2014년도 제51회 변리사 제2차 국가자격시험 문제지

교시	시험과목	시험시간	수험번호	성명
2교시	데이터구조론	120분		

【 문제-1 】 (30점)

다음 트리에 관한 질문에 답하시오.

(1) 2-3 트리와 AVL 트리의 정의, 구조, 성능을 각각 비교 설명하시오. (5점)

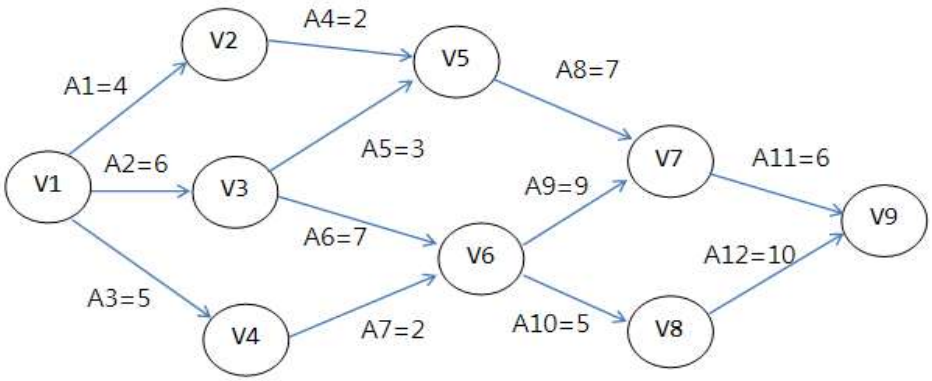
(2) 다음 정수가 하나씩 입력되어 2-3 트리가 만들어질 때의 모습을 단계별로 보이시오. (9점)
[15, 30, 10, 25, 45, 20, 12, 50, 60]

(3) 위 (2)번에서 최종 생성된 2-3 트리로부터 25, 20, 10, 45, 30, 60, 15를 순서대로 삭제할 때의 변화 상태를 각각 보이시오. (10점)

(4) 일반적으로 2-3 트리와 달리, 2-3-4 트리에서 노드 삽입시 후진분할 (backward split) 부담을 줄이기 위해 수행하는 작업이 무엇인지 설명하시오. (6점)

【 문제-2 】 (20점)

다음은 프로젝트의 공정(V)과 작업(A)들을 AOE(Activity on Edge) 네트워크로 표현한 것으로 V1에서 프로젝트가 시작되어 V9에서 완료된다. 간선의 가중치는 작업의 수행 시간을 나타내며 정점으로 들어오는 작업들이 모두 완료되어야 다음 작업을 수행할 수 있다.



(1) AOV(Activity on Vertex) 네트워크와 AOE(Activity on Edge) 네트워크의 정의와 사용 목적을 각각 쓰시오. (3점)

(2) 위 그래프의 모든 작업에 대해 가장 빠른 시작 시간(earliest time), 프로젝트를 지연시키지 않는 가장 늦은 시작시간(latest time), 임계도(criticality)를 각각 계산하시오. (9점)

(3) 위 그래프에서 프로젝트가 시작되어 완료될 때까지 임계경로(critical path), 프로젝트 최소완료시간과 임계작업(critical activity)을 쓰시오. (5점)

(4) 프로젝트 수행에 임계작업이 미치는 영향을 설명하고, 위 그래프에서 자신의 실행시간을 줄이면 전체 프로젝트 완료시간을 단축할 수 있는 작업을 모두 쓰시오. (3점)

【 문제-3 】 (30점)

다음과 같이 정수 배열이 주어졌을 때, 이를 오름차순으로 정렬하고자 한다.

[36 9 3 57 6 91 2 15 24 1]

(1) 다음 각 정렬 알고리즘을 사용하였을 때 배열 내용의 변화를 단계별로 나열하시오. (14점)

1) 선택정렬(Selection Sort)
2) 쉘정렬(Shell Sort)
3) 기수정렬(Radix Sort)
4) 퀵정렬(Quick Sort) - 주어진 배열 내에서 파티션을 시행해야 함

(2) 다음 각 정렬 알고리즘의 평균과 최악의 경우에 대한 시간 복잡도를 기술하고, 이를 바탕으로 각 알고리즘을 적용하기 적합한 사례를 논하시오. (8점)

1) 선택정렬(Selection Sort)
2) 쉘정렬(Shell Sort)
3) 기수정렬(Radix Sort)
4) 퀵정렬(Quick Sort)

(3) 다음 각 질문에 답하시오. (8점)

1) 주어진 배열이 오름차순으로 정렬되었을 때, 보간탐색(Interpolation Search) 방법으로 36을 탐색하는 과정을 단계별로 식을 사용하여 기술하시오.
2) 1)의 탐색 성능을 분석하고 보간 탐색이 우수한 성능을 보이는 데이터의 사례를 제시하고 설명하시오.

【 문제-4 】 (20점)

큐(Queue)와 덱(Deque)에 대해 다음 질문에 답하시오.

(1) 큐와 덱의 차이점을 설명하시오. (4점)

(2) 큐에 원소를 삽입하고 삭제하는 C 함수를 완성하시오. (8점)

```
typedef struct { ... } Element;     // 큐 원소 구조체
typedef struct _QueueNode {    // 큐 노드 구조체
        Element data;
        struct _QueueNode* link;
} QueueNode;

typedef struct {                    // 큐 구조체
        QueueNode* front;           // 큐의 맨 앞 노드에 대한 포인터
        QueueNode* rear;            // 큐의 맨 뒤 노드에 대한 포인터
} Queue;

void enqueue(Queue* queue, Element element) // 원소 삽입
{
        QueueNode* newNode = (QueueNode *)malloc(sizeof(QueueNode));
        newNode->data = element;
        newNode->link = NULL;
        if (queue->front == NULL) {
                                    1)
        }
        else {
                                    2)
        }
}

Element dequeue(Queue* queue) // 원소 삭제
{
        QueueNode* remNode;
        Element element;
        if (queue->front == NULL) { /* 오류 처리 */ }
        else {
                element = queue->front->data;
                remNode = queue->front;
                                    3)
                free(remNode);
                return element;
        }
}
```

(3) 덱에 원소를 앞에서 삽입하고 뒤에서 삭제하는 C 함수를 완성하시오. (8점)

```
typedef struct { ... } Element;     // 덱 원소 구조체
typedef struct _DeqNode {           // 덱 노드 구조체
        Element data;
        struct _DeqNode* rlink;
        struct _DeqNode* llink;
} DeqNode;

typedef struct {                    // 덱 구조체
        DeqNode* first;             // 덱의 맨 앞 노드에 대한 포인터
        DeqNode* last;              // 덱의 맨 뒤 노드에 대한 포인터
} Deque;

void insertFirst(Deque *deque, Element element) // 원소 삽입
{
        DeqNode* newNode = (DeqNode *)malloc(sizeof(DeqNode));
        newNode->data = element;
        if (deque->first == NULL) {
```

1)

```
                newNode->rlink = newNode->llink = NULL;
        }
        else {
```

2)

```
                newNode->llink = NULL;
        }
}

Element deleteLast(Deque *deque) // 원소 삭제
{
        DeqNode* remNode;
        Element element;
        if (deque->first == NULL) { /* 오류 처리 */ }
        else {
                element = deque->last->data;
                remNode = deque->last;
```

3)

```
                free(remNode);
                return element;
        }
}
```