Algorytmy i Struktury Danych
Wykład 4 - Wybrane struktury danych

mgr inż. Andrii Shekhovtsov

Uniwersytet WSB Merito

30 listopada 2023

Spis treści

Struktury danych

Czym są struktury danych?

Struktura danych - sposób przechowywania danych w pamięci komputera. Na strukturach danych operują algorytmy, umożliwiające dodawanie danych do struktury, lub ich usunięcie, a także przeszukiwanie struktury w celu znalezienia danych.

Gdzie szukać różnych struktur danych?

Języki programowania często posiadają rozbudowaną bibliotekę standardową, umożliwiającą używanie różnych struktur danych. Przykładowo, w Python mamy takie typy danych jak listy, krotki, słowniki, zbiory, i inne.

Dlaczego to jest potrzebne?

Wybór struktury danych do zadania

Programista powinien być świadom tego jakich struktur danych używa i jaką złożoność obliczeniową posiadają algorytmy na nich operujące. Programista ma być w stanie odpowiedzieć na takie pytania jak:

- Którą strukturę danych należy użyć gdy chcemy często szukać w niej wartości?
- Którą strukturę danych użyć gdy chcemy szybko wykonywać dodawanie i usuwanie z jej końców?
- Która struktura danych będzie zużywała mniej pamięci?
- i inne...

Przykładowe struktury danych

Istnieje dużo różnych struktur danych:

- Tablica,
- Stos,
- Kolejka,
- Lista powiązana,
- Tablica mieszająca,
- Drzewo binarne,
- Kopiec binarny,
- i inne.

Każda jest zdefiniowana inaczej, a działania na nich różną złożoność obliczeniową.

Przykładowe struktury danych - c.d.

Data Structure	Time Complexity								Space Complexity
	Average				Worst				Worst
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion	
Array	Θ(1)	Θ(n)	Θ(n)	Θ(n)	O(1)	O(n)	O(n)	O(n)	O(n)
Stack	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Queue	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Singly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Doubly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Skip List	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n log(n))
Hash Table	N/A	Θ(1)	Θ(1)	Θ(1)	N/A	O(n)	O(n)	O(n)	O(n)
Binary Search Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)
Cartesian Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(n)	O(n)	O(n)	O(n)
B-Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Red-Black Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Splay Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(log(n))	O(log(n))	O(log(n))	O(n)
AVL Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
KD Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)

Źródło: `https://www.bigocheatsheet.com/`

Stos

Definicja

Stos (ang. Stack) - liniowa struktura danych, działająca na zasadzie LIFO (Last In, First Out). Kolejne wartości są dokładane na górę stosu, i zdejmowane także z góry stosu.

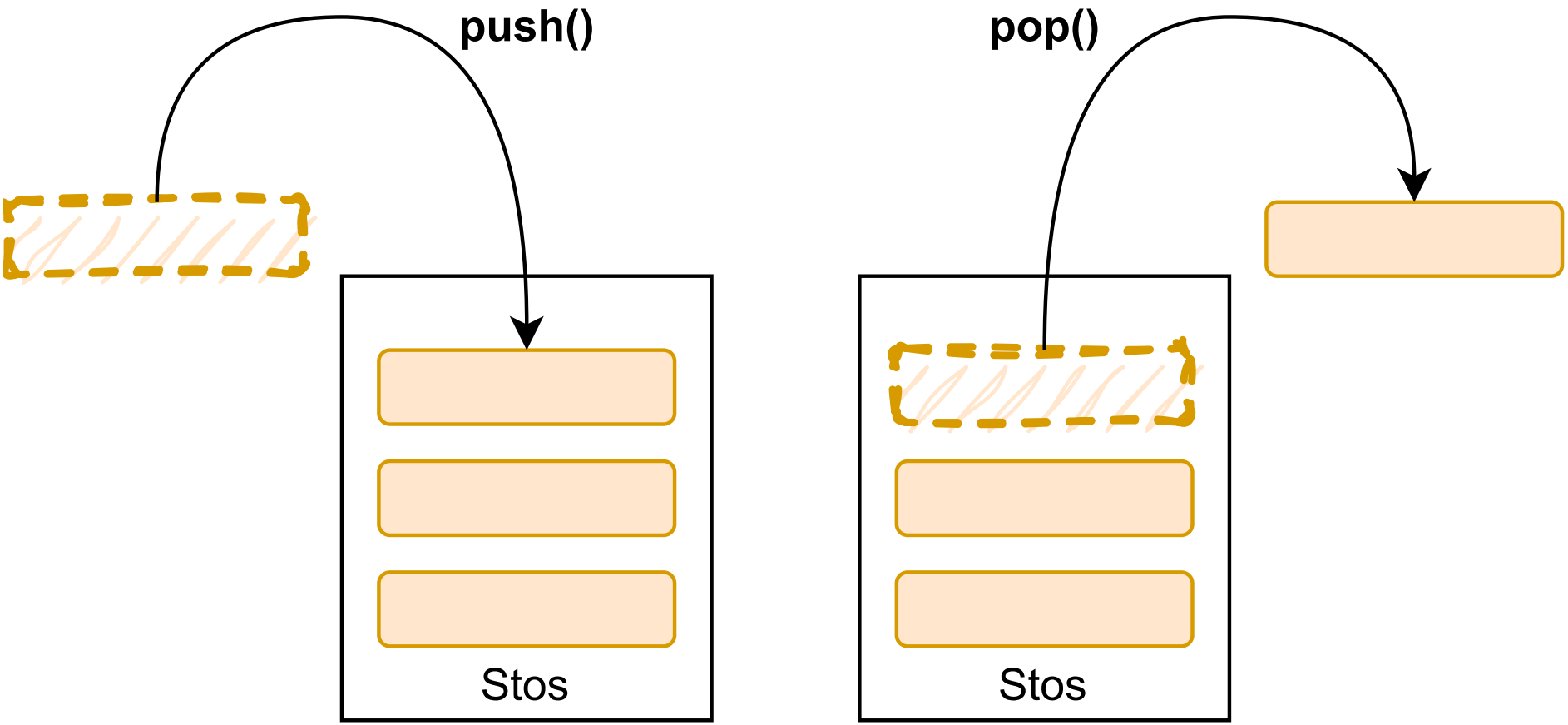
Podstawowe operacje

- push(object) - dodaj obiekt na górę stosu,
- pop(object) - usuń element z góry stosu i zwróć jego wartość,
- len() - sprawdzenie ilości elementów na stosie.

Wszystkie te operacje mają stałą złożoność obliczeniową O(1).

Stos - Operacje

Stos - Operacje



Przykład użycia stosu w Python

```
>>> from collections import deque
>>> stos = deque()
>>> stos.append(42)
>>> stos
deque([42])
>>> stos.append(3)
>>> stos
deque([42, 3])
>>> stos.pop()
3
>>> stos
deque([42])
```

Sprawdzenie zgodności nawiasów

Funkcja sprawdź($napis$)

- Utwórz pusty stos S
- **Dla** każdego symbolu sym w $napis$ powtarzaj:
  - **Jeżeli** sym to "("
    - Dodaj sym do S
  - **Inaczej jeżeli** sym to ")"
    - Jeżeli stos jest pusty **zwróć** $False$
    - Zdejmij symbol $sym2$ ze stosu S
    - **Jeżeli** $sym2$ nie jest "(" **zwróć** $False$

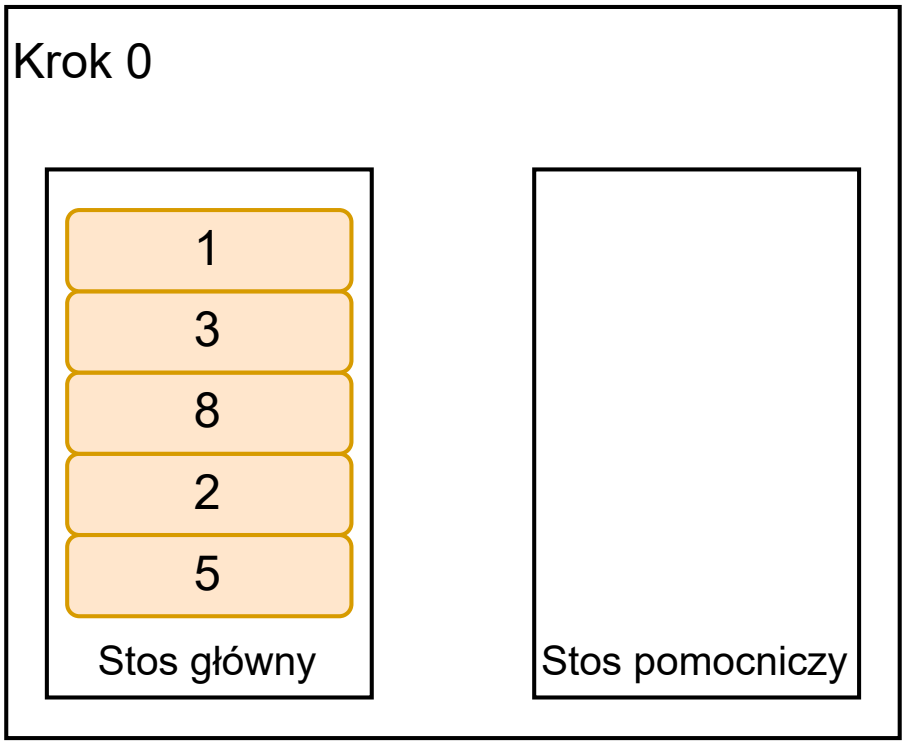
Zwróć $True$ jeżeli stos jest pusty lub $False$ inaczej

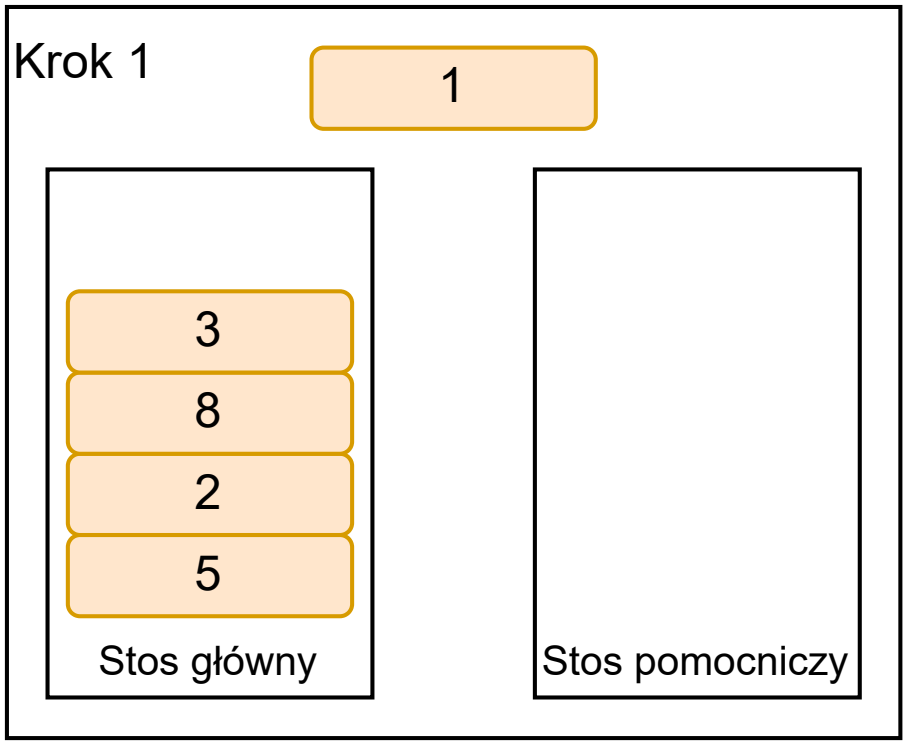
Obliczenie wyrażenia w Odwrotnej Notacji Polskiej (ONP)

Funkcja oblicz($napis$)

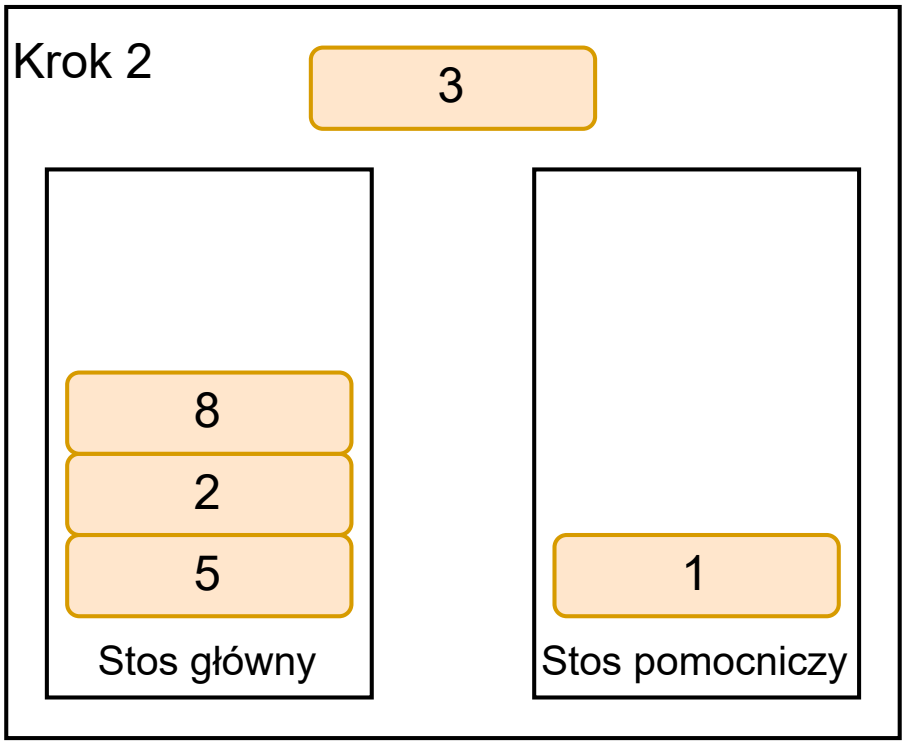
- Utwórz pusty stos S
- **Dla** każdego symbolu sym w $napis$ powtarzaj:
  - **Jeżeli** sym jest liczbą
    - Dodaj sym do S
  - **Inaczej**
    - Pobierz z S wartość do zmiennej a
    - Pobierz z S wartość do zmiennej b
    - Zapisz wynik działania $b\ sym\ a$ do S
- **Zwróć** wartość pobraną z S

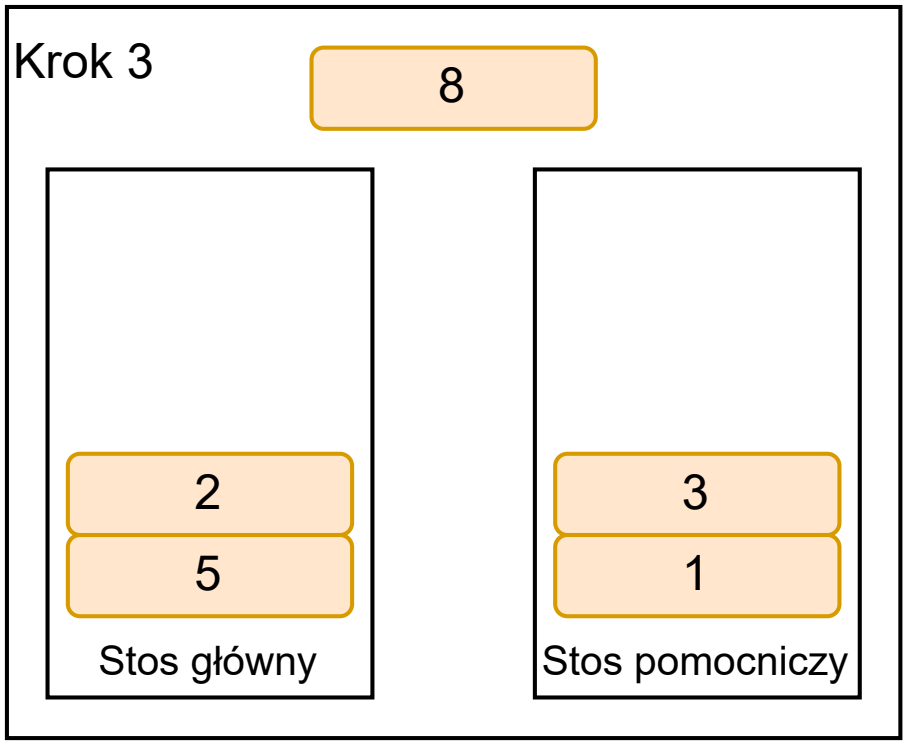
Sortowanie stosu za pomocą stosu pomocniczego



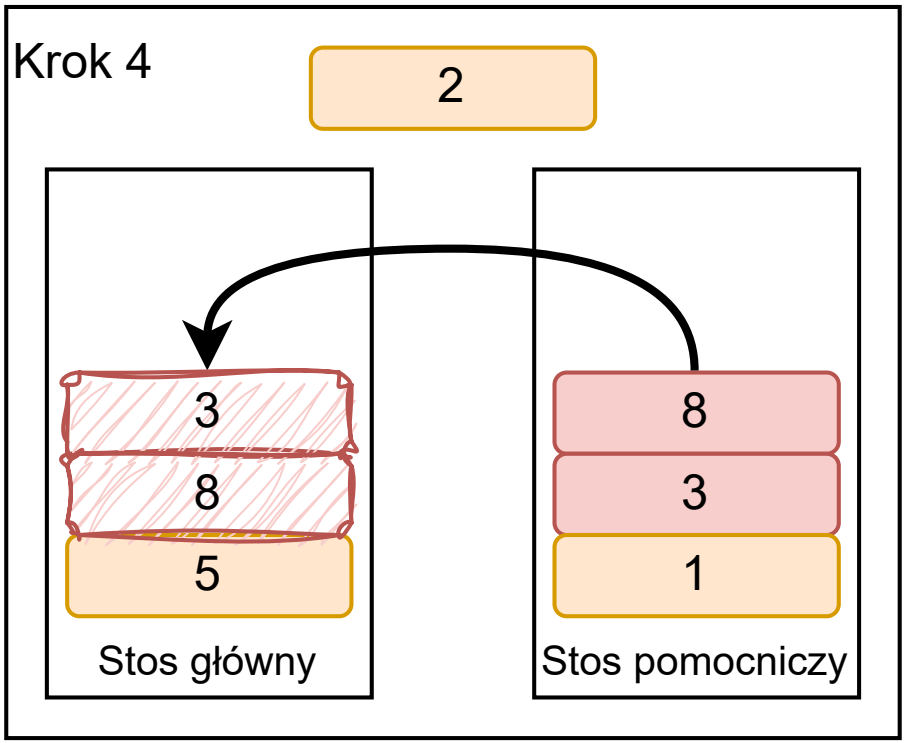


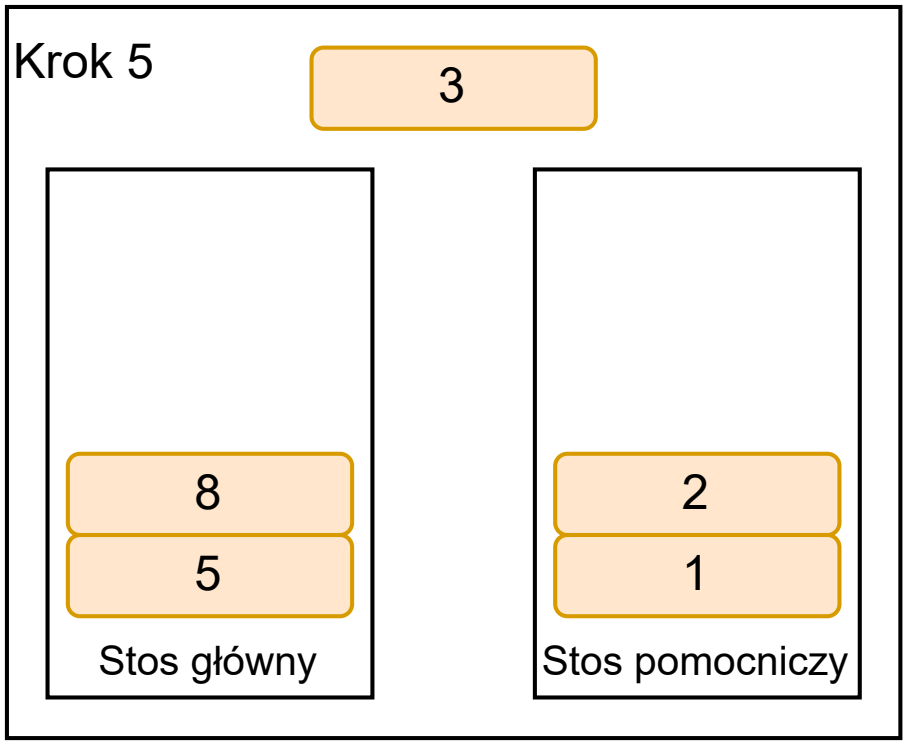
Sortowanie stosu za pomocą stosu pomocniczego - c.d.



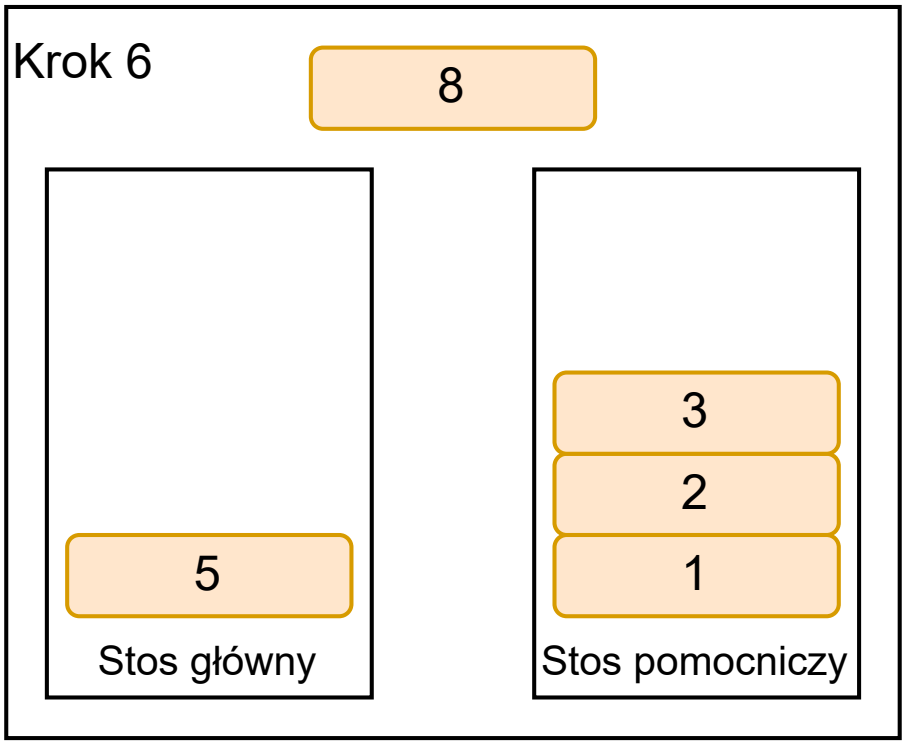


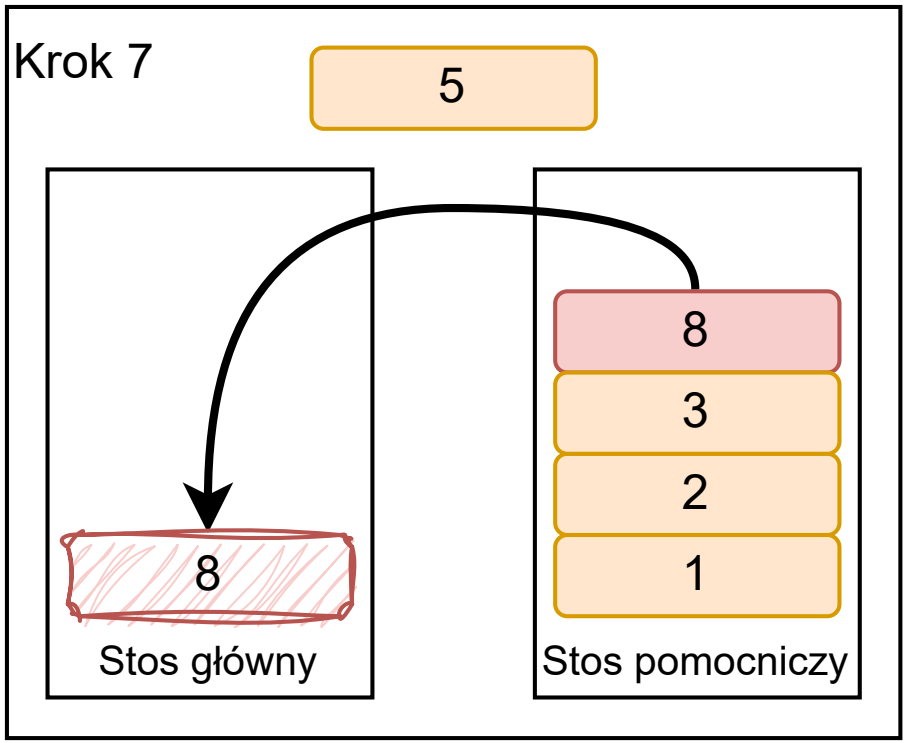
Sortowanie stosu za pomocą stosu pomocniczego - c.d.



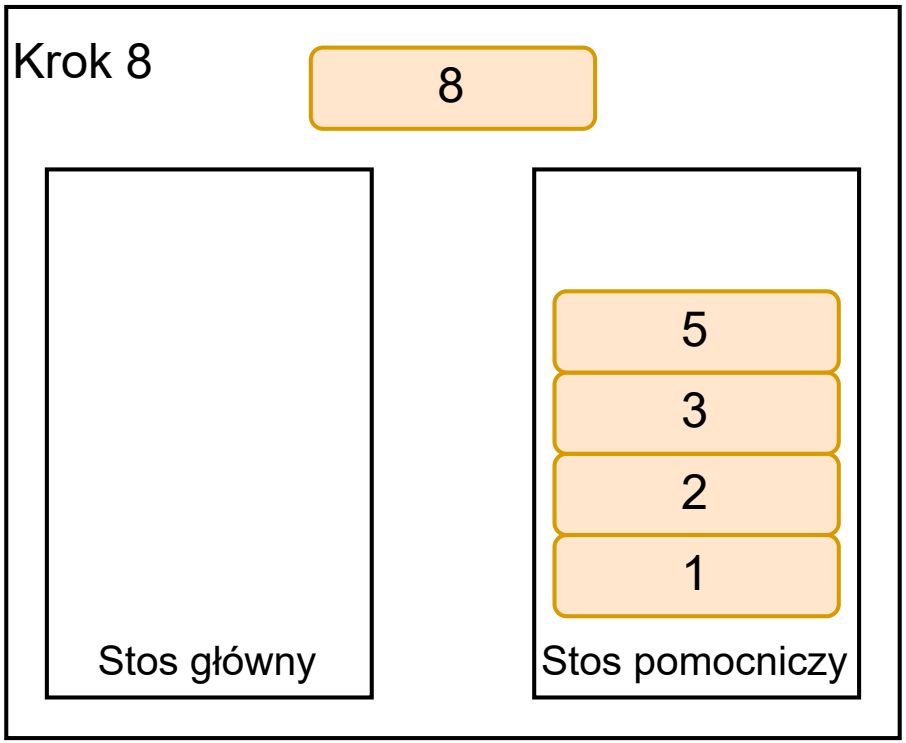


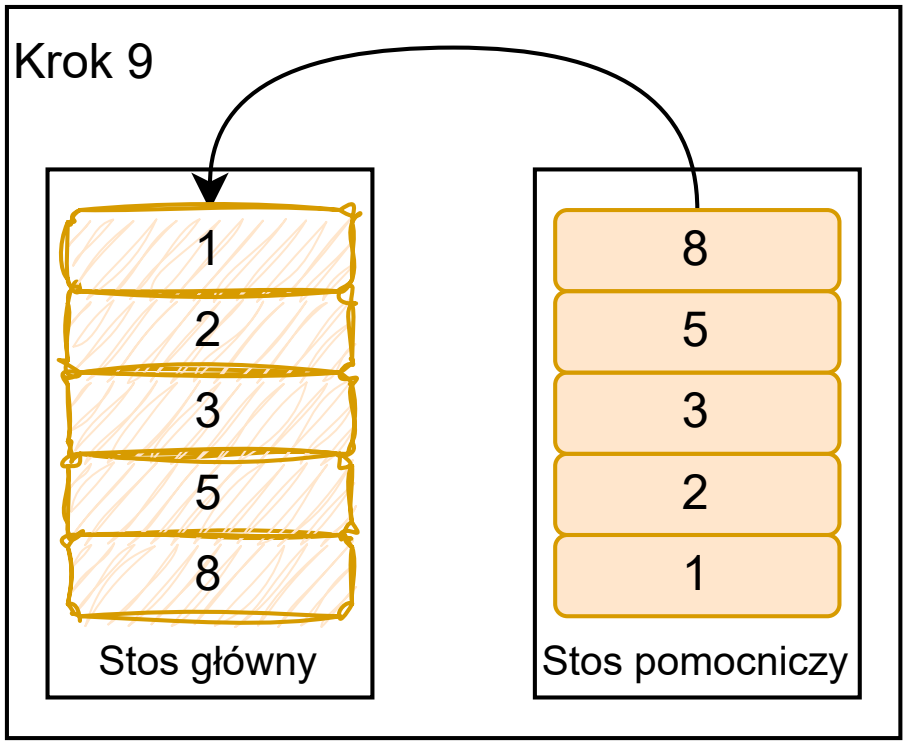
Sortowanie stosu za pomocą stosu pomocniczego - c.d.





Sortowanie stosu za pomocą stosu pomocniczego - c.d.





Kolejka

Definicja

Kolejka (ang. queue) - liniowa struktura danych, działająca na zasadzie FIFO (First In, First Out). Kolejne wartości są dodawane na koniec kolejki, a pobierane z jej początku.

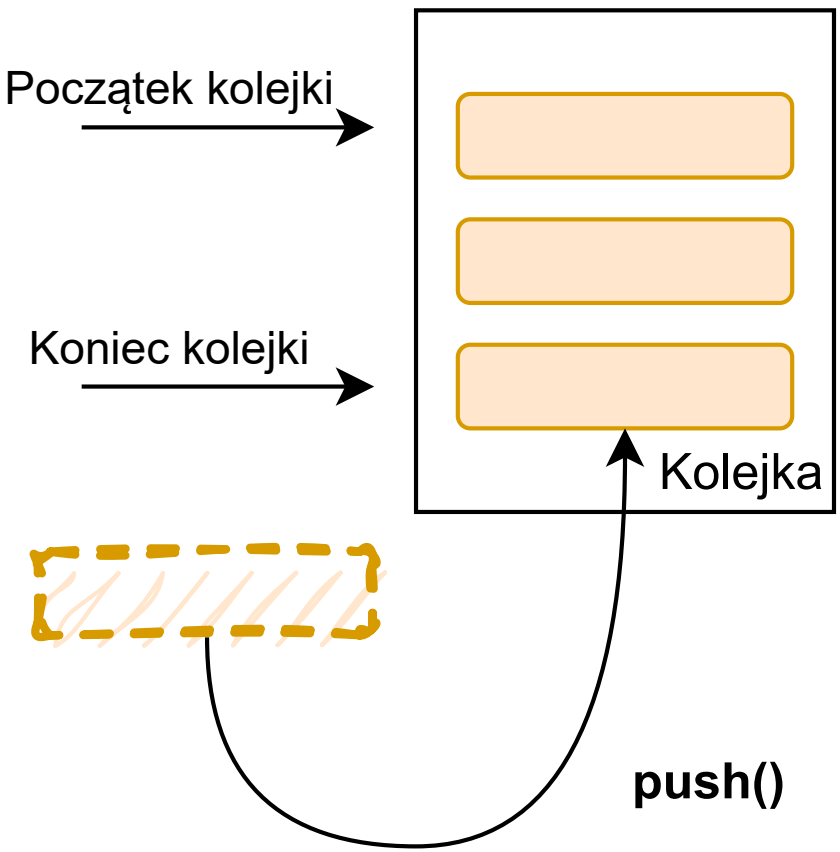
Podstawowe operacje

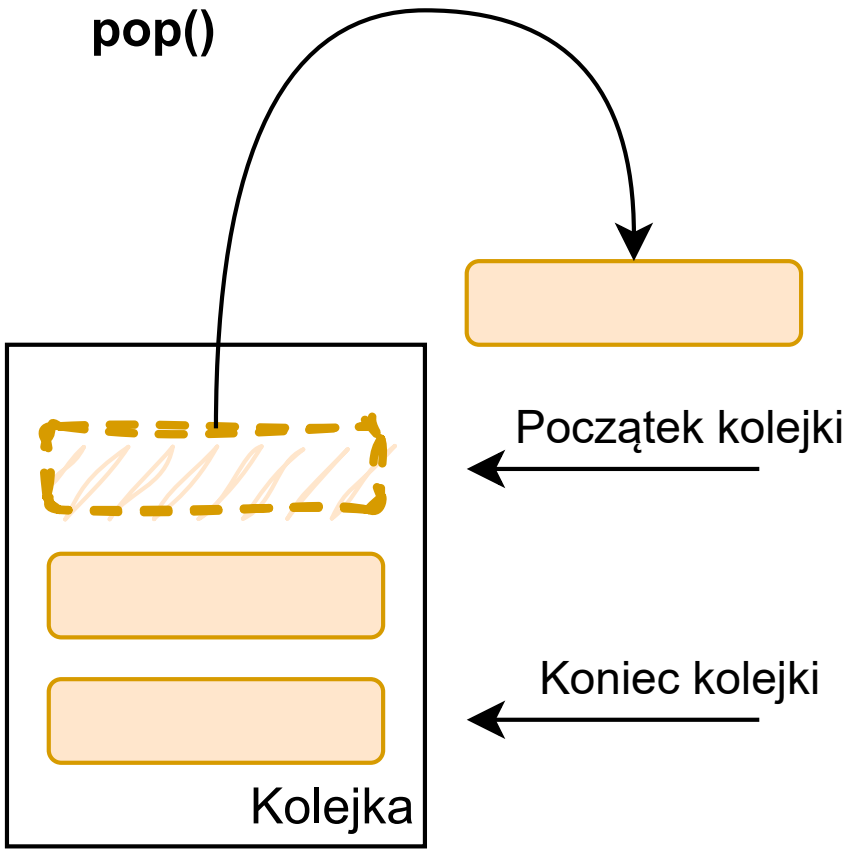
- push(object) - dodaj obiekt na koniec kolejki,
- pop(object) - usuń element z początku kolejki i zwróć go,
- len() - sprawdzenie ilości elementów w kolejce.

Wszystkie te operacje mają stałą złożoność obliczeniową O(1).

Kolejka - Operacje

Kolejka - Operacje





Przykład użycia kolejki w Python

```
>>> from collections import deque
>>> kolejka = deque()
>>> kolejka.append(42)
>>> kolejka
deque([42])
>>> kolejka.append(3)
>>> kolejka
deque([42, 3])
>>> kolejka.popleft()
42
>>> kolejka
deque([3])
```

Stos vs Kolejka - podsumowanie

Stos:

- Dodajemy elementy na koniec
- Pobieramy elementy z końca
- LIFO - Last In, First Out
- „ostatni na wejściu, pierwszy na wyjściu”

Kolejka:

- Dodajemy elementy na koniec
- Pobieramy elementy z początku
- FIFO - First In, First Out
- „pierwszy na wejściu, pierwszy na wyjściu”

Tablica mieszająca

Definicja

Tablica mieszająca (ang. hash table) - struktura danych, która jest jednym ze sposobów realizacji tablicy asocjacyjnej, czyli abstrakcyjnego typu który powiązuje klucze z wartościami. Odwołanie do przechowywanych obiektów dokonuje się na podstawie klucza, a mechanizmy tablicy mieszającej pozwalają na uzyskanie szybkiego dostępu do tych obiektów.

Czy Python ma w sobie implementacje tablicy mieszającej?

Tablica mieszająca - c.d.

Podstawowe operacje

- insert(key, value) - dodaj parę key-value do tablicy mieszającej,
- get(key) - zwróć value odpowiadające kluczowi key, lub zwróć None,
- remove(key) - wymaż parę key-value z tablicy mieszającej,
- len() - sprawdzenie ilości elementów w tablice mieszającej.

W przypadku średnim złożoność obliczeniowa tych operacji to O(1).
Jednak, w przypadku najgorszym złożoność tych operacji wynosi O(N).

Funkcja skrótu (hash function)

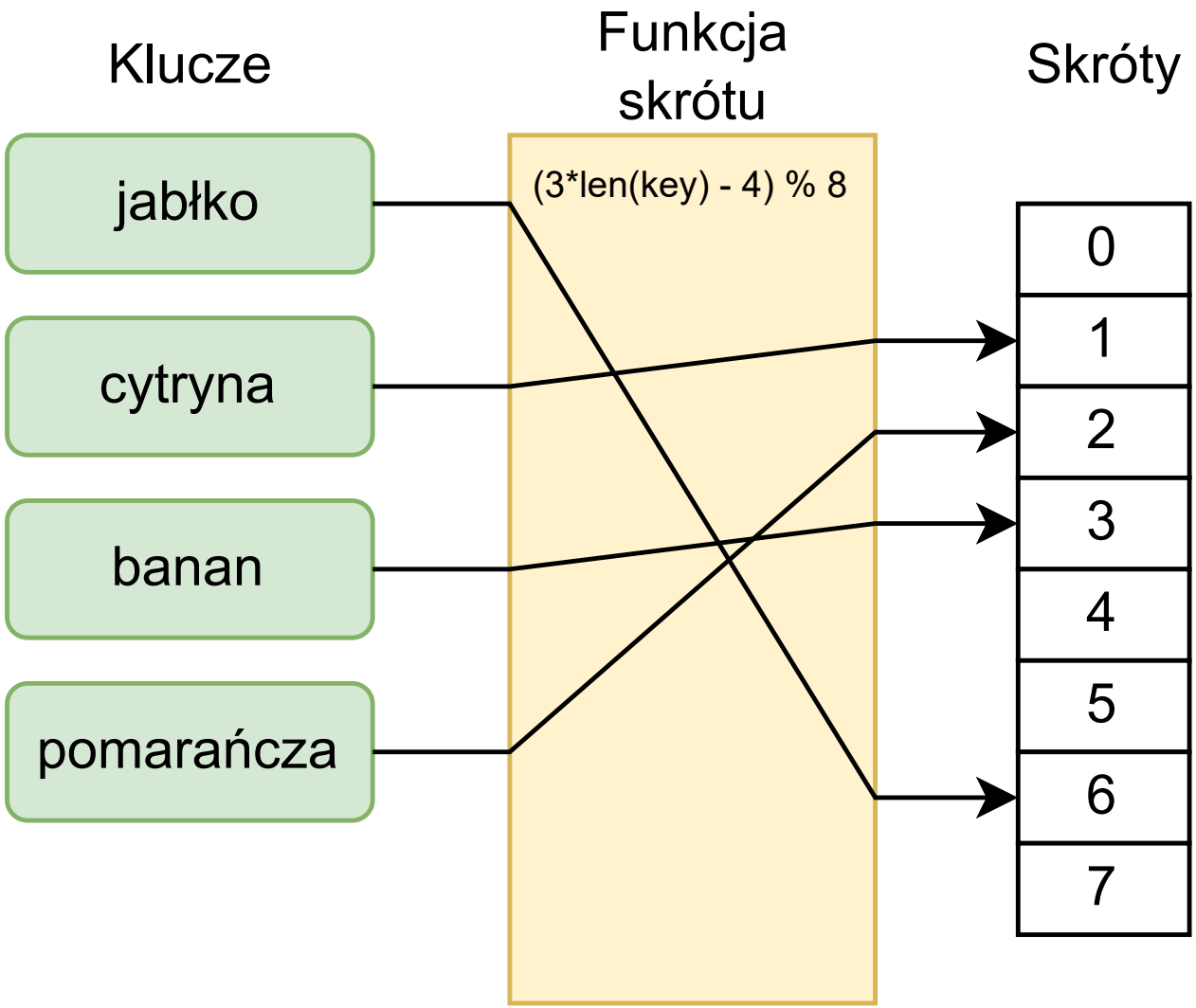
Definicja

Funkcja skrótu, **funkcja mieszająca** lub **funkcja haszująca** – funkcja przyporządkowująca dowolnie dużej ilości danych krótką wartość o stałym rozmiarze, tzw. skrót nieodwracalny.

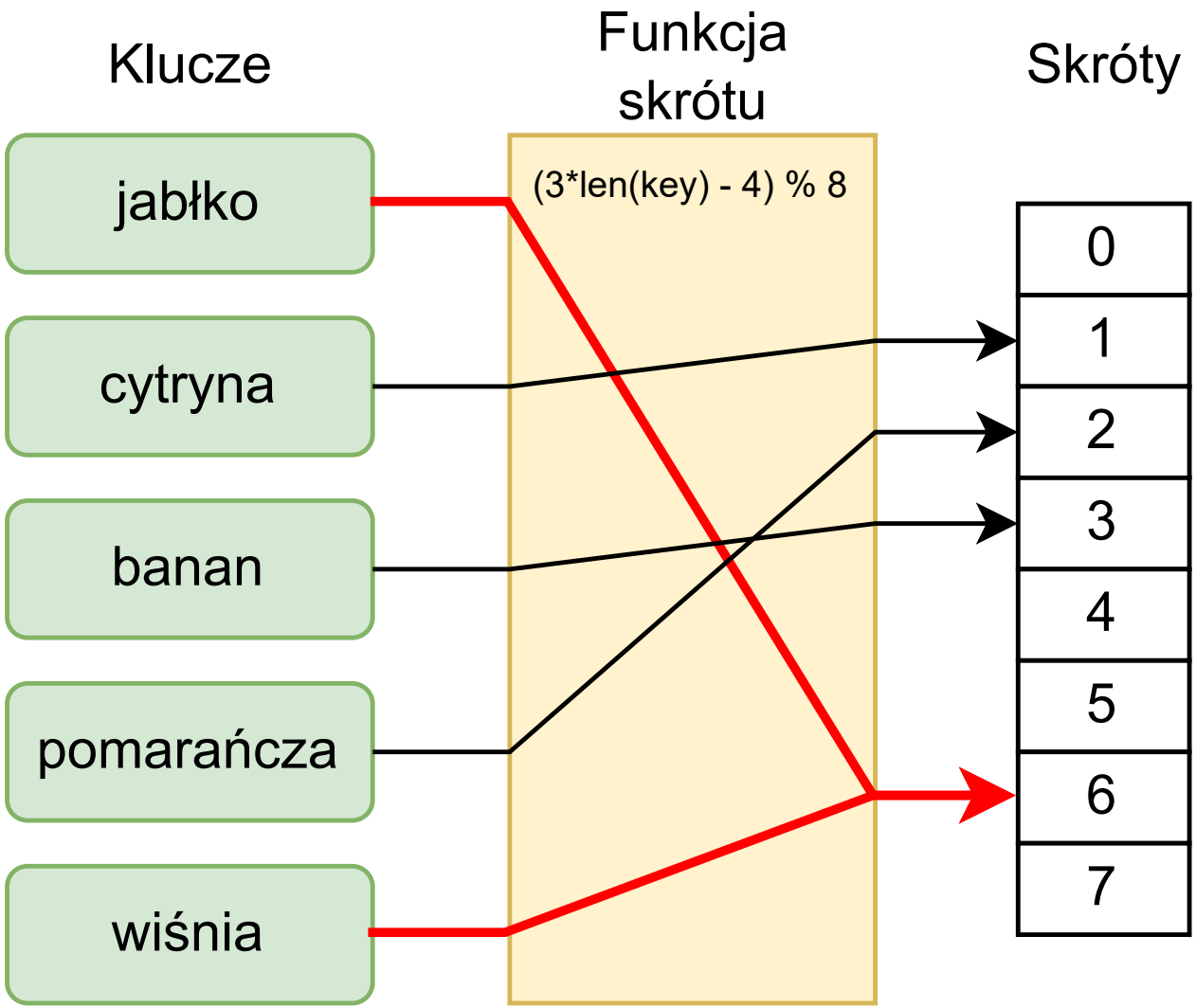
Własności dobrej funkcji skrótu

- Możliwość szybkiego obliczenia skrótu
- Nieodwracalność
- Brak kolizji

Funkcja skrótu - c.d.



Funkcja skrótu - c.d.



Zjawisko kolizji

Definicja formalna

Kolizją funkcji skrótu $H(x)$ nazywamy parę **różnych** argumentów x_1 i x_2, takich że $H(x_1) = H(x_2)$.

Dlaczego jest to zjawisko niepożądane?

- W kontekście kryptografii - jeżeli znajdziemy kolizje dla konkretnego hasła, to możemy zalogować się nie znając tego hasła.
- W kontekście tablicy mieszającej - kolizje uniemożliwiają wstawienie elementów we właściwe miejsce, co spowalnia kolejne wstawienia i wyszukiwania.

Różne sposoby rozwiązywania kolizji

- Metoda łańcuchowa

  Nie przechowujemy elementów bezpośrednio w tablice, a w skojarzonych z każdym indeksem listach (O(N)) lub drzewach (O(log N)).

- Adresowanie otwarte Definiujemy funkcję przyrostu $p(i)$, gdzie i oznacza numer próby wstawienia.
  - Szukanie liniowe $p(i) = i$
  - Szukanie kwadratowe $p(i) = i^2$
  - Mieszanie podwójne $p(i) = i \cdot h'(K)$, gdzie h' jest dodatkową funkcją skrótu, a K kluczem.

- Definiowanie współczynnika wypełnienia i dodanie „zakładki".

  Zdefiniujemy współczynnik wypełnienia $\alpha = m/n$ jako iloraz liczby zapisanych elementów m do fizycznego rozmiaru tablicy n. W przypadku gdy α przekroczy ustalony próg, należy zwiększyć rozmiar tablicy n i odpowiednio przeliczyć wszystkie skróty (hashe).

Wstawianie do tablicy mieszającej (szukanie liniowe)

Funkcja wstaw(T, key, $value$)

- Oblicz idx jako $idx = H(key)\ mod\ len(T)$
- Jeżeli $T[idx]$ jest puste lub $T[idx][0] = key$
  - Zapisz parę $(key,\ value)$ pod indeksem idx
  - Zakończ działanie funkcji
- *Komentarz:* Rozwiązywanie kolizji
- Dla wartości idx w przedziale $[0,\ len(T) - 1]$ wykonaj
  - Jeżeli $T[idx]$ jest puste lub $T[idx][0] = key$
    - Zapisz parę $(key,\ value)$ pod indeksem idx
    - Zakończ działanie funkcji
- Wyświetl komunikat o błędzie.
- Zakończ działanie funkcji

Wyszukiwanie w tablice mieszającej (szukanie liniowe)

Funkcja pobierz(T, key)

- Oblicz idx jako $idx = H(key)\ mod\ len(T)$
- Jeżeli $T[idx]$ nie jest puste i $T[idx][0] = key$
  - Zakończ działanie funkcji i **zwróć** $T[idx][1]$, będące $value$ przypisanym do klucza key
- *Komentarz:* Rozwiązywanie kolizji
- Dla wartości idx w przedziale $[0,\ len(T) - 1]$ wykonaj
  - Jeżeli $T[idx]$ nie jest puste i $T[idx][0] = key$
    - Zakończ działanie funkcji i **zwróć** $T[idx][1]$, będące $value$ przypisanym do klucza key
- Wyświetl komunikat o błędzie.
- Zakończ działanie funkcji

Przykład użycia tablicy mieszającej (słownika) w Python

```
>>> ht = {}
>>> ht['apple'] = 50
>>> ht['lemon'] = 20
>>> ht.get('apple')
50
>>> ht.get('banana')
>>> ht.get('banana', 'Brak wartości')
'Brak wartości'
>>> ht.pop('apple')
50
>>> len(ht)
1
>>>
```

Graf - definicja

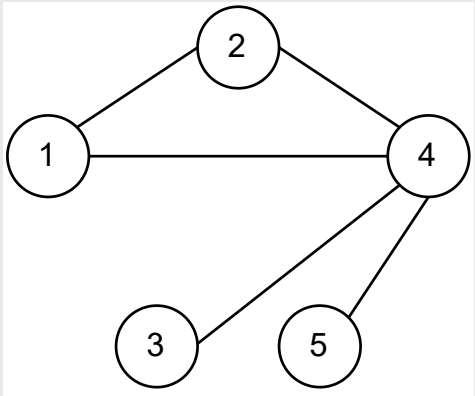
Definicja

Graf - podstawowy obiekt rozważań teorii grafów, struktura matematyczna służąca do przedstawiania i badania relacji między obiektami. W uproszczeniu graf to zbiór **wierzchołków**, które mogą być połączone **krawędziami** w taki sposób, że każda krawędź kończy się i zaczyna w którymś z wierzchołków.

Rodzaje grafów

Rozróżniamy grafy:

- Skierowane i nieskierowane,
- Ważone i bez wagowe,
- Grafy acykliczne i cykliczne,
- Drzewa

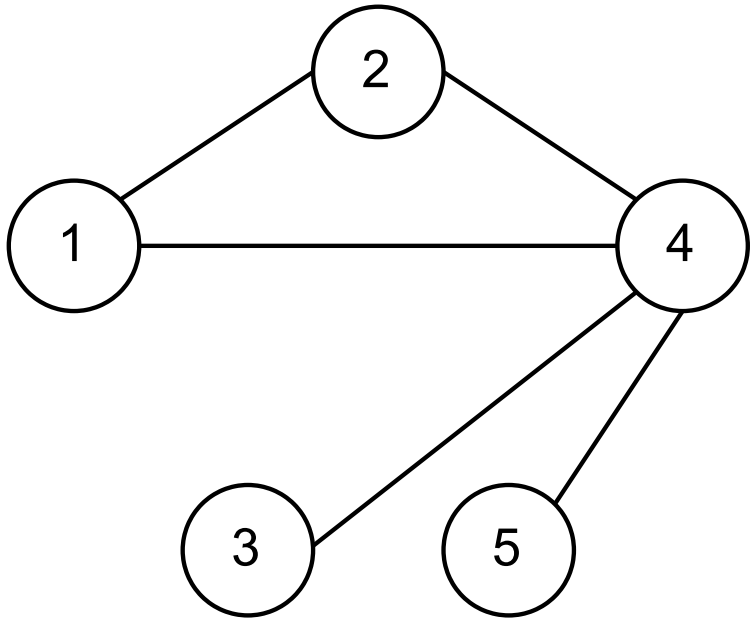


Typowe zadania wykonywane na grafach

- Przeszukiwanie grafu,
  - DFS (Depth-First Search) - przeszukiwanie wgłąb,
  - BFS (Breadth-First Search) - przeszukiwanie wszerz.
- Szukanie minimalnego drzewa spinającego,
  - Algorytm Kruskala.
- Szukanie najkrótszej ścieżki w grafie ważonym,
  - Algorytm Dijkstry,
- Detekcja cykli w grafie.

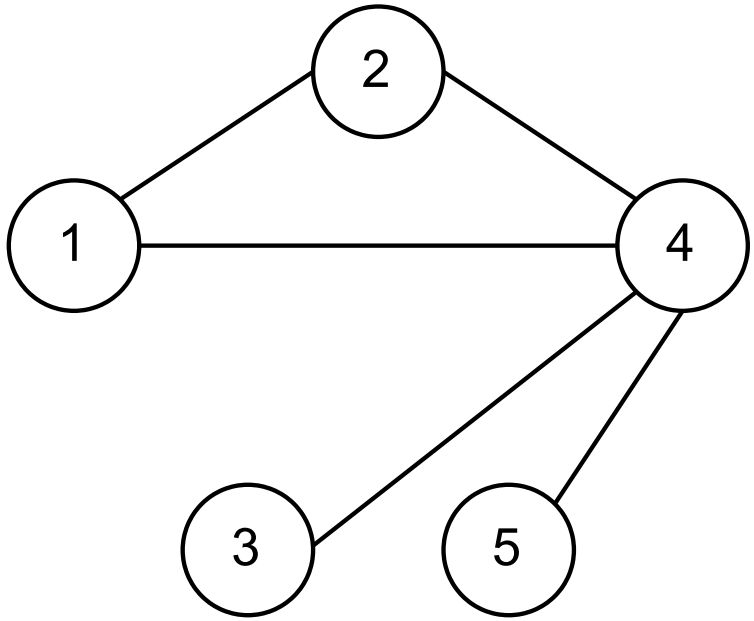
Reprezentacja grafów - Lista sąsiedztwa

$$
\begin{aligned}
1 &\rightarrow 1, 2, 4 \\
2 &\rightarrow 1, 4 \\
3 &\rightarrow 4 \\
4 &\rightarrow 1, 2, 3, 5 \\
5 &\rightarrow 4
\end{aligned}
$$



Reprezentacja grafów - Lista sąsiedztwa (Python)

```
graph = {
    1: [1, 2, 4],
    2: [1, 4],
    3: [4],
    4: [1, 2, 3, 5],
    5: [4],
}
```



Reprezentacja grafów - Graf ważony (Python)

```
1 TODO - zrobić z literami
      zamist cyfr
```

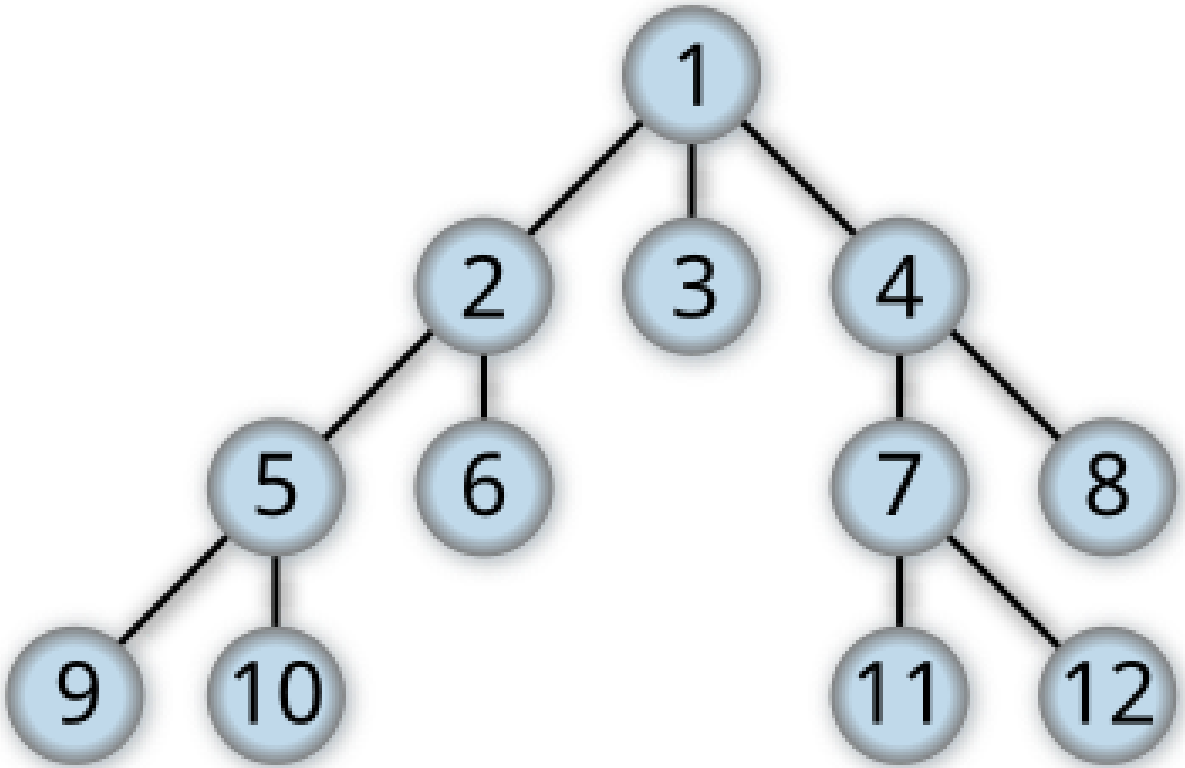
TODO

Przeszukiwanie wszerz - Opis algorytmu

Przeszukiwanie wszerz (ang. Breadth-first search, BFS) - jeden z najprostszych algorytmów przeszukiwania grafu. Przechodzenie grafu rozpoczyna się od zadanego wierzchołka s i polega na odwiedzeniu wszystkich osiągalnych z niego wierzchołków.

- Złożoność obliczeniowa $O(V + E)$, gdzie V to liczba wierzchołków, a E to liczba krawędzi.
- Można użyć do znalezienie najkrótszej ścieżki w grafie nieskierowanym.

Przeszukiwanie wszerz - c.d.



Źródło:
https://pl.wikipedia.org/wiki/Plik:Breadth-first-tree.svg

Przeszukiwanie wszerz - Pseudokod

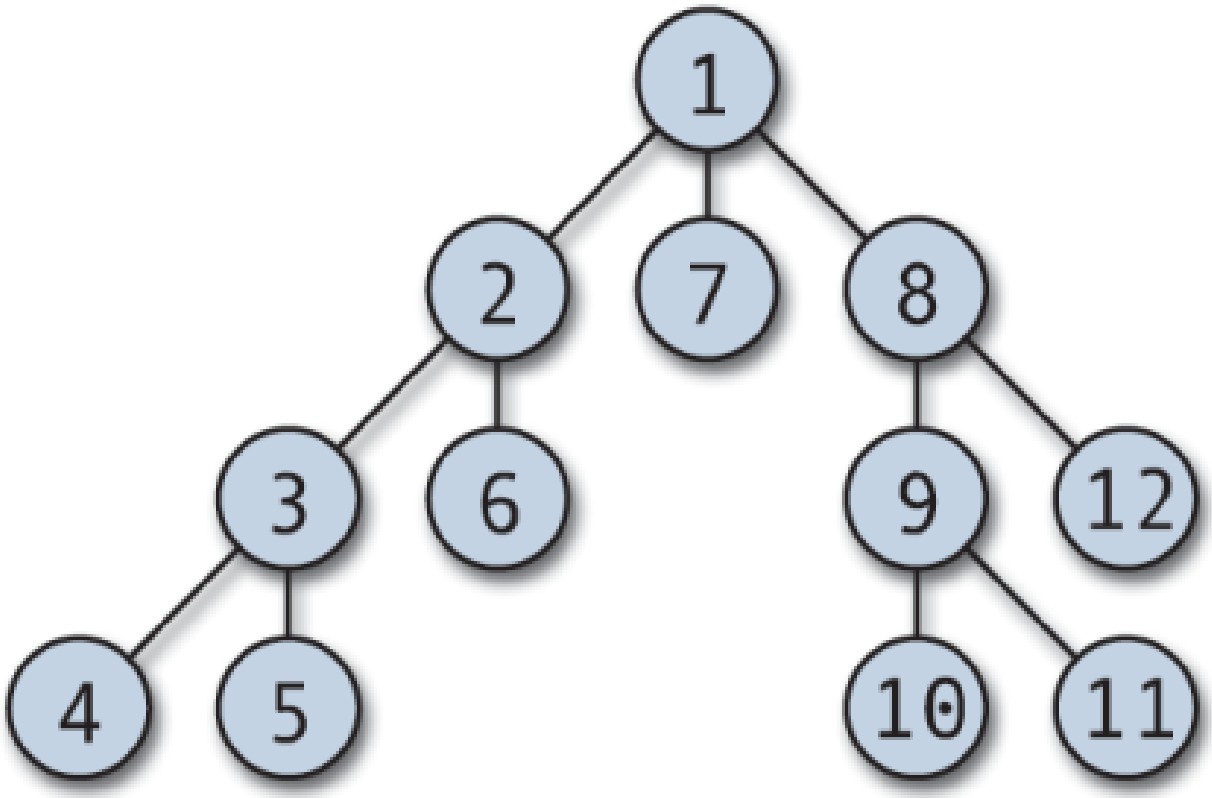
TODO

Przeszukiwanie w głąb - Opis algorytmu

Przeszukiwanie w głąb (ang. Depth-first search) - algorytm przeszukiwania grafu. Przeszukiwanie w głąb polega na badaniu wszystkich krawędzi wychodzących z podanego wierzchołka. Po zbadaniu wszystkich krawędzi wychodzących z danego wierzchołka algorytm powraca do wierzchołka, z którego dany wierzchołek został odwiedzony

- Złożoność obliczeniowa $O(V + E)$, gdzie V to liczba wierzchołków, a E to liczba krawędzi.
- Do sprawdzania, czy istnieje ścieżka między dwoma wierzchołkami w grafie.

Przeszukiwanie w głąb - c.d.



Źródło:
https://pl.wikipedia.org/wiki/Plik:Depth-first-tree.png

Przeszukiwanie w głąb - Pseudokod

TODO

Algorytm Dijkstry - Opis algorytmu

Algorytm Dijkstry - służy do znajdowania najkrótszej ścieżki z pojedynczego źródła w **grafie ważonym** o nieujemnych wagach krawędzi.

- Jest to przykład algorytmu zachłannego.
- Wymyślony został przez holenderskiego informatyka Edsgera Dijkstrę (1930 - 2002).
- Złożoność obliczeniowa zależy od implementacji kolejki priorytetowej (tablica - $O(V^2)$, kopiec - $O(E \cdot \log V)$).
- Jest używany do znalezienia najkrótszej ścieżki pomiędzy dwoma zadanymi wierzchołkami.

Algorytm Dijkstry - Pseudokod

TODO

Dziękuję za uwagę!

Życzę miłego dnia :)