Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный технологический университет»

**РЕФЕРАТ**

на тему:

“Очередь с приоритетом”

Выполнил:

Халалеенко Андрей Николаевич

Проверил:

Белодед Николай Иванович

Минск 2022

Оглавление

1. Понятие очереди с приоритетом
2. Разница между очередью с приоритетом и обычной очередью
3. Реализация очереди с приоритетом
4. Операции очередей с приоритетом

1)Вставка элемента в очередь с приоритетом

[Алгоритм вставки в очередь с приоритетом]

2)Удаление элемента из очереди с приоритетом

[Алгоритм удаления элемента из очереди с приоритетом]

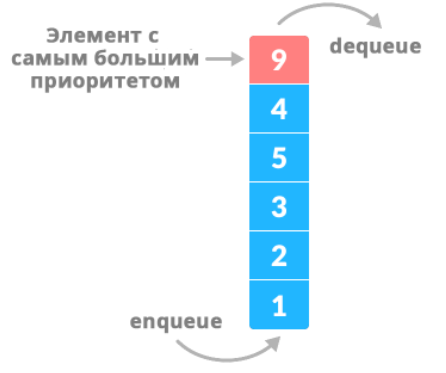
1. Реализация очереди с приоритетом
2. Где можно применять очередь с приоритетом
3. Вывод
4. **Понятие очереди с приоритетом**

Очереди с приоритетом — разновидность очередей, в которой у каждого элемента есть свой приоритет. Обслуживаются они в соответствии со своими приоритетом. Если у элементов одинаковый приоритет, то обслуживаются они по их порядку в очереди.

Значение элемента, как правило, и определяет его приоритет.

Примером очереди с приоритетом может служить порядок приема пациентов в больнице: чем тяжелее состояние пациента, тем выше его приоритет, а, следовательно, и принят он должен быть раньше остальных.

То есть, у элемента с самым большим значением самый высокий приоритет. Правда, это не всегда так. Самый высокий приоритет может быть у элемента и с самым малым значением. В остальных случаях мы можем задавать приоритеты элементам по своему усмотрению.



1. **Разница между очередью с приоритетом и обычной очередью**

Обычная очередь подчиняется принципу FIFO «первый вошел — первый вышел». В очередях с приоритетом элементы удаляются в соответствии с их приоритетом. То есть, элемент с самым высоким приоритетом удаляется из очереди в первую очередь.

1. **Реализация очереди с приоритетом**

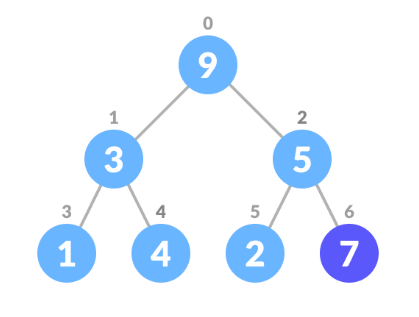
Очереди с приоритетом можно реализовать с помощью следующих структур данных: массив, связный список, куча и двоичное дерево поиска. Среди всех этих структур выделяется куча — это самый эффективный способ реализации очереди с приоритетом.

1. **Операции очередей с приоритетом**

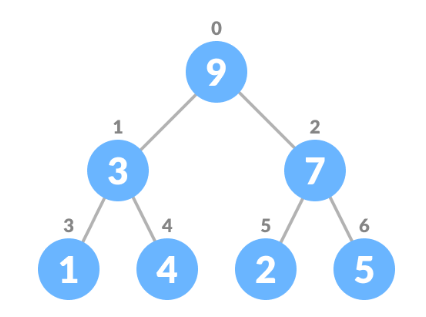
Основные операции очередей с приоритетом: вставка, удаление и просмотр элемента.

*1)Вставка элемента в очередь с приоритетом*

Вставляется новое значение



Сортируется дерево

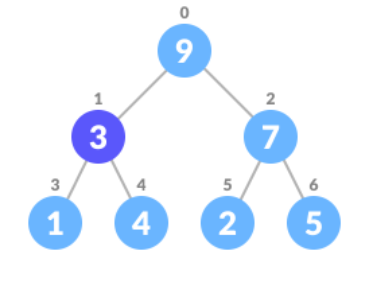


**Алгоритм вставки в очередь с приоритетом**

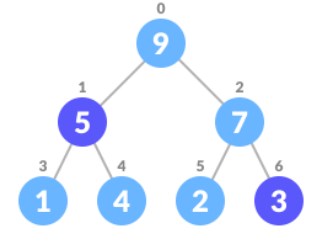
if нет узла,   
  создайте новый узел с помощью newNode.  
else узел уже есть  
  вставьте новый узел в конец дерева (последний узел слева — направо)  
отсортируйте массив

*2)Удаление элемента из очереди с приоритетом*

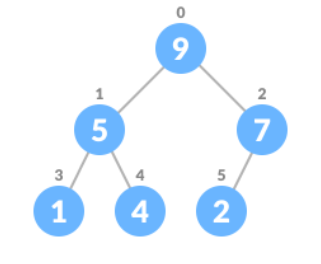
Выбор элемента, который нужно удалить



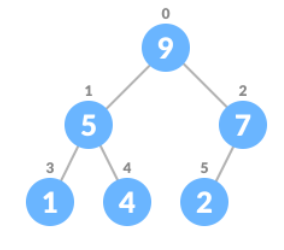
Поменять местами последний и удаляемый элемент



Удалить последний элемент



Отсортировать дерево



**Алгоритм удаления элемента из очереди с приоритетом**

if узел удаляемыйУзел = конечныйУзел  
  Удалить узел  
else поменять местами удаляемыйУзел и последнийУзел  
  Удалить удаляемыйУзел  
  
отсортировать дерево

1. **Реализация очереди с приоритетом**

#include <iostream>

#include <stdio.h>

int size = 0;

void swap(int\* a, int\* b) {

int temp = \*b;

\*b = \*a;

\*a = temp;

}

// Функция сортирует дерево

void heapify(int array[], int size, int i) {

if (size == 1)

printf("В куче один элемент");

else {

// Находим самое большое значение среди

// корневого, правого и левого дочернего элемента

int largest = i;

int l = 2 \* i + 1;

int r = 2 \* i + 2;

if (l < size && array[l] > array[largest])

largest = l;

if (r < size && array[r] > array[largest])

largest = r;

// Меняем местами и продолжаем сортировку,

// если значение корневого элемента не самое большое

if (largest != i) {

swap(&array[i], &array[largest]);

heapify(array, size, largest);

}

}

}

// Функция вставляет элемент

void insert(int array[], int newNum) {

if (size == 0) {

array[0] = newNum;

size += 1;

}

else {

array[size] = newNum;

size += 1;

for (int i = size / 2 - 1; i >= 0; i--){

heapify(array, size, i);

}

}

}

// Функция удаляет элемент

void deleteRoot(int array[], int num) {

int i;

for (i = 0; i < size; i++) {

if (num == array[i])

break;

}

swap(&array[i], &array[size - 1]);

size -= 1;

for (int i = size / 2 - 1; i >= 0; i--) {

heapify(array, size, i);

}

}

// Вывод массива в консоль

void printArray(int array[], int size) {

for (int i = 0; i < size; ++i)

printf("%d ", array[i]);

printf("\n");

}

int main() {

setlocale(LC\_CTYPE, "Russian");

int array[10];

insert(array, 3);

insert(array, 4);

insert(array, 9);

insert(array, 5);

insert(array, 2);

printf("Массив max-heap: ");

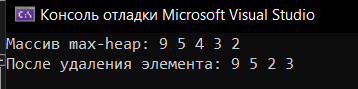
printArray(array, size);

deleteRoot(array, 4);

printf("После удаления элемента: ");

printArray(array, size);

}



1. **Где можно применять очередь с приоритетом**

Удобна для применения в:

1. Алгоритм Дейкстры
2. Реализация стека
3. Балансировка нагрузки и обработка прерываний в операционных системах
4. Сжатие данных в коде Хаффмана
5. **Вывод**

Очередь с приоритетом - это структура данных, представляющая собой обычную очередь, элементы которой упорядочены по их приоритету.

Очередь с приоритетами может работать как по вставке, так и по извлечению. Если очередь работает по вставке, то при добавлении нового элемента мы ищем в очереди позицию, в которую будет произведена вставка. Если очередь работает по вставке, то при извлечении элемента из очереди, мы ищем позицию, с которой элемент должен быть удален.