Zapis liczby w zadanej podstawie

Chcemy zapisać liczbę całkowitą $n > 0$ w podstawie $b \geqslant 2$. Dzielenie z resztą daje:

$$n = bq + r, \qquad 0 \leqslant r < b.$$

Reszta r jest ostatnią cyfrą, a iloraz q – liczbą pozostałą po usunięciu tej cyfry.

1. Przygotuj pusty stos.
2. Dziel liczbę przez b: odkładaj reszty na stos, a dalsze dzielenie wykonuj dla ilorazu całkowitego.
3. Gdy iloraz osiągnie zero, zdejmuj cyfry ze stosu.

Stos odwraca kolejność: otrzymujemy cyfry od najważniejszej.

Dla $n = 0$ wynikiem jest po prostu cyfra 0.

Zmiana podstawy – przykład

Zapisujemy $19_{(10)}$ w systemie dwójkowym.

dzielona liczba	iloraz całkowity przez 2	reszta na stos
19	9	1
9	4	1
4	2	0
2	1	0
1	0	1

Reszty odkładamy kolejno: $1, 1, 0, 0, 1$.
Zdejmujemy ze stosu: $1, 0, 0, 1, 1$, zatem $19_{(10)} = 10011_{(2)}$.

Dla innych podstaw zmieniamy dzielnik; dla podstaw powyżej 10 potrzebujemy dodatkowych symboli cyfr.

Przykładowa implementacja: change_base.py, stack.py.
Skrypt: wykład 2 (cz. 2), zapis liczby w zadanej podstawie.

Kolejka

Kolejka (*queue*) – struktura, w której obiekty stoją jeden za drugim.

Podstawowe operacje:

- $\mathrm{enqueue(x)}$ – dokłada x na koniec kolejki.
- $\mathrm{dequeue()}$ – usuwa i zwraca element z początku kolejki (o ile nie jest pusta).

Tak zwana struktura **FIFO** (*first in, first out*): pierwszy włożony element jest wyciągany jako pierwszy.
Operacje dla wygody: sprawdzanie rozmiaru i niepustości kolejki.

Przykładowa implementacja: $\mathrm{my_queue.py}$.
W tej implementacji opartej na liście Pythona dokładanie elementu ma złożoność $\Theta(n)$, a pobieranie $\Theta(1)$, gdzie n to długość kolejki.

(Skrypt: wykład 3 (cz. 2), kolejka.)

Przykład: myjnia samochodowa

Jedno stanowisko obsługuje samochody w kolejności przyjazdu.

- Symulacja trwa n minut.
- W każdej minucie z prawdopodobieństwem r przyjeżdża jeden samochód.
- Czas mycia losujemy równomiernie spośród $1, 2, 3, 4, 5$ minut.
- Samochód pamięta chwilę przyjazdu, myjnia – aktualnie myty samochód.

Opis klas: car.py, carwash.py.

Myjnia samochodowa – krok symulacji

Początkowo stanowisko jest wolne, a kolejka pusta.

W każdej minucie:

1. Losujemy przyjazd i dokładamy nowy samochód do kolejki.
2. Jeżeli stanowisko jest wolne, pobieramy pierwszy oczekujący samochód i rozpoczynamy mycie.
3. Wykonujemy minutę mycia; czysty samochód opuszcza stanowisko.

Implementacja symulacji: simulation.py.

Myjnia samochodowa – czas oczekiwania

Przy rozpoczęciu mycia mierzymy:

$$\text{oczekiwanie} = \text{chwila rozpoczęcia} - \text{chwila przyjazdu}.$$

Średnią liczymy dla samochodów, które rozpoczęły mycie.

- Średnie zapotrzebowanie: $3r$ minut mycia na minutę.
- Wydajność stanowiska: minuta mycia na minutę.
- Dla $r > \frac{1}{3}$ oczekujemy narastania kolejki.

Eksperymenty: `simulation.py`.
(Skrypt: wykład 3 (cz. 2), przykład myjni.)

Lista łączona

Lista łączona (*linked list*) – struktura danych, w której elementy przechowywane są w *węzłach*. Węzły są ułożone w łańcuch.

Węzeł przechowuje obiekt i odwołanie do następnego węzła. Ostatni węzeł nie ma następnika.

Lista pamięta **głowę** (pierwszy węzeł) i **ogon** (ostatni węzeł).

Lista pusta nie ma głowy ani ogona. Lista jednoelementowa ma głowę i ogon w tym samym węźle.

Implementacja: materiały do wykładu i skrypt, wykład 4 (cz. 2), listy łączone.

Lista łączona – operacje

- **Dołożenie na początek:** nowy węzeł wskazuje dotychczasową głowę i ją zastępuje.
- **Dołożenie na koniec:** dotychczasowy ogon wskazuje nowy węzeł, który zostaje ogonem.
- W pustej liście nowy węzeł zostaje głową i ogonem.
- **Usunięcie głowy:** zastępujemy ją następnikiem; po usunięciu jedynego węzła usuwamy też ogon.

Lista łączona – koszty operacji

Niech n to liczba węzłów w liście. Czasy asymptotyczne operacji:

- Dokładanie na obu końcach i usuwanie głowy: $\Theta(1)$.
- Usuwanie ogona wymaga znalezienia poprzednika: $\Theta(n)$.
- Wyznaczanie rozmiaru przez przejście listy: $\Theta(n)$.

Zastosowanie: kolejka ze stałym czasem dokładania i usuwania elementu.

Implementacja i przykłady: skrypt wykład 4 (cz. 2) i materiały do wykładu.

Hermetyzacja

Interfejs publiczny opisuje dozwolone działania. **Szczegóły implementacji** pozostają ukryte.

- Korzystamy z operacji listy; samodzielna zmiana ogona może zerwać łańcuch węzłów.
- Węzły, głowa i ogon mogą być niepubliczne.
- Python: początkowe podkreślenie oznacza umownie niepubliczną nazwę; dwa podkreślenia uruchamiają przekształcanie nazwy (*name mangling*), nie pełną kontrolę dostępu.

Wariant z ukrytymi szczegółami: skrypt, rozdział 4 (cz 2).