zaakceptowane z pewnym prawdopodobieństwem, które zależy od różnicy wyników i aktualnej temperatury. To prawdopodobieństwo jest obliczane jako:

$$P = \exp\left(\frac{\text{score\_diff}}{\text{temperature}}\right)$$

3. **Schładzanie**:
   - Temperatura jest stopniowo zmniejszana zgodnie z określoną regułą chłodzenia (np. mnożenie przez współczynnik chłodzenia, który jest mniejszy od 1). W miarę spadku temperatury algorytm staje się mniej skłonny do akceptowania gorszych rozwiązań.
4. **Zakończenie**:
   - Algorytm kończy się po osiągnięciu określonej liczby iteracji lub po osiągnięciu bardzo niskiej temperatury, kiedy przestaje akceptować gorsze rozwiązania.
   - Ostateczne rozwiązanie jest uważane za przybliżone optimum globalne.

Opis algorytmu w kontekście ataku na analizowany kryptosystem

Dane wejściowe:

- c - szyfrogram,
- π_0 - początkowe mapowanie,
- g - n-gramy referencyjne (bigramy),
- N - maks. iteracje,
- T_0 -początkowa temperatura,
- α współczynnik chłodzenia

Opis Symboli:

- c: Szyfrogram, który chcemy odszyfrować.
- π: Permutacja, mapowanie liter (klucz szyfrujący).
- g: Referencyjne n-gramy (bigramy).
- S_{curr}: Aktualna wartość funkcji celu.

Algorithm 2 SA

$\pi \leftarrow \pi_0$
$S_{\text{curr}} \leftarrow f_{c,g}(\pi)$
$\pi_{\text{best}} \leftarrow \pi$
$S_{\text{best}} \leftarrow S_{\text{curr}}$
$T \leftarrow T_0$
for $k = 1$ to N **do**
    $a \sim U(\mathcal{A})$, $b \sim U(\mathcal{A} \setminus \{a\})$ ▷ $\mathcal{A} = \{A, B, \ldots, Z\}$
    $\pi' \leftarrow \sigma_{a,b}(\pi)$
    $S_{\text{new}} \leftarrow f_{c,g}(\pi')$
    $\Delta S \leftarrow S_{\text{new}} - S_{\text{curr}}$
    if $\Delta S > 0$ **or** $\exp\left(\frac{\Delta S}{T}\right) > u$ dla $u \sim \mathrm{U}(0,1)$ **then**
        $\pi \leftarrow \pi'$
        $S_{\text{curr}} \leftarrow S_{\text{new}}$
        if $S_{\text{curr}} > S_{\text{best}}$ **then**
            $\pi_{\text{best}} \leftarrow \pi$
            $S_{\text{best}} \leftarrow S_{\text{curr}}$
        end if
    end if
    $T \leftarrow \alpha \times T$
end for
Return: π_{best}

- S_{new}: Nowa wartość funkcji celu po zamianie liter.
- T: Temperatura, która kontroluje prawdopodobieństwo akceptacji gorszych rozwiązań.
- $u \sim \mathrm{U}(0,1)$: Liczba losowa z rozkładu jednostajnego na przedziale $[0,1]$.
- $\sigma_{a,b}(\pi)$: funkcja losowej zamiany miejscami liter w permutacji π.
- $f_{c,g}(\pi)$: Funkcja celu, może one być zdefiniowana w następujący sposób:

$$f_{c,g}(\pi) : \sum_{i=1}^{n} \nu_i \cdot \frac{\phi_i}{\max \Phi}$$

  gdzie:

  - ν_i: oznacza częstość i-tego bigramu w analizowanym szyfrogramie c, który odszyfrowany został przy pomocy klucza: π.
  - ϕ_i: oznacza częstość i-tego bigramu w rozkładzie referencyjnym (bigramy: g).
  - $\max \Phi$: to maksymalna wartość w rozkładzie referencyjnym bigramów $\Phi = \{\phi_1, \phi_2, \ldots, \phi_n\}$.

  Należy zwrócić uwagę na to, iż funkcja $f_{c,g}(\pi)$ implementuje algorytm normalizacji (dzielenie przez $\max \Phi$), dzięki temu eliminuje się potencjalny problem z dokładnością numeryczną.

Uwaga. W celu poprawy stabilności uzyskiwanych wyników oraz unikania lokalnych maksimów, algorytm symulowanego wyżarzania należy uruchamiać wielokrotnie. Za każdym razem inicjując algorytm inną permutacją klucza. Następnie porównywać uzyskany wynik z poprzednio uzyskanym wynikiem i zachowując lepszy. Symbolicznie, podejście to można opisać w następujący sposób:

gdzie:

- $\Lambda_{\max}$: Najlepszy dotychczasowy wynik funkcji celu, reprezentujący maksymalną wartość oceny spośród wszystkich iteracji.
- Θ_{opt}: Najlepsze znalezione rozwiązanie (mapowanie liter - klucz), które maksymalizuje funkcję celu.
- k: Indeks iteracji, liczba całkowita biegnąca od 1 do N, gdzie N to liczba restartów algorytmu.

Algorithm 3 Optymalizacja z wieloma restartami

$\Lambda_{\max} \leftarrow -\infty$
$\Theta_{\text{opt}} \leftarrow \varnothing$
for $k \leftarrow 1$ **to** N **do**
    $\pi_k \sim U(\mathcal{A}_{26})$ ▷ Losowa permutacja z $\mathcal{A}_{26}$, zbioru 26 liter
    $\Psi_k, \Lambda_k \leftarrow \text{Optimize}(\pi_k)$
    if $\Lambda_k > \Lambda_{\max}$ **then**
        $\Lambda_{\max} \leftarrow \Lambda_k$
        $\Theta_{\text{opt}} \leftarrow \Psi_k$
    end if
end for
return $\Theta_{\text{opt}}, \Lambda_{\max}$

- π_k: Losowa permutacja 26 liter alfabetu łacińskiego, wybrana z rozkładu jednostajnego $\mathcal{A}_{26}$.
- Ψ_k: Rozwiązanie wygenerowane w k - tej iteracji, bazujące na permutacji π_k.
- Λ_k: Wartość funkcji celu dla rozwiązania Ψ_k.
- $\mathcal{A}_{26}$: Zbiór wszystkich możliwych permutacji 26-literowego alfabetu.
- $U(\mathcal{A}_{26})$: Rozkład jednostajny na zbiorze $\mathcal{A}_{26}$, z którego losowana jest permutacja π_k.

Zadanie 4

Dokonać kryptoanalizy heurystycznej na zaimplementowany w ramach pierwszego zadania monoalfabetyczny kryptosystem podstawieniowy. Założenia ataku są takie jak w zadaniach 2 i 3. Do ataku wykorzystać genetyczny algorytm optymalizacji.

Algorytm genetyczny

Algorytm genetyczny (AG) to metoda optymalizacji inspirowana zasadami ewolucji biologicznej. Wykorzystuje pojęcia takie jak selekcja naturalna, krzyżowanie (rekombinacja) oraz mutacja. Jego celem jest znalezienie rozwiązania problemu, które będzie jak najlepsze według określonej funkcji celu (fitness). Algorytmy genetyczne są często stosowane do rozwiązywania problemów trudnych obliczeniowo, takich jak optymalizacja w przestrzeni wielowymiarowej, planowanie, projektowanie, identyfikacja parametryczna modeli matematycznych czy też kryptoanaliza.

Ogólny opis algorytmu

Podstawowe komponenty algorytmu genetycznego:

- **Populacja**: Zbiór możliwych rozwiązań, zwanych osobnikami lub chromosomami. Populacja ewoluuje z pokolenia na pokolenie.
- **Chromosom**: Reprezentacja kandydata na rozwiązanie. Zwykle jest przedstawiany jako ciąg znaków lub cyfr binarnych, na które zamieniane są inne struktury danych, zależnie od charakteru problemu.
- **Funkcja dopasowania** (fitness function): Funkcja, która ocenia jakość każdego osobnika. Określa, jak dobrze dany chromosom rozwiązuje problem.

Operatory genetyczne:

- **Selekcja** lub inaczej **Reprodukcja**: Proces wybierania najlepszych osobników, które będą rodzicami przyszłych generacji.
  - Popularne metody selekcji obejmują selekcję ruletkową, turniejową lub rankingową.
- **Krzyżowanie** (crossover): Proces łączenia dwóch osobników (rodziców) w celu wygenerowania nowych osobników (potomków).

  - Celem jest wymiana genów między rodzicami, co może prowadzić do tworzenia lepszych rozwiązań.
- **Mutacja**: Proces wprowadzania niewielkich zmian w genotypie osobnika, aby zapewnić różnorodność genetyczną w populacji i uniknąć zbieżności do lokalnych minimów.

Parametry algorytmu:

- **Rozmiar populacji**: m - liczba osobników w każdej generacji (w rozpatrywanym przypadku jest to liczba kluczy).
- **Prawdopodobieństwo krzyżowania**: p_c - procent rodziców który ulega krzyżowaniu.
- **Prawdopodobieństwo mutacji**: p_m - prawdopodobieństwo modyfikacji genów (w rozpatrywanym przypadku jest prawdopodobieństwo zmian liter w kluczu).
- **Maksymalne odchylenie standardowe** *funkcji dopasowania*: $\max s_f$ - kryterium zbieżności.
- **Maksymalna liczba generacji**: $i_{\max}$.

Ogólny schemat działania algorytmu genetycznego:

1. **Inicjalizacja**:
   - Algorytm zaczyna od stworzenia początkowej populacji m losowych kluczy szyfrujących.
     - Klucz to permutacja alfabetu, która reprezentuje potencjalne rozwiązanie.
2. **Ewaluacja funkcji dopasowania**:
   - Obliczyć wartość *funkcji dopasowania* korzystając funkcji logarytmicznej wiarygodności dla każdego klucza.
     - Porównując częstości bigramów w tekście zaszyfrowanym po zdekodowaniu przy użyciu klucza z częstościami referencyjnymi.
3. **Selekcja**:
   - Wykorzystać algorytm *ruletkowej selekcji*, w której osobniki są wybierane z populacji z prawdopodobieństwem proporcjonalnym do ich wartości funkcji dopasowania.
     - To oznacza, że lepiej dopasowane osobniki mają większą szansę na wybór.
   - Dla każdego osobnika (klucza) π_i w populacji, prawdopodobieństwo wyboru wynosi:

$$p_i = \frac{S(\pi_i)}{\sum_{j=1}^{m} S(\pi_j)}$$

     gdzie $S(\pi_i)$ jest funkcją wartości dopasowania dla osobnika π_i.

Algorithm 4 Selekcja Ruletkowa

Require: $\mathcal{P} = \{\pi_1, \pi_2, \ldots, \pi_m\}$, $S(\pi_i) \forall \pi_i \in \mathcal{P}$
Ensure: π_{selected}
$F \leftarrow \sum_{i=1}^{m} S(\pi_i)$ ▷ Sumaryczna wartość funkcji dopasowania
$p_i \leftarrow \frac{S(\pi_i)}{F} \quad \forall i \in \{1, 2, \ldots, m\}$ ▷ Prawdopodobieństwa wyboru
$r \sim \mathrm{U}(0, 1)$ ▷ Losowa liczba z rozkładu jednostajnego
$C \leftarrow 0$ ▷ Suma skumulowana
for $i \leftarrow 1$ **to** m **do**
    $C \leftarrow C + p_i$
    if $r \leq C$ **then**
        return π_i ▷ Zwróć wybrany osobnik
    end if
end for

4. **Operatory genetyczne**:

- Dokonać operacji *krzyżowania* (crossover):
  - Dwa rodzice są łączeni w celu stworzenia dwóch potomków.
  - Wykorzystać algorytm *krzyżowania jednopunktowego*, co oznacza, że losowo wybierany jest punkt cięcia, po którym geny (litery) są wymieniane między rodzicami.

    Rodzice: $\pi_1 = [x_1, \ldots, x_k, \ldots, x_{26}]$, $\pi_2 = [y_1, \ldots, y_k, \ldots, y_{26}]$

    Dzieci: $\pi_1 = [x_1, \ldots, x_k, y_{k+1}, \ldots, y_{26}]$, $\pi_2 = [y_1, \ldots, y_k, x_{k+1}, \ldots, x_{26}]$

Algorithm 5 Krzyżowanie

Require: π_1, π_2 ▷ Rodzice
Ensure: $\pi_{\text{child1}}, \pi_{\text{child2}}$ ▷ Potomkowie
$k \sim \mathrm{U}\{1, \ldots, 25\}$ ▷ Losowy punkt cięcia
$\pi_{\text{child1}} \leftarrow [\pi_1[1:k] \cup \pi_2[k+1:26]]$
$\pi_{\text{child2}} \leftarrow [\pi_2[1:k] \cup \pi_1[k+1:26]]$
return $\pi_{\text{child1}}, \pi_{\text{child2}}$

- Dokonać operacji *mutacji*:
  - Z niewielkim prawdopodobieństwem zamienić dwa znaki w kluczu miejscami (geny w chromosomie).
    - Zapewnia to różnorodność genetyczną i zapobiega zbieżności do lokalnych minimów.

$$\pi = [x_1, \ldots, x_a, \ldots, x_b, \ldots, x_{26}] \Rightarrow \pi' = [x_1, \ldots, x_b, \ldots, x_a, \ldots, x_{26}]$$

Algorithm 6 Mutacja

Require: π ▷ Chromosom
Ensure: π' ▷ Zmutowany chromosom
$a, b \sim \mathrm{U}\{1, \ldots, 26\}, \quad a \neq b$ ▷ Losowe indeksy
$\pi' \leftarrow \sigma_{a,b}(\pi)$ ▷ Zamiana miejsc elementów $\pi[a]$ i $\pi[b]$
return π'

5. **Warunki zakończenia**:
   - Algorytm powinien sprawdzać, czy populacja osiągnęła zbieżność, to znaczy czy odchylenie standardowe wartości dopasowania jest mniejsze niż przyjęte *maksymalne odchylenie standardowe.*
   - Jeśli nie osiągnięto zbieżności, to należy kontynuować proces iteracyjny aż osiągnięcia *maksymalnej liczby generacji*
6. **Zwrócenie najlepszego rozwiązania**:
   - Po zakończeniu algorytmu wybierany powinien być klucz, który miał najwyższą wartość funkcji dopasowania.
     - Jest to klucz, który najlepiej odtwarza oryginalny tekst.

Opis algorytmu w kontekście ataku na analizowany kryptosystem

Dane wejściowe:

- c - szyfrogram.
- g - n-gramy referencyjne (bigramy).
- m - rozmiar populacji.
- p_c - prawdopodbieństwo krzyżowania.
- p_m - prawdopodobieństwo mutacji.

- $i_{\max}$ - maksymalna liczba generacji.
- s_f - odchylenie standardowe funkcji dopasowania do wszystkich osobników z populacji $\mathcal{P}$.

Opis użytych symboli:

- π - Pojedynczy chromosom (klucz), permutacja liter alfabetu łacińskiego.
- π_{best} - Najlepsza permutacja (klucz szyfrujący) znaleziona podczas ewolucji.
- $S(\pi)$ - Wartość funkcji dopasowania dla permutacji π.
- $\mathcal{P}$ - Populacja permutacji kluczy szyfrujących.
- $\arg\max$ - Oznacza wybór klucza z największą wartością funkcji dopasowania.
- $s_f^{\max}$ - Maksymalna dopuszczalna wartość odchylenia standardowego.
- $\mathcal{S}_{\text{RW}}(\mathcal{P}, S)$ - Selekcja ruletkowa, która wybiera osobnika π z populacji $\mathcal{P}$ proporcjonalnie do wartości funkcji dopasowania $S(\pi)$.
- $\mathcal{C}$ - Operator krzyżowania.
- $\mathcal{M}$ - Operator mutacji.

Algorithm 7 GA

Require: $c, g, m, p_c, p_m, i_{\max}, s_f^{\max}$
Ensure: π_{best}

$\mathcal{P} \leftarrow \{\pi_i \sim \text{Perm}(\mathcal{A}) \mid i = 1, \ldots, m\}$ ▷ Losowe permutacje alfabetu
$S(\pi) \leftarrow f_{c,g}(\pi) \quad \forall \pi \in \mathcal{P}$
$\pi_{\text{best}} \leftarrow \arg\max_{\pi \in \mathcal{P}} S(\pi)$
for $i \leftarrow 1$ **to** $i_{\max}$ **do**
  $s_f \leftarrow \sqrt{\frac{1}{m} \sum_{\pi \in \mathcal{P}} \left(S(\pi) - \frac{1}{m} \sum_{\pi \in \mathcal{P}} S(\pi)\right)^2}$ ▷ Oblicz odchylenie standardowe
  if $s_f \leq s_f^{\max}$ **then**
    break ▷ Zakończ, jeśli populacja jest zbieżna
  end if
  $\mathcal{P}' \leftarrow \emptyset$
  for $j \leftarrow 1$ **to** $\frac{m}{2}$ **do**
    $\pi_1 \leftarrow \mathcal{S}_{\text{RW}}(\mathcal{P}, S)$ ▷ Selekcja ruletkowa
    $\pi_2 \leftarrow \mathcal{S}_{\text{RW}}(\mathcal{P} \setminus \{\pi_1\}, S)$ ▷ Selekcja ruletkowa bez π_1
    if $u_1 \sim \text{U}(0,1) < p_c$ **then**
      $(\pi_{\text{child1}}, \pi_{\text{child2}}) \leftarrow \mathcal{C}(\pi_1, \pi_2)$ ▷ Krzyżowanie
    else
      $(\pi_{\text{child1}}, \pi_{\text{child2}}) \leftarrow (\pi_1, \pi_2)$
    end if
    $\mathcal{P}' \leftarrow \mathcal{P}' \cup \{\pi_{\text{child1}}, \pi_{\text{child2}}\}$
  end for
  for all $\pi \in \mathcal{P}'$ **do**
    if $u_2 \sim \text{U}(0,1) < p_m$ **then**
      $\pi \leftarrow \mathcal{M}(\pi)$ ▷ Mutacja
    end if
  end for
  $S(\pi) \leftarrow f_{c,g}(\pi) \quad \forall \pi \in \mathcal{P}'$
  $\pi_{\text{best}} \leftarrow \arg\max\left(S(\pi_{\text{best}}), \max_{\pi \in \mathcal{P}'} S(\pi)\right)$
  $\mathcal{P} \leftarrow \mathcal{P}'$
end for
return π_{best}

Zadanie 5

Dokonać analizy pracy zaimplementowanych algorytmów, porównując ich wydajność w ataku na analizowany kryptosystem.