Belajaritma

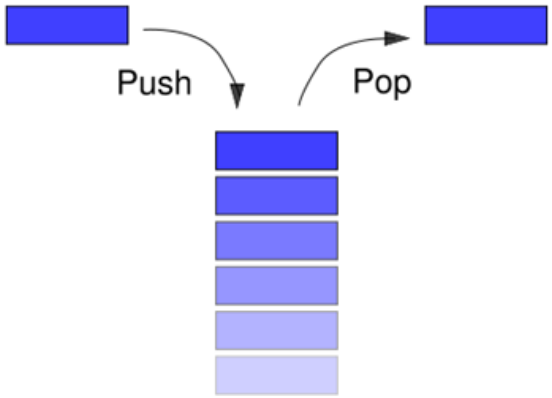
Data Structure

Sesi III : Stack and Queue
(PDF Example)

Himpunan Mahasiswa
HIMTI (Teknik Informatika)
BINUS
UNIVERSITY

Stack

1. Stack Concept



- Stack adalah kumpulan dari *data structure* yang memiliki 2 macam operasi utama: **push** (stack), **pop** (mengambil/menghapus) data yang paling terakhir di input.
- Data stack disimpan dengan cara ***Last In First Out (LIFO)***, data terakhir yang di input maka akan keluar pertama kali.
- Stack bisa diimplementasikan dengan *Linked list* maupun *Array*.

Meskipun Implementasi stack dengan Array lebih mudah, namun memiliki beberapa kelemahan diantaranya :

- Static data - data yang diinput bersifat *fixed* sehingga dapat menyebabkan kurangnya / kelebihan space memory.
- Data yang diinput selalu disimpan secara berurutan.

- Dalam stack, dikenal 2 macam variabel pointer utama, yakni :

- **TOP**, yakni digunakan untuk menyimpan alamat dari elemen yang terletak di paling atas dalam *stack*.
- **MAX,** yakni digunakan untuk menyimpan banyaknya maksimum elemen yang dapat ditampung dalam sebuah *stack*.

- If TOP = NULL, menandakan bahwa stack tersebut kosong.
- If TOP = MAX - 1, menandakan bahwa stack tersebut telah menampung maksimum elemen yang dapat ditampung.
- TOP = 4, *insertion* dan *deletion* akan dilakukan dari titik ini.

Implementasi *Linked list* dalam stack:

- dalam sebuah *linked stack*, sebuah node memiliki 2 bagian :
  - Untuk menyimpan data
  - Untuk menyimpan alamat dari node selanjutnya
- Pointer awal (*START)* dalam sebuah linked list digunakan sebagai TOP.
- Semua *insertion* dan *deletion* dilakukan dari node yang ditunjuk oleh TOP.

Note: TOP juga dikenal sebagai *peek* (puncak).

Beberapa macam operasi dalam stack:

- pushTop(x): menambahkan sebuah *node x* kedalam stack.

  Implementasi coding: *Insertion*

```
19
20  mahasiswa *pushTop(char namaInsert[], int umurInsert, char alamatInsert[]){
21      //2 macam kemungkinan
22      //1. top = NULL / tidak ada data
23      if(top == NULL){
24          top = pabrikNode(namaInsert, umurInsert, alamatInsert);
25      }
26      //2. top != NULL / data lebih dari 1
27      else{
28          mahasiswa *newNode = pabrikNode(namaInsert, umurInsert, alamatInsert);
29          newNode->next = top;
30          top = newNode;
31      }
32  }
33
```

- popTop(x): melepas sebuah node dari bagian puncak / *TOP*

  Implementasi coding: *Deletion*

```
33
34  mahasiswa *popTop(){
35      //2 macam kemungkinan
36      //1. top = NULL / tidak ada data
37      if(top == NULL){
38          printf("Data Tidak Tersedia\n");
39          return 0;
40      }
41      //2. jika top != NULl // setidaknya ada 1 data
42      else{
43          mahasiswa *temp = top;
44          top = top->next;
45          free(temp);
46      }
47  }
48
```

- peek(x): untuk memunculkan / meng-*return* item dari bagian puncak / *TOP*

Implementasi coding: *Display - top*

```
64  mahasiswa *peek(){
65      //2 macam kemungkinan
66      //1. top = NULL / tidak ada data
67      if(top == NULL){
68          printf("Tidak ada data");
69          return 0;
70      }
71      //2. top != NULL
72      printf("Nama : %s Umur: %d Alamat: %s\n", top->nama, top->umur, top->alamat);
73  }
74
```

- printAll(x): untuk memunculkan seluruh node dalam sebuah *stack* dimulai dari bagian puncak / *TOP*.

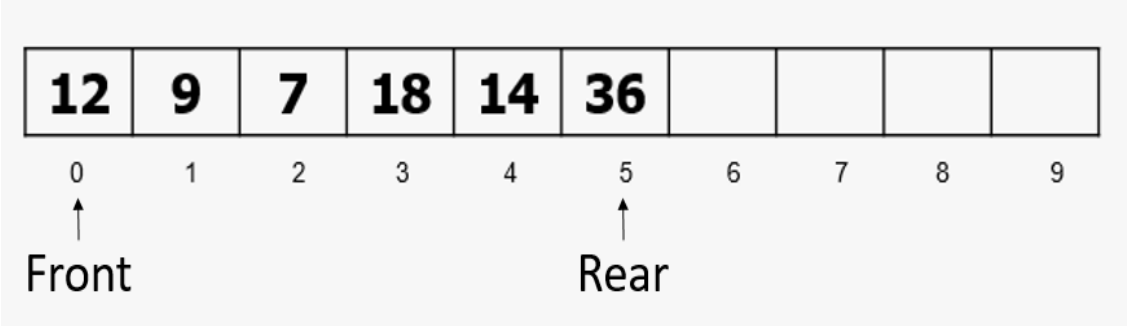
  Implementasi coding: *Display*

```
49  mahasiswa *printAll(){
50      //2 macam kemungkinan
51      //1. top = NULL / tidak ada data
52      if(top == NULL){
53          printf("Tidak ada data");
54          return 0;
55      }
56      //2.top != NULL / setidaknya terdapat 1 data
57      mahasiswa *curr = top;
58      while(curr != NULL){
59          printf("Nama : %s Umur: %d Alamat: %s\n", curr->nama, curr->umur, curr->alamat);
60          curr = curr->next;
61      }
62  }
63
```

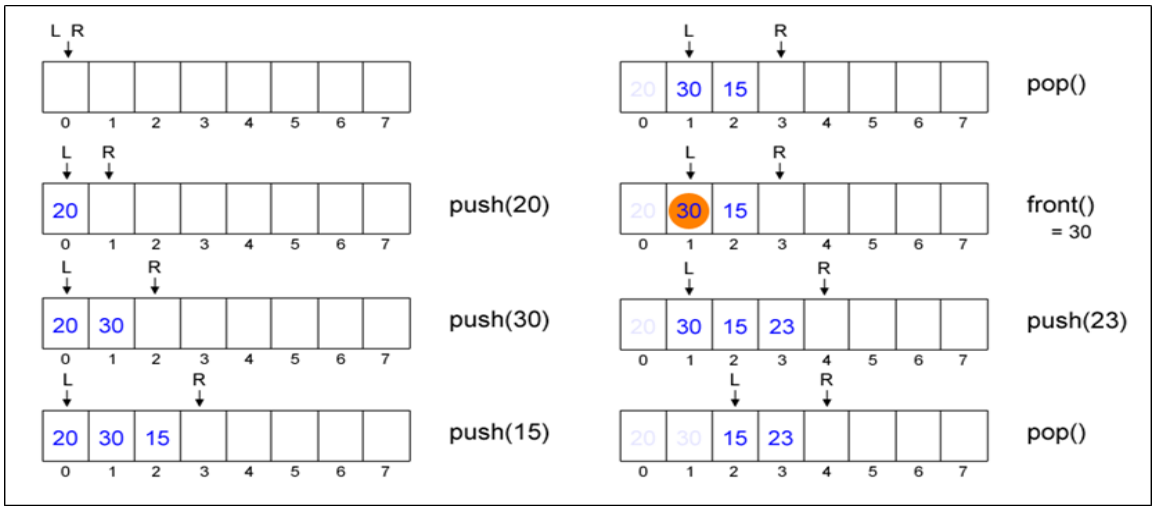
Himpunan Mahasiswa
HIMTI (Teknik Informatika)
BINUS
UNIVERSITY

Queue

1. Queue Concept
   - Queue adalah sekumpulan struktur data dengan 2 operasi dasar yaitu **Push** dan **Pop**
   - Dengan metode **First In First Out (FIFO) =** Data yang duluan masuk juga akan duluan keluar.
   - Queue memiliki 2 variabel:
     - **Front** dan **Rear** yang mengindikasikan data terdepan dan data terakhir.
   - Contohnya:



     - **Front = 0** dan **Rear = 5**
2. Queue Operation
   - push(x) : Menambahkan data x ke belakang Queue.
   - pop() : Mendelete data yang berada di depan Queue.
   - front() : Meng-return data yang berada di Queue paling depan. front juga disebut sebagai **peek**.
3. Example of Queue Operations



4. Queue Code (http://bit.ly/TutorQueueCode)

- Struct Declaration:

```
3
4   struct node{
5       int value;
6       struct node *next;
7   } *front = NULL, *rear = NULL;
8
```

- Function Insertion:

```
9   void insertNode(int a){ //Push yang digunakan pada code ini adalah pushTail
10      struct node *newNode;
11      newNode = (struct node *) malloc(sizeof(struct node));
12      newNode->value = a;
13      newNode->next = NULL;
14      if (front == NULL){
15          front = rear = newNode;
16      }
17      else {
18          rear->next = newNode;
19          rear = newNode;
20      }
21  }
```

- Function Deletion:

```
22
23  void deleteNode(){ //Pop yang digunakan pada code ini adalah popHead
24      struct node *temp = front;
25      if (front ==  rear){
26          free(front);
27          front = rear = NULL;
28      }
29      else {
30          front = front->next;
31          free(temp);
32      }
33  }
34
```

- Function Display:

```
34
35  void display(){
36      struct node *temp = front;
37      if (front == NULL){
38          printf ("Queue is empty");
39      }
40      else {
41          while (temp != NULL){
42              printf ("%d -> ", temp->value);
43              temp = temp->next;
44          }
45          printf ("NULL\n");
46      }
47  }
48
```

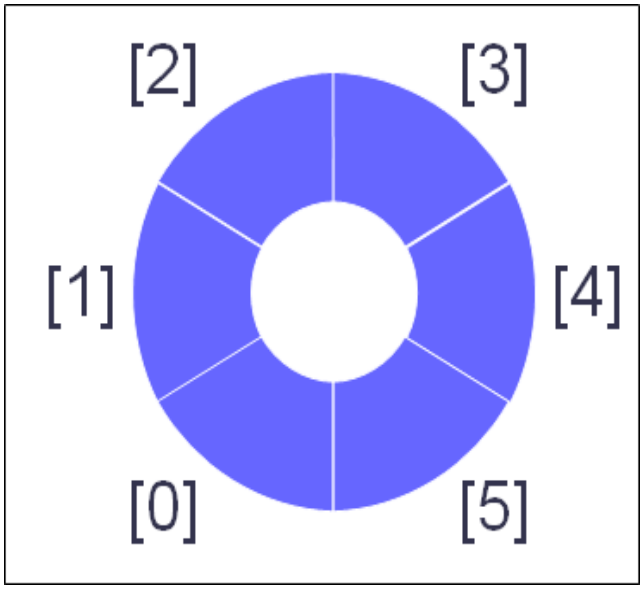
- Function Top/Peek:

```
48
49  void peek(){
50      printf ("The Front Value is %d\n", front->value);
51  }
52
```

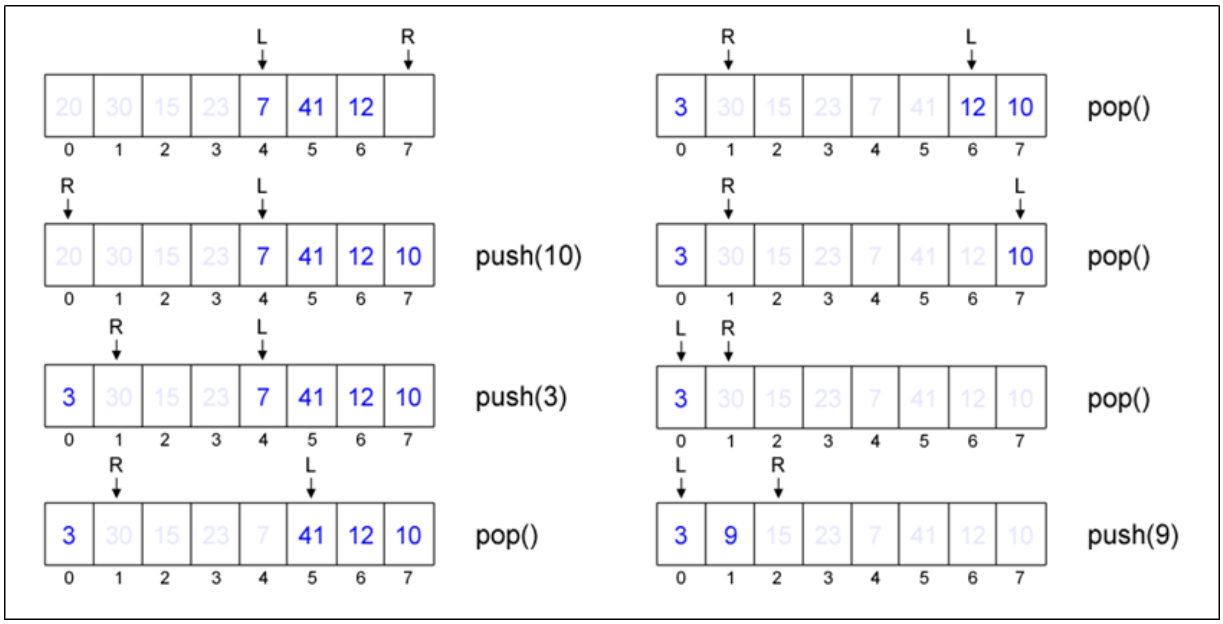
- Function Size (Meng-return jumlah data yang ada di Queue)

- Function is_empty (Meng-return true jika tidak ada data pada Queue dan false jika sebaliknya)
- Function is_full (Meng-return true jika data pada Queue mencapai limit maximum dan false jika tidak)

5. Circular Queue
    - Circular Queue adalah suatu struktur data linier dimana operasi dilakukan berdasarkan prinsip FIFO (First In First Out) dan posisi terakhir dihubungkan kembali ke posisi pertama dan membentuk sebuah lingkaran. Ini juga disebut 'Ring Buffer'.



- Contoh:



6. Deques - Double Ended Queue
   - Deque adalah daftar di mana elemen dapat disisipkan atau dihapus di kedua ujungnya. Ini juga dikenal sebagai head-tail linked list, karena elemen dapat ditambahkan atau dihapus dari depan (head) atau belakang (tail).
   - Operations:
     - Function **insertFront()**: Menambahkan data di depan queue.
     - Function **insertLast()**: Menambahkan data di belakang queue.
     - Function **deleteFront()**: Menghapus data yang ada di depan queue.
     - Function **deleteLast()**: Menghapus data yang ada di belakang queue.
     - Function **getFront()**: Mengambil data yang ada di depan queue.
     - Function **getRear()**: Mengambil data yang ada di belakang queue.
     - Function **isEmpty()**: Mengecek apakah queue kosong atau tidak.
     - Function **isFull()**: Mengecek apakah queue penuh atau tidak.
   - Ada 2 jenis deques:

- Input Restricted Deque

  Dalam deque ini, insertion hanya dapat dilakukan di salah satu ujung sedangkan deletion dapat dilakukan dari kedua ujungnya.

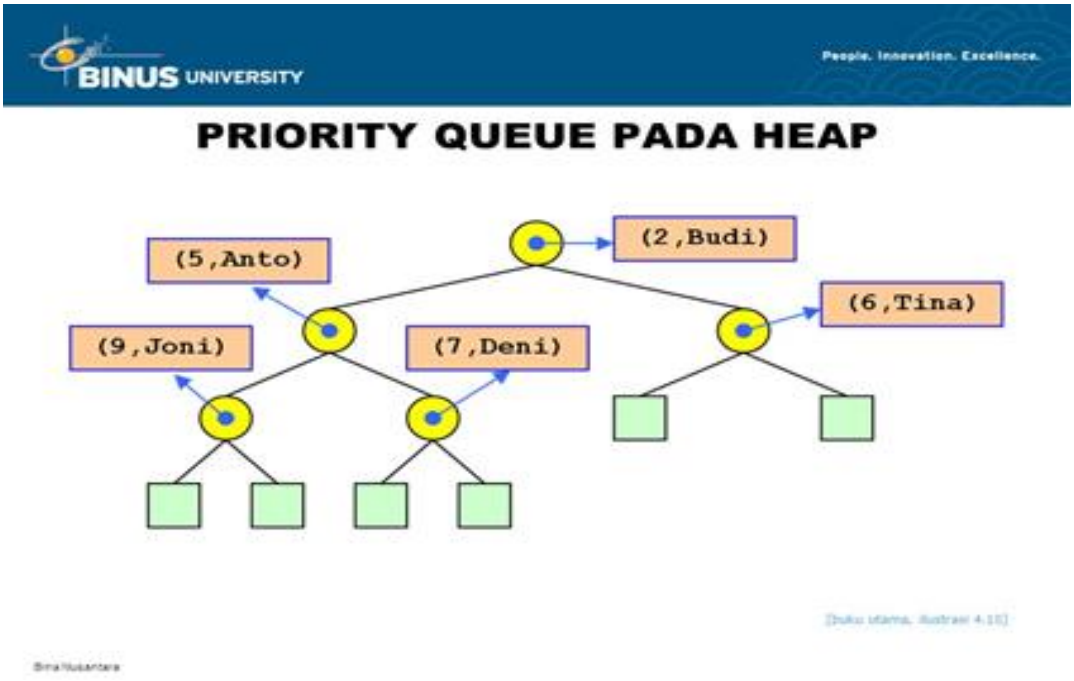
- Output Restricted Deque

  Dalam dequeue ini, deletion hanya dapat dilakukan di salah satu ujung sedangkan insertion dapat dilakukan di kedua ujungnya.

7. Priority Queue

- Priority Queue adalah Queue dengan basis HPIFO (Highest Priority In First Out) berbeda dengan Queue yang sangat bergantung kepada waktu kedatangan (elemen yang datang pertama akan selalu diambil atau dihapus pertama pula), sedangkan pada Priority Queue elemen yang akan diambil atau dihapus adalah elemen yang memiliki prioritas tertinggi (bila yang menjadi prioritas adalah waktu kedatangan maka Priority Queue berfungsi seperti Queue).
- 2 Tipe Priority Queue:
  - Ascending Priority Queue,yaitu Queue yang diurutkan dengan prioritas menaik, yaitu dari elemen yang memiliki prioritas terendah hingga prioritas tertinggi.
  - Descending Priority Queue,yaitu Queue yang diurutkan dengan prioritas menurun, yaitu dari elemen yang memiliki prioritas tertinggi hingga prioritas terendah.
- Untuk mempresentasikan Priority Queue dapat dilakukan dengan dua cara, yaitu Set dan List. Dengan Set, data di masuk kan ke dalam Queue

berdasarkan waktu kedatangan, sedangkan pengambilannya tetap berdasarkan prioritas. Keuntungan dari Set adalah operasi *enQueue* sangat cepat dan sederhana, tetapi operasi *deQueue* menjadi sangat kompleks karena diperlukan pencarian elemen dengan prioritas tertinggi, sedangkan dengan List, data di *enQueue* dan di deQueue berdasarkan prioritas.



Referensi:

- https://www.geeksforgeeks.org/queue-data-structure/
- https://www.geeksforgeeks.org/stack-data-structure-introduction-program/
- https://www.google.com/amp/s/kaaeka.wordpress.com/2011/12/18/priority-queue/amp/
- http://library.binus.ac.id/eColls/eThesisdoc/Bab2HTML/2009100390MTIFBab2/page11.html

- PPT Stack & Queue Binus University