자료구조 HW #6 큐의 응용 : 시뮬레이션의 개선

전자전기공학부 20196891 이민영

(1) 서비스하는 사람이 여러 명(예를 들어 3명)일 경우

C 코드 큐.m (기존 설정으로 파일 첨부)

```c
#include <stdio.h>
#include <stdlib.h>

#define MAX_QUEUE_SIZE 100
#define No_SERVICEMAN 3

#define TRUE 1
#define FALSE 0

typedef struct {
        int id;
        int arrival_time;
        int service_time;
} element;

typedef struct {
        element queue[MAX_QUEUE_SIZE];
        int front, rear;
} Queuetype;
Queuetype queue;

// 에러
void error(char str[]) {
        printf("%s₩n", str);
        exit(1);
}

// 큐의 초기화 함수
void init(Queuetype *q) {
        q->front = q->rear = 0;
}

// 공백 상태 검출 함수
int is_empty(Queuetype *q) {
        return (q->front == q->rear);
}

// 포화 상태 검출 함수
int is_full(Queuetype* q) {
        return ((q->rear + 1) % MAX_QUEUE_SIZE == q->front);
}

// 삽입 함수
void enqueue(Queuetype* q, element item) {
```

```
        if (is_full(q))
                error("큐가 포화상태입니다");
        q->rear = (q->rear + 1) % MAX_QUEUE_SIZE;
        q->queue[q->rear] = item;
}

// 삭제 함수
element dequeue(Queuetype* q) {
        if (is_empty(q))
                error("큐가 공백상태입니다");
        q->front = (q->front + 1) % MAX_QUEUE_SIZE;
        return q->queue[q->front];
}

//0에서 1 사이의 난수 생성 함수
double random()
{
        return rand() / (double)RAND_MAX;
}

// 시뮬레이션에 필요한 여러가지 상태변수
int duration = 10;       //시뮬레이션 시간
double arrival_prob = 0.7;       // 하나의 시간 단위에 도착하는 평균 고객
int max_serv_time = 5;           // 하나의 고객에 대한 최대 서비스 시간
int clock;

// 시뮬레이션의 결과
int customers = 0;                       // 전체 고객수
int served_customers = 0;// 서비스받은 고객수
int waited_time = 0;             // 고객들이 기다린 시간
int actual_service = 0;          // 서비스하는 사람이 서비스한 총 시간

// 랜덤숫자를 생성하여 고객들이 도착했는지 도착하지 않았는지를 판단
int is_customer_arrived()
{
        if (random() < arrival_prob)
                return TRUE;
        else
                return FALSE;
}

// 새로 도착한 고객을 큐에 삽입
void insert_customer(int arrival_time)
{
        element customer;

        customer.id = customers++;
        customer.arrival_time = arrival_time;
        customer.service_time = (int)(max_serv_time * random()) + 1;
        enqueue(&queue, customer);
        printf("고객 %d이 %d분에 들어옵니다. 서비스 시간은 %d분입니다.₩n", customer.id,
customer.arrival_time, customer.service_time);
}

// 큐에서 기다리는 고객을 꺼내어 고객의 서비스 시간 반환
```

```
int remove_customer()
{
        element customer;
        int service_time = 0;

        if (is_empty(&queue)) return 0;
        customer = dequeue(&queue);
        service_time = customer.service_time - 1;
        served_customers++;
        waited_time += clock - customer.arrival_time;
        printf("고객 %d이 %d분에 서비스를 시작합니다. 대기시간은 %d분이었습니다.₩n",
customer.id, clock, clock - customer.arrival_time);
        return service_time;
}

// 통계치 출력
print_stat()
{
        printf("서비스 받은 고객 수 = %d₩n", served_customers);
        printf("전체 대기 시간 = %d분₩n", waited_time);
        printf("1인당 평균 대기 시간 = %f분₩n", (double)waited_time/served_customers);
        printf("아직 대기중인 고객 수 = %d₩n", customers - served_customers);
        printf("서버스하지 못한 시간의 총합 = %d₩n", 30 - actual_service);
}

// 시뮬레이션 프로그램
void main()
{
        int service_time[No_SERVICEMAN] = {0, 0, 0};

        clock = 0;
        while (clock < duration) {
                clock++;
                printf("현재시각 : %d₩n", clock);
                if (is_customer_arrived()) {
                        insert_customer(clock);
                }
                for (int k = 0; k < No_SERVICEMAN; k++) {
                        if (service_time[k] > 0) {
                                service_time[k]--;
                                actual_service++;
                        }
                        else {
                                service_time[k] = remove_customer();
                        }
                }

        }
        print_stat();
}
```

결과 출력

① 기존 설정

```
// 시뮬레이션에 필요한 여러가지 상태변수
int duration = 10;        //시뮬레이션 시간
double arrival_prob = 0.7;        // 하나의 시간 단위에 도착하는 평균 고객
int max_serv_time = 5;            // 하나의 고객에 대한 최대 서비스 시간
```

```
현재시각 : 1
고객 0이 1분에 들어옵니다. 서비스 시간은 3분입니다.
고객 0이 1분에 서비스를 시작합니다. 대기시간은 0분이었습니다.
현재시각 : 2
고객 1이 2분에 들어옵니다. 서비스 시간은 5분입니다.
고객 1이 2분에 서비스를 시작합니다. 대기시간은 0분이었습니다.
현재시각 : 3
고객 2이 3분에 들어옵니다. 서비스 시간은 3분입니다.
고객 2이 3분에 서비스를 시작합니다. 대기시간은 0분이었습니다.
현재시각 : 4
고객 3이 4분에 들어옵니다. 서비스 시간은 5분입니다.
고객 3이 4분에 서비스를 시작합니다. 대기시간은 0분이었습니다.
현재시각 : 5
현재시각 : 6
현재시각 : 7
고객 4이 7분에 들어옵니다. 서비스 시간은 5분입니다.
고객 4이 7분에 서비스를 시작합니다. 대기시간은 0분이었습니다.
현재시각 : 8
현재시각 : 9
고객 5이 9분에 들어옵니다. 서비스 시간은 2분입니다.
고객 5이 9분에 서비스를 시작합니다. 대기시간은 0분이었습니다.
현재시각 : 10
고객 6이 10분에 들어옵니다. 서비스 시간은 1분입니다.
고객 6이 10분에 서비스를 시작합니다. 대기시간은 0분이었습니다.
서비스 받은 고객 수 = 7
전체 대기 시간 = 0분
1인당 평균 대기 시간 = 0.000000분
아직 대기중인 고객 수 = 0
서버스하지 못한 시간의 총합 = 14
```

② 서비스하는 사람이 많을 때의 대기열과 서비스 진행을 확인하기 위해 설정을 수정

```
// 시뮬레이션에 필요한 여러가지 상태변수
int duration = 15;        //시뮬레이션 시간
double arrival_prob = 0.8;        // 하나의 시간 단위에 도착하는 평균 고객
int max_serv_time = 8;            // 하나의 고객에 대한 최대 서비스 시간
```

```
현재시각 : 1
고객 0이 1분에 들어옵니다. 서비스 시간은 5분입니다.
고객 0이 1분에 서비스를 시작합니다. 대기시간은 0분이었습니다.
현재시각 : 2
고객 1이 2분에 들어옵니다. 서비스 시간은 7분입니다.
고객 1이 2분에 서비스를 시작합니다. 대기시간은 0분이었습니다.
현재시각 : 3
고객 2이 3분에 들어옵니다. 서비스 시간은 4분입니다.
고객 2이 3분에 서비스를 시작합니다. 대기시간은 0분이었습니다.
현재시각 : 4
고객 3이 4분에 들어옵니다. 서비스 시간은 8분입니다.
현재시각 : 5
현재시각 : 6
고객 4이 6분에 들어옵니다. 서비스 시간은 2분입니다.
고객 3이 6분에 서비스를 시작합니다. 대기시간은 2분이었습니다.
현재시각 : 7
고객 4이 7분에 서비스를 시작합니다. 대기시간은 1분이었습니다.
현재시각 : 8
고객 5이 8분에 들어옵니다. 서비스 시간은 5분입니다.
현재시각 : 9
고객 6이 9분에 들어옵니다. 서비스 시간은 1분입니다.
고객 5이 9분에 서비스를 시작합니다. 대기시간은 1분이었습니다.
고객 6이 9분에 서비스를 시작합니다. 대기시간은 0분이었습니다.
현재시각 : 10
고객 7이 10분에 들어옵니다. 서비스 시간은 3분입니다.
고객 7이 10분에 서비스를 시작합니다. 대기시간은 0분이었습니다.
현재시각 : 11
고객 8이 11분에 들어옵니다. 서비스 시간은 2분입니다.
현재시각 : 12
현재시각 : 13
고객 9이 13분에 들어옵니다. 서비스 시간은 1분입니다.
```

고객 8이 13분에 서비스를 시작합니다. 대기시간은 2분이었습니다.

현재시각 : 14

고객 10이 14분에 들어옵니다. 서비스 시간은 1분입니다.

고객 9이 14분에 서비스를 시작합니다. 대기시간은 1분이었습니다.

고객 10이 14분에 서비스를 시작합니다. 대기시간은 0분이었습니다.

현재시각 : 15

고객 11이 15분에 들어옵니다. 서비스 시간은 5분입니다.

고객 11이 15분에 서비스를 시작합니다. 대기시간은 0분이었습니다.

서비스 받은 고객 수 = 12

전체 대기 시간 = 7분

1인당 평균 대기 시간 = 0.583333분

아직 대기중인 고객 수 = 0

서버스하지 못한 시간의 총합 = 17

(2) Customer가 없을 경우

```
double arrival_prob = 0; // 하나의 시간 단위에 도착하는 평균 고객
```

현재시각 : 1

현재시각 : 2

현재시각 : 3

현재시각 : 4

현재시각 : 5

현재시각 : 6

현재시각 : 7

현재시각 : 8

현재시각 : 9

현재시각 : 10

서비스 받은 고객 수 = 0

전체 대기 시간 = 0분

1인당 평균 대기 시간 = -nan(ind)분

아직 대기중인 고객 수 = 0

서버스하지 못한 시간의 총합 = 30

(3) 서비스하는 사람들이 서비스를 못하는 시간(service_idle_time)의 총합을 결과로서 출력

※ 코드의 설정과 출력 결과는 앞선 (1)과 (2)에서 확인 가능하다

① 서비스하는 사람이 3명일 때, 기존 설정

서버스하지 못한 시간의 총합 = 14

한 시각에 한 손님씩 입장하고 서비스 시간이 짧은데 반해 서비스하는 직원은 3명이기 때문에 손님의 대기가 없었고, 직원들이 쉬고 있는 시간이 많았다. 따라서 3명의 서비스 가능한 총 시간 30분 중에 서비스하지 못한 시간의 총합은 14분이었다.

② 서비스하는 사람이 3명일 때, 수정한 설정

서버스하지 못한 시간의 총합 = 17

손님이 자주 입장하고 대기시간 인당 서비스 시간 또한 길었기 때문에 대기인원이 많아 직원들이 더 많이 일했다. 따라서 직원 3명의 서비스 가능한 총 시간 45분 중에 서비스하지 못한 시간은 17분이었다.

③ Customer가 없을 때

10분 동안 customer가 없었으므로 직원들은 모두 10분 동안 서비스하지 못했다. 따라서 3명의 직원이 서비스하지 못한 총 시간은 30분이다.