


Estruturas de Dados / Programação 2
Filas

Márcio Ribeiro
marcio@ic.ufal.br
twitter.com/marciomribeiro

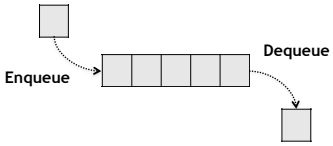
Still restricting access to some items...

Unlocking game levels

Queue

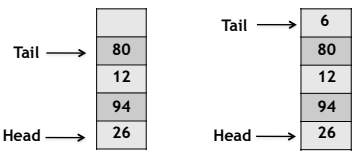
Queue

- FIFO: First In, First Out
- Just like the lines we are used to in banks, shows etc

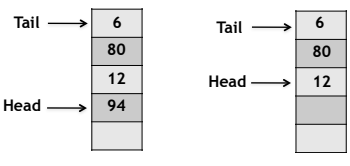


How does it work?

Adding 6...

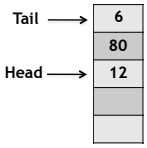


Removing 26 and 94...

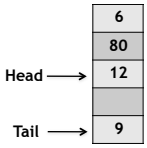


What about now? How to add at the end?

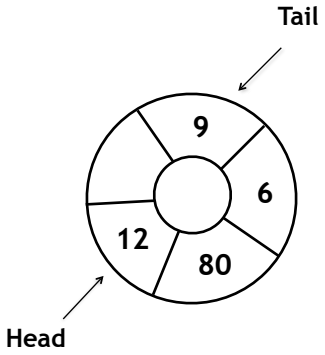
- The array contains empty elements…



- Circular Queue!
  - Adding 9!



Circular Queue



Abstract Data Type: Queue

Queue ADT

```
queue* create_queue();

void enqueue(queue *queue, int item);

int dequeue(queue *queue);

int is_empty(queue *queue);
```

Creating a Queue

```
#define MAX_QUEUE_SIZE 10

struct queue {
  int current_size;
  int head;
  int tail;
  int items[MAX_QUEUE_SIZE];
};

queue* create_queue()
{
  queue *new_queue = (queue*) malloc(sizeof(queue));
  new_queue->current_size = 0;
  new_queue->head = 0;
  new_queue->tail = MAX_QUEUE_SIZE – 1;
  return new_queue;
}
```

Enqueue items to the Queue

```
void enqueue(queue *queue, int item)
{
  if (queue->current_size >= MAX_QUEUE_SIZE) {
    printf("Queue overflow");
  } else {
    queue->tail = (queue->tail + 1) % MAX_QUEUE_SIZE;
    queue->items[queue->tail] = item;
    queue->current_size++;
  }
}
```

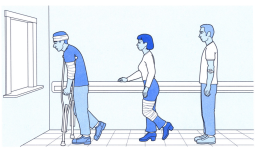
Dequeue items from the Queue

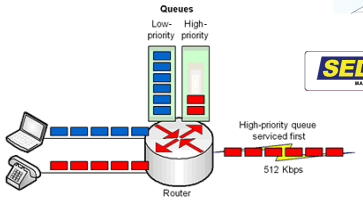
```
int dequeue(queue *queue)
{
  if (is_empty(queue)) {
    printf("Queue underflow");
    return -1;
  } else {
    int dequeued = queue->items[queue->head];
    queue->head = (queue->head + 1) % MAX_QUEUE_SIZE;
    queue->current_size--;
    return dequeued;
  }
}
```

Sometimes elements in a queue may have priorities...

Priorities

- Distributing CPU time among threads, network packages
- Hospital emergency rooms
- Priority letters





Priority Queue

Priority Queue

- When adding an item, we define a priority for it
- Instead of storing in chronological order, we archive items with the highest priority before all others
- Items are removed according to their priority
- No FIFO anymore

Example

```
enqueue(pq, “a”, 17);   [<a,17>]
enqueue(pq, “b”, 12);   [<a,17>,<b,12>]
enqueue(pq, “c”, 100);  [<c,100>,<a,17>,<b,12>]
enqueue(pq, “d”, 22);   [<c,100>,<d,22>,<a,17>,<b,12>]
dequeue(pq);            [<d,22>,<a,17>,<b,12>]
dequeue(pq);            [<a,17>,<b,12>]
```

Abstract Data Type: Priority Queue

PriorityQueue ADT

```
priority_queue* create_priority_queue();

void enqueue(priority_queue *pq, int i, int p);

node* dequeue(priority_queue *pq);

int maximum(priority_queue *pq);

int is_empty(priority_queue *pq);

void print_priority_queue(priority_queue *pq);
```

Implementing PriorityQueue with Lists

```
struct node {
  int item;
  int priority;
  node *next;
};
```

```
struct priority_queue {
  node *head;
};
```

Enqueue items to the Priority Queue

```
void enqueue(priority_queue *pq, int i, int p)
{
  node *new_node = (node*) malloc(sizeof(node));
  new_node->item = i;
  new_node->priority = p;
  if ((is_empty(pq)) || (p > pq->head->priority)) {
    new_node->next = pq->head;
    pq->head = new_node;
  } else {
    node *current = pq->head;
    while ((current->next != NULL) &&
           (current->next->priority > p)) {
      current = current->next;
    }
    new_node->next = current->next;
    current->next = new_node;
  }
}
```

Dequeue items from the Priority Queue

```
node* dequeue(priority_queue *pq)
{
  if (is_empty(pq)) {
    printf("Priority Queue underflow");
    return NULL;
  } else {
    node *node = pq->head;
    pq->head = pq->head->next;
    node->next = NULL;
    return node;
  }
}
```

Maximum

```
int maximum(priority_queue *pq)
{
  if (is_empty(pq)) {
    printf("Priority Queue underflow");
    return -1;
  } else {
    return pq->head->i;
  }
}
```

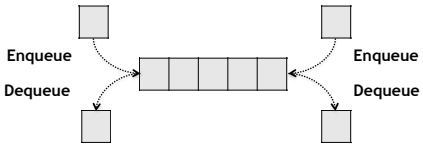
Efficiency of Priority Queues

- Enqueue
  - O(n)
- Dequeue
  - O(1)
- We can do much better using Heaps!
  - O(log n)
- We will come back to Heaps soon

Double-ended Queue

Double-ended Queue

- Known as “Deque”
- Supports item insertion and removal at both ends

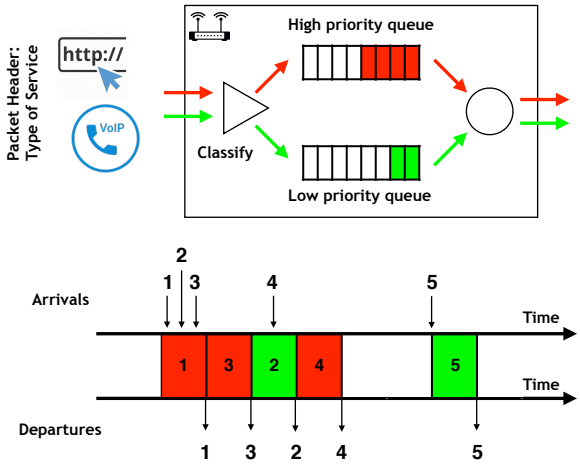


Homework

- Define the ADT for a Deque
- Implement it!

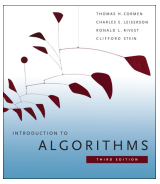
Application

Transmitting network packages



From *http://www2.ic.uff.br/~michael/kr1999/6-multimedia/6_06-scheduling_and_policing.htm*

References



Chapter 10

Chapter 4