● Queue는 어떤 자료구조인가?

▶ 큐는 시간 순서상 먼저 집어 넣은 데이터가 먼저 나오는 선입선출, First in First Out(FIFO) 형식으로 데이터를 저장하는 자료 구조입니다. 시간 복잡도는 enqueue는 O(1), dequeue도 O(1)입니다. 활용 예시로는 하나의 자원을 공유하는 프린터나 캐시 구현, 프로세스 스케듈링, 또는 너비우선탐색(BFS) 등이 있습니다.

● enqueue & dequeue

▶ queue에 데이터를 추가하는 것을 enqueue라고 하고, 데이터를 추출 하는 것은 dequeue라고 합니다. enqueue의 경우 queue의 맨 뒤에 데이터를 추가하면 완료되기 때문에 시간복잡도는 O(1) 입니다. 이와 비슷하게 dequeue의 경우 맨 앞의 데이터를 삭제하면 완료 되기 때문에, 동일하게 O(1)의 시간복잡도를 갖습니다.

● 구현 방식

▶ Array-Based queue (ArrayList를 사용해서 queue를 구현): enqueue와 dequeue 과정에서 남는 메모리가 생기기 때문에, 메모리 낭비를 줄이기 위해서 주로 원형큐(Circular queue) 형식으로 구현을 합니다. Circular Queue에서는 enqueue를 할 때 더이상 배열의 끝에 자리가 없을 경우 dequeue로 인해 비어버린 앞자리가 있다면 그 앞자리에 원소를 추가하여 메모리 낭비를 줄입니다.

▶ List-Based queue(Linked-List를 사용해서 queue를 구현): 재할당이나 메모리 낭비의 걱정을 할 필요가 없어집니다. enqueue를 할 때마다 새로운 메모리를 할당하고, dequeue를 할 때마다 메모리 할당했던 것을 삭제하면 되기 때문입니다.

● 확장 & 활용 방식

▶ queue의 개념에서 확장한 자료구조들로는 양쪽에서 enqueue, dequeue가 가능한 deque(double-ended queue)와, 시간 순서가 아닌 우선 순위가 높은 순서로 dequeue 할 수 있는 priority queue가 있습니다.

▶ queue의 활용 예시로는 하나의 자원을 공유하는 프린터나, CPU task scheduling, Cache구현, 너비우선탐색(BFS)등이 있습니다.

→ 프린터: 선입선출이 필요한 대기열로, 먼저 올린 문서를 먼저 프린트합니다.

→ CPU task scheudling (process shceudling): 프로세스 스케쥴링은 실행될 프로세스가 여러 개 있으면 하나만 실행되고 나머지는 CPU 가 자유로워질 때까지 대기하는 것으로 선입선출의 방식을 따릅니다. 큐의 종듀에는 작업 큐(Job Queue), 준비 큐(Ready Queue), 장치 큐(Device Queue)가 있습니다. 작업 큐는 메모리 할당을 대기 중인 프로세스들로 구성되고, 준비 큐는 CPU 할당을 대기 중인 프로세스들로 구성되며, 장비 큐는 입출력 장치 할당을 대기 중인 프로세스들로 구성됩니다.

→ 캐시(Cache)구현: 데이터베이스의 데이터를 정적인 메모리에 보관해뒀다가 필요할 때 데이터베이스가 아니라 메모리에서 바로 꺼내 쓰기 위한 것입니다.

→ 너비우선탐색(BFS): 최단 경로를 찾는 것으로, 선입선출 방식인 큐 자료구조를 이용하여 시작 노드에서 가장 가까운 노드부터 우선적으로 탐색을 진행합니다.

● Array-Base와 List-Base의 경우 어떤 차이가 있나?

▶ Array-Base의 경우 메모리를 효율적으로 사용하기 위해서 circular queue로 구현하는 것이 일반적입니다. 또한, enqueue가 계속 발생하면 fixed size를 넘어서게 되기 때문에, dynamic array와 같은 방법으로 Array의 size를 확장시켜야 합니다. 그럼에도 enqueue의 시간복잡도는 (amortized)O(1)를 유지할 수 있습니다.

List-Base의 경우 보통 singly-lilnked list로 구현을 합니다. enqueue는 단순히 singly-lilnked list에서 append를 하는 것으로 구현되고, 이 때 시간복잡도는 O(1)입니다. dequeue는 맨 앞의 원소를 삭제하고 first head를 변경하면 되기 때문에, 이 연산도 O(1)의 시간이 걸립니다.

요약하자면, 두 가지 종듀의 자료구조로 queue를 구현을 하더라도 enqueue와 dequeue는 모두 O(1)의 시간복잡도를 갖습니다. Array-Base의 경우 전반적으로 performance가 더 좋지만, resize가 발생하는 worst case의 경우에는 훨씬 더 느릴 수 있습니다 (O(n)). List-Base의 경우에는 enqueue를 할 때마다 memory allocation을 해야하기 때문에, 전반적인 runtime이 느릴 수 있습니다.

Resources: inflearn 개발남 노씨