LAPORAN PRAKTIKUM

PRAKTIKUM STRUKTUR DATA

PERTEMUAN 6

DEQUEUE



Disusun oleh:

Nama : Aditya Lucky Zulkarnaen

NIM : 24/537764/SV/24449

Kelas : PLB1

Dosen Pengampu : Dr. Umar Taufiq, S.Kom., M.Cs.

PROGRAM STUDI D-IV TEKNOLOGI REKAYASA PERANGKAT LUNAK

DEPARTEMEN TEKNIK ELEKTRO DAN INFORMATIKA

SEKOLAH VOKASI

UNIVERSITAS GADJAH MADA

YOGYAKARTA

2025

**PERTEMUAN 6**

**DEQUEUE**

1. **Tujuan Percobaan**
2. Memahami konsep dasar dan karakteristik deque sebagai struktur data yang dapat diakses dari kedua ujungnya.
3. Mempelajari implementasi deque dalam Python menggunakan modul collections serta fungsi-fungsi utamanya.
4. Mengaplikasikan deque dalam berbagai kasus penggunaan, seperti palindrome checker dan sliding window problems.
5. **Dasar Teori**
6. Dequeue

Deque (Double-Ended Queue) adalah struktur data yang memungkinkan penambahan dan penghapusan elemen dari kedua ujungnya, baik depan maupun belakang. Berbeda dengan queue biasa yang hanya mendukung operasi First In First Out (FIFO), deque dapat berfungsi sebagai queue maupun stack, tergantung pada bagaimana operasinya diatur. Hal ini menjadikan deque lebih fleksibel dibandingkan struktur data linear lainnya. (Oregon State University, n.d.)

Jenis Deque :

* Input-Restricted Deque: Penambahan elemen hanya dapat dilakukan di satu ujung, sedangkan penghapusan elemen dapat dilakukan di kedua ujung.
* Output-Restricted Deque: Penghapusan elemen hanya dapat dilakukan di satu ujung, sedangkan penambahan elemen dapat dilakukan di kedua ujung. (Shruti, 2025)

1. Aplikasi Dequeue

* Deque sering digunakan dalam berbagai aplikasi, seperti:
* Palindrome Checker: Memeriksa apakah suatu string atau angka adalah palindrome.
* Task Scheduling: Mengelola tugas-tugas yang memerlukan fleksibilitas dalam urutan eksekusi.
* Sliding Window Problems: Digunakan untuk mencari nilai maksimum atau minimum dalam jendela geser pada array. (Ricciardi, 2024)

1. Deque pada Python dan Fungsinya

Python menyediakan implementasi deque melalui modul bawaan collections. Struktur ini dirancang untuk operasi yang efisien pada kedua ujungnya dengan kompleksitas waktu O(1). (Enterprise DNA, n.d.) Fungsi-Fungsi Utama Deque:

1. Membuat Dequeue

from collections import deque

dq = deque() # Membuat deque kosong

dq = deque([1, 2, 3]) # Membuat deque dengan elemen awal

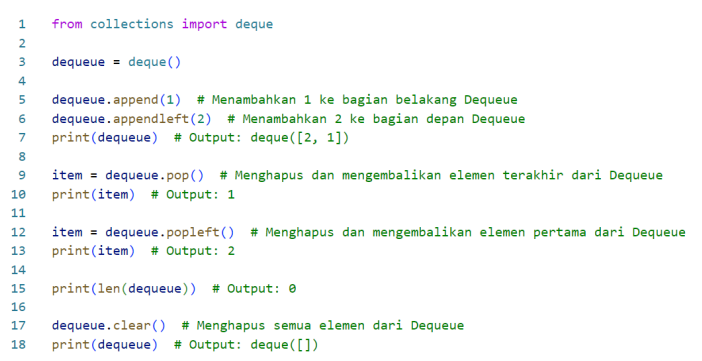
1. Operasi Dasar

* append(x): Menambahkan elemen ke ujung kanan.
* appendleft(x): Menambahkan elemen ke ujung kiri.
* pop(): Menghapus dan mengembalikan elemen dari ujung kanan.
* popleft(): Menghapus dan mengembalikan elemen dari ujung kiri.

1. Operasi Lanjutan

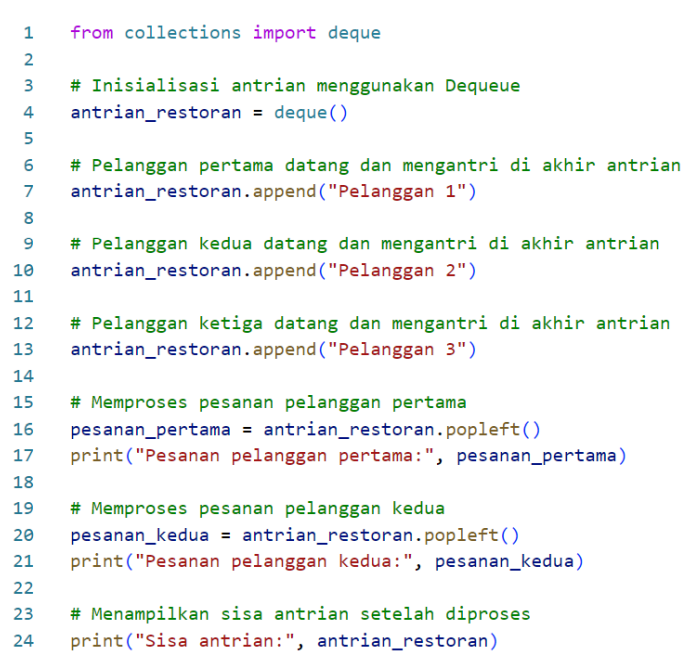
* extend(iterable): Menambahkan beberapa elemen ke ujung kanan.
* extendleft(iterable): Menambahkan beberapa elemen ke ujung kiri.
* rotate(n): Memutar elemen sebanyak nn langkah; positif untuk ke kanan, negatif untuk ke kiri. (Okasaki, 1998)

1. **Hasil dan Pembahasan**
2. Penjelasan Kode 1



* from collections import deque : Mengimpor deque dari collections, yang merupakan struktur data antrian ganda yang memungkinkan penambahan dan penghapusan elemen dari kedua ujungnya.
* dequeue = deque() : Membuat objek deque kosong.
* dequeue.append(1) : Menambahkan 1 ke belakang deque.
* appendleft(2) : Menambahkan 2 ke depan deque.
* item = dequeue.pop() : Menghapus elemen terakhir (1) dari deque dan menyimpannya dalam item.
* item = dequeue.popleft() : Menghapus elemen pertama (2) dari deque.
* print(len(dequeue)) : Karena semua elemen telah dihapus, panjang deque adalah 0.
* dequeue.clear() : clear() → Menghapus semua elemen dalam deque.

1. Penjelasan Kode 2



* from collections import deque : Mengimpor deque dari modul collections, yang merupakan struktur data antrian ganda yang memungkinkan penambahan dan penghapusan elemen dari kedua ujungnya.
* antrian\_restoran = deque() : Membuat objek deque kosong untuk merepresentasikan antrian restoran.
* antrian\_restoran.append("Pelanggan 1") : Menambahkan "Pelanggan 1" ke bagian belakang antrian.
* antrian\_restoran.append("Pelanggan 2") : Menambahkan "Pelanggan 2" ke bagian belakang antrian.
* antrian\_restoran.append("Pelanggan 3") : Menambahkan "Pelanggan 3" ke bagian belakang antrian.
* pesanan\_pertama = antrian\_restoran.popleft() : Menghapus elemen pertama ("Pelanggan 1") dari deque dan menyimpannya dalam pesanan\_pertama.
* pesanan\_kedua = antrian\_restoran.popleft() : Menghapus elemen pertama ("Pelanggan 2") dari deque dan menyimpannya dalam pesanan\_kedua.
* print(len(antrian\_restoran)) : Karena dua pelanggan telah diproses, panjang deque sekarang adalah 1.
* print(antrian\_restoran) : Menampilkan sisa antrian yang masih berisi "Pelanggan 3".

1. Implementasi Dequeue

Deque dapat digunakan untuk menyelesaikan berbagai masalah algoritmik. Berikut adalah contoh implementasi sederhana

1. Palindrome Checker

from collections import deque

def is\_palindrome(string):

dq = deque(string)

while len(dq) > 1:

if dq.popleft() != dq.pop():

return False

return True

print(is\_palindrome("radar")) # Output: True

print(is\_palindrome("hello")) # Output: False

* from collections import deque : Mengimpor deque dari modul collections, yang digunakan untuk memeriksa apakah suatu kata adalah palindrom.
* def is\_palindrome(string): : Mendefinisikan fungsi is\_palindrome() untuk mengecek apakah sebuah kata merupakan palindrom.
* dq = deque(string) : Mengubah string menjadi objek deque, sehingga karakter dalam string dapat diakses dari kedua ujungnya.
* while len(dq) > 1: : Melakukan iterasi selama panjang deque lebih dari 1.
* if dq.popleft() != dq.pop(): : Menghapus dan membandingkan karakter pertama (popleft()) dengan karakter terakhir (pop()). Jika berbeda, bukan palindrom, sehingga mengembalikan False.
* return True : Jika semua karakter cocok saat dibandingkan, string adalah palindrom dan fungsi mengembalikan True.
* print(is\_palindrome("radar")) : Memeriksa apakah "radar" adalah palindrom. Output: True.
* print(is\_palindrome("hello")) : Memeriksa apakah "hello" adalah palindrom. Output: False.

1. Sliding Window Maximum

from collections import deque

def sliding\_window\_max(nums, k):

dq = deque()

result = []

for i in range(len(nums)):

while dq and dq[0] < i - k + 1:

dq.popleft()

while dq and nums[dq[-1]] < nums[i]:

dq.pop()

dq.append(i)

if i >= k - 1:

result.append(nums[dq[0]])

return result

print(sliding\_window\_max([1,3,-1,-3,5,3,6,7], 3)) # Output: [3,3,5,5,6,7]

* from collections import deque : Mengimpor deque dari modul collections, yang digunakan untuk menyimpan indeks dalam window.
* def sliding\_window\_max(nums, k): : Mendefinisikan fungsi sliding\_window\_max() untuk mencari nilai maksimum dalam setiap window berukuran k.
* dq = deque() : Inisialisasi deque kosong untuk menyimpan indeks elemen dalam nums.
* result = [] : List kosong untuk menyimpan hasil nilai maksimum setiap window.
* for i in range(len(nums)): : Melakukan iterasi pada setiap indeks i dalam nums.
* while dq and dq[0] < i - k + 1: : Menghapus indeks dari deque yang berada di luar jendela ukuran k.
* dq.popleft() : Menghapus elemen pertama dari deque.
* while dq and nums[dq[-1]] < nums[i]: : Menghapus indeks dari deque jika elemen yang baru lebih besar dari elemen yang disimpan.
* dq.pop() : Menghapus elemen terakhir dari deque.
* dq.append(i) : Menambahkan indeks i ke deque.
* if i >= k - 1: : Mulai menambahkan nilai maksimum ke result setelah ukuran window mencapai k.
* result.append(nums[dq[0]]) : Elemen pertama di deque adalah indeks dari nilai maksimum dalam window.
* return result : Mengembalikan daftar nilai maksimum dari setiap window.
* print(sliding\_window\_max([1,3,-1,-3,5,3,6,7], 3)) : Mencetak hasil fungsi dengan input [1,3,-1,-3,5,3,6,7] dan window k=3. Output: [3,3,5,5,6,7].

1. **Kesimpulan**

Deque (Double-Ended Queue) adalah struktur data yang fleksibel karena memungkinkan operasi penambahan dan penghapusan elemen dari kedua ujungnya, baik depan maupun belakang. Berbeda dengan queue biasa yang mengikuti prinsip FIFO (First In First Out), deque dapat digunakan sebagai queue maupun stack, tergantung pada cara operasinya dilakukan. Python menyediakan deque melalui modul collections, yang memungkinkan operasi dasar seperti append(), appendleft(), pop(), dan popleft() dengan efisiensi tinggi. Selain itu, deque memiliki variasi seperti Input-Restricted Deque dan Output-Restricted Deque yang membatasi operasi tertentu pada satu ujung.

Deque memiliki berbagai aplikasi dalam pemrograman, seperti pemeriksaan palindrome, penjadwalan tugas (task scheduling), dan penyelesaian masalah sliding window. Implementasi deque dalam Python sangat berguna dalam optimasi algoritma yang membutuhkan akses cepat ke kedua ujung struktur data. Dengan kompleksitas waktu O(1) untuk operasi dasar, deque menjadi pilihan yang efisien untuk berbagai kebutuhan pemrosesan data.

1. **Daftar Pustaka**

Shruti. (2025, Januari 26). Learn Deque in Data Structure From Scratch. Retrieved from simplilearn: https://www.simplilearn.com/tutorials/data-structure-tutorial/dequeue-in-data-structure

Oregon State University. (n.d.). Queues and Deques. Retrieved March 26, 2025, from https://classes.engr.oregonstate.edu/eecs/spring2020/cs261/notes/queues.html

Enterprise DNA. (n.d.). Deque in Python – Tutorial With Examples. Retrieved March 26, 2025, from https://blog.enterprisedna.co

Ricciardi, A. S. (2024, October 4). Queue, Deque, and Priority Queue: Key Data Structures Explained. Retrieved from Medium: https://levelup.gitconnected.com/queue-deque-and-priority-queue-key-data-structures-explained-1509f133d4c5

Okasaki, C. (1998). Simple and Efficient Purely Functional Queues and Deques. Journal of Functional Programming, 5(4), 583–592. https://doi.org/10.1017/S0956796897002861