

## 자료구조론

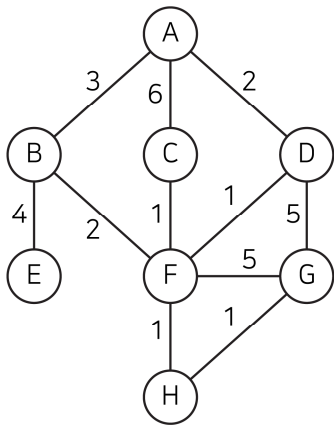
1. 배열로 구현한 최대 힙(max heap)에 키 값 <15, 10, 12, 8, 9, 7, 20>을 차례대로 삽입하였을 때, 값 7이 저장되어 있는 색인(index) 번호는? (단, 배열에 저장되는 키 값의 첫 번째 저장위치 색인 번호는 1이다)

① 4  
② 5  
③ 6  
④ 7

2. 비어 있는 이진 탐색 트리(binary search tree)에 키 값 <7, 5, 1, 8, 3, 6, 0, 9, 4, 2>를 순서대로 삽입한 후, 중위 순회(in-order traversal)한 결과는?

① 0 1 2 3 4 5 6 7 8 9  
② 0 2 4 3 1 6 5 9 8 7  
③ 7 5 1 0 3 2 4 6 8 9  
④ 9 8 6 4 2 3 0 1 5 7

3. 다음 그래프의 정점 A에서 시작하여 다른 모든 정점까지의 최단 경로를 찾고자 한다. 다익스트라(Dijkstra) 알고리즘을 사용하여 찾은 정점 G까지의 최단 경로를 바르게 나열한 것은?



① A → B → F → G  
② A → C → F → G  
③ A → D → F → G  
④ A → D → F → H → G

4. 다음 C 코드에서 밑줄 친 'x=x+3;'의 수행 빈도수는? (단, n은 0 이상의 정수이다)

```
void complexity(int n){
    int i, j, x=0;
    for (i=0; i<=n; i++) {
        for (j=i; j<=n; j++)
            x=x+3;
    }
}
```

①  $\frac{n(n-1)}{2}$   
②  $\frac{n(n+1)}{2}$   
③  $\frac{(n-1)(n+1)}{2}$   
④  $\frac{(n+1)(n+2)}{2}$

5. 다음 C 코드로 표현한 알고리즘의 시간 복잡도는?

```
for (i=1; i<n; i++) {
    MaxData=arr[i];
    for (j=i-1; j>=0; j--) {
        if (arr[j]>MaxData)
            arr[j+1]=arr[j];
        else
            break;
    }
    arr[j+1]=MaxData;
}
```

①  $O(\log n)$   
②  $O(n \log n)$   
③  $O(n^2)$   
④  $O(n^3)$

6. 다음 자료에 대하여 퀵 정렬(quick sort)로 오름차순 정렬을 하고자 한다. 기준값인 피벗(pivot)으로 정렬 대상 자료의 첫 번째 원소를 사용할 때, 두 번째 분할(partition) 연산 수행 후의 결과는?

< 27, 5, 37, 4, 61, 11, 59, 15, 48, 20 >

- ① 4 5 11 20 15 27 59 61 48 37  
 ② 4 5 11 15 20 27 59 61 48 37  
 ③ 4 5 11 15 20 27 59 37 48 61  
 ④ 4 5 11 20 15 27 59 37 48 61

7. 다음 재귀함수를 호출하는 C 프로그램의 실행 결과는?

```
#include <stdio.h>
void RecursiveFunc(int n) {
    if (n <= 0) {
        printf("R-%d ", n);
        return;
    }
    printf("R-%d ", n);
    if (n%2 == 0)
        RecursiveFunc(n-2);
    else
        RecursiveFunc(n-1);

    printf("R-%d ", n);
}

int main(void)
{
    RecursiveFunc(3);
    return 0;
}
```

- ① R-3 R-2 R-0  
 ② R-3 R-1 R-0 R-2 R-3  
 ③ R-3 R-2 R-0 R-2 R-3  
 ④ R-3 R-2 R-1 R-2 R-3

8. 다음은 병합 정렬(merge sort) 알고리즘을 구현한 파이썬 코드이다. (가), (나)에 들어갈 내용을 바르게 연결한 것은?

```
sorted = []

def merge_sort(arr, left, right):
    if left < right:
        middle = (left + right) // 2
        merge_sort(arr, left, middle)
        merge_sort(arr, middle+1, right)
        merge(arr, left, middle, right)

def merge(arr, left, middle, right):
    global sorted
    i = left
    j = middle + 1
    k = (가)
    while i <= middle and (나):
        if arr[i] <= arr[j]:
            sorted.insert(k, arr[i])
            i = i + 1
        else:
            sorted.insert(k, arr[j])
            j = j + 1
        k = k + 1

    if i > middle:
        sorted[k:k+right-j+1] = arr[j:right+1]
    else:
        sorted[k:k+middle-i+1] = arr[i:middle+1]

    arr[left:right+1] = sorted[left:right+1]
```

(가)

(나)

- |          |            |
|----------|------------|
| ① left   | j <= right |
| ② left   | k <= right |
| ③ middle | j <= right |
| ④ middle | k <= right |

9. 다음은 C 언어 구조체를 이용하여 배열과 단순 연결 리스트로 큐를 표현한 것이다.

(가) 배열을 이용한 큐 구조체	(나) 연결 리스트를 이용한 큐 구조체
<pre>typedef struct _Queue {     int front;     int rear;     int numOfData;     int queArr[50]; } QType;</pre>	<pre>typedef struct _node {     int data;     struct _node *next; } Node; typedef struct _Queue {     Node *front;     Node *rear;     int numOfData; } QType;</pre>

아래의 <조건>에 따라 (가), (나) 각각에 대해 Enqueue 함수를 세 번 호출하고 Dequeue 함수를 한 번 호출한 후, 큐에 할당된 메모리 공간의 크기(byte)를 계산한 값으로 바르게 연결한 것은?

- <조 건>
- 큐 변수는 QType q; 형태로 선언한다.
  - Enqueue 함수는 큐의 front에 데이터를 추가하는 함수로 함수 원형은 void Enqueue(QType \*q, int value)이다.
  - Dequeue 함수는 큐의 rear에서 데이터를 삭제하는 함수이며, 함수 원형은 void Dequeue(QType \*q)이다.
  - int형 자료의 경우 4바이트, 포인터형 자료의 경우 8바이트의 메모리 공간을 사용한다.
  - Enqueue와 Dequeue 함수에서는 배열 및 연결 리스트 특성에 맞게 필요한 메모리를 할당·회수하는 동작이 적절히 일어난다.
  - 연결 리스트에서 삭제 연산은 메모리 회수를 포함한다.
  - 구조체 변수를 메모리에 할당할 때 바이트 패딩 없이 할당한다.

(가)	(나)
① 24	44
② 24	56
③ 212	44
④ 212	56

10. 다음 C 프로그램의 실행 결과는? (단, 2차원 배열의 경우 행 우선으로 메모리에 저장된다)

```
#include <stdio.h>
int main(void)
{
    int i, j, test[3][4];
    for (i=0; i<3; i++)
        for (j=0; j<4; j++)
            test[i][3-j]=i*4+j;
    printf("%d ", *((test+1)+5));
}
```

- |     |      |
|-----|------|
| ① 5 | ② 6  |
| ③ 9 | ④ 10 |

11. 다음은 구조체 배열에서 최댓값을 찾아 출력하는 C 프로그램이다. (가), (나)에 들어갈 내용을 바르게 연결한 것은?

```
#include <stdio.h>
typedef struct _data {
    int value;
} DataType;
DataType *max(DataType *a) {
    int i, max=a[0].value, index=0;
    for (i=0; i<5; i++)
        if (max<a[i].value) {
            max=a[i].value;
            index=i;
        }
    return (가);
}

int main(void)
{
    int i;
    DataType data[5], *result;
    for (i=0; i<5; i++)
        scanf("%d", &data[i].value);

    result=max(data);
    printf("max = %d", (나));
    return 0;
}
```

(가)	(나)
① &a[index]	result->value
② &a[index]	result.value
③ a[index]	result->value
④ a[index]	(*result).value

12. 다음 전위 표기식을 후위 표기식으로 변환한 것은?

/ + a b - c d

- ① a b + c d - /
- ② a b c d + - /
- ③ a b / c d + -
- ④ a b c d / + -

13. 다음은 단순 연결 리스트를 이용하여 스택 자료구조를 구현한 파이썬 코드이다. (가), (나)에 들어갈 내용을 바르게 연결한 것은?

```
class Node:
    def __init__(self, data, next):
        self.data = data
        self.next = next
```

```
class MyStack:
    def __init__(self):
        self.top = None
```

```
def push(self, data):
```

(가)

```
self.top = n
```

```
def pop(self):
```

```
if self.top != None:
```

```
n = self.top
```

(나)

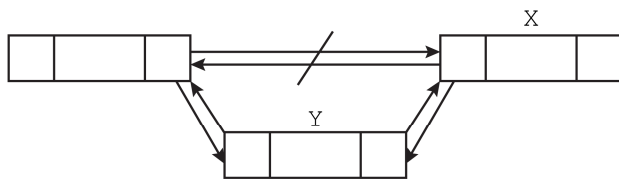
```
return n.data
```

(가)

(나)

- |                            |                          |
|----------------------------|--------------------------|
| ① n = Node(data, None)     | self.top = n.next.next   |
| ② n = Node(data, None)     | self.top = self.top.next |
| ③ n = Node(data, self.top) | self.top = n.next.next   |
| ④ n = Node(data, self.top) | self.top = self.top.next |

14. 그림과 같은 이중 연결 리스트(doubly linked list)에서 노드 X의 왼쪽에 노드 Y를 삽입할 때, 연산의 순서를 바르게 나열한 것은? (단, Llink는 왼쪽 노드를 가리키는 포인터 변수이고, Rlink는 오른쪽 노드를 가리키는 포인터 변수이다)



- a. X → Llink → Rlink = Y  
b. Y → Llink = X → Llink  
c. X → Llink = Y  
d. Y → Rlink = X

- ① a → c → b → d  
② a → c → d → b  
③ b → c → a → d  
④ b → d → a → c

15. 최소 1개 이상의 노드로 이루어진 원형 연결 리스트(circular linked list)에서 포인터 L이 리스트의 첫 번째 노드를 가리킨다고 할 때, 데이터의 값이 x인 노드를 탐색하여 반환하는 C 코드는?

```
① ListNode *searchCList(ListNode *L, int x) {
    ListNode *p = L;
    while (p != L) {
        if (p->data == x) return p;
        p = p->link;
    }
    return NULL;
}
```

```
② ListNode *searchCList(ListNode *L, int x) {
    ListNode *p = L;
    while (p->link != L) {
        if (p->data == x) return p;
        p = p->link;
    }
    return NULL;
}
```

```
③ ListNode *searchCList(ListNode *L, int x) {
    ListNode *p = L;
    do {
        if (p->data == x) return p;
        p = p->link;
    } while (p != L);
    return NULL;
}
```

```
④ ListNode *searchCList(ListNode *L, int x) {
    ListNode *p = L;
    do {
        if (p->data == x) return p;
        p = p->link;
    } while (p->link != L);
    return NULL;
}
```

16. 0이 아닌 원소의 개수가 7개인  $9 \times 12$  희소행렬에 대한 설명으로 옳지 않은 것은?

- ① 행렬을 그대로 선형 순차 리스트로 표현하여 2차원 배열로 구현하면, 108개의 배열 원소에 대한 공간이 필요하다.
- ② 0이 아닌 원소들에 대한 <행 번호, 열 번호, 값>의 쌍을 2차원 배열로 구현하면 14개의 배열 원소에 대한 공간이 필요하다.
- ③ 행렬을 그대로 선형 순차 리스트로 표현하여 2차원 배열로 구현하는 것은 실제로 사용하지 않는 공간이 많아 기억 공간의 활용도가 떨어진다.
- ④ <행 번호, 열 번호, 값>의 쌍을 2차원 배열로 구현한 희소행렬에 대하여 전치 행렬을 구하는 방법의 시간 복잡도는 최악의 경우,  $O(\text{열의 수}^2 \times \text{행의 수})$ 이다.

17. 다음 C 코드에서 `heapify(a, 7)`을 수행한 후 배열 `a[]`의 내용은? (단, 모든 배열 원소값은 양의 정수로 한정하고, -1은 연산 대상이 아니며, 0은 삭제된 값을 나타낸다)

```
int a[8] = {-1, 1, 2, 3, 6, 4, 7, 5};

int heapify(int *a, int n) {
    int item=a[1], temp=a[n];
    int i=1, j=2;

    a[n]=0, n=n-1;
    while (j<=n) {
        if (j<n && a[j]>a[j+1]) j++;
        if (temp<a[j]) break;
        else {
            a[i]=a[j];
            i=j;
            j=j*2;
        }
    }
    a[i]=temp;
    return item;
}
```

- ① 

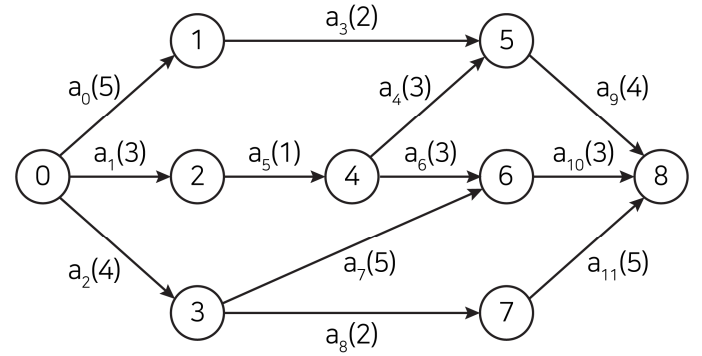
0	1	2	3	4	5	6	7
-1	2	3	4	6	5	7	0
- ② 

0	1	2	3	4	5	6	7
-1	2	4	3	6	5	7	0
- ③ 

0	1	2	3	4	5	6	7
-1	2	3	4	5	6	7	0
- ④ 

0	1	2	3	4	5	6	7
-1	2	4	3	5	6	7	0

18. 다음 그래프는 AOE(Activity on Edge) 네트워크를 보여 준다. 정점의 숫자는 공정을 나타내고, 간선 위에 있는  $a_i$ 는 작업을, 괄호 안의 숫자는 그 작업을 수행하는 데 필요한 최소한의 시간을 나타낼 때, 임계 경로의 길이는?



- ① 10
- ② 11
- ③ 12
- ④ 13

19. 키 값 <2, 6, 3, 4, 5, 1, 7, 8>을 순서대로 삽입해서 레드 블랙 트리를 생성할 때, 단일 회전과 이중 회전이 발생하는 횟수를 바르게 나열한 것은?

	단일 회전	이중 회전
①	1	1
②	1	2
③	2	1
④	2	2

20. 다음 이진 트리의 전체 노드 수를 계산하는 C 코드에서 (가)에 들어갈 내용은? (단, n은 최초에 이진 트리 내 루트 노드를 가리키는 포인터 변수이며, TNode는 이진 트리의 노드 구조를 나타낸다)

```
typedef struct Treenode {
    char data;
    struct Treenode *left;
    struct Treenode *right;
} TNode;

int count_node(TNode *n) {
    if (n==NULL) return 0;
    return (가) + count_node(n->right);
}
```

- ① 1  
 ② n->data  
 ③ count\_node(n->left)  
 ④ 1+count\_node(n->left)
21. 다음 탐색 알고리즘을 이용하여 아래 리스트에서 키(key) 34를 찾을 경우, 필요한 반복 연산 횟수는? (단, low 변수에는 0, high 변수에는 15가 전달된다)

```
int a_search(int list[], int key, int low, int high) {
    int middle;
    while (low <= high) {
        middle = (low + high) / 2;
        if (key == list[middle])
            return middle;
        else if (key > list[middle])
            low = middle + 1;
        else
            high = middle - 1;
    }
    return -1;
}
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	6	11	13	18	20	22	27	29	30	34	38	41	42	45	47

- ① 3  
 ② 4  
 ③ 5  
 ④ 6

22. 버킷(bucket)이 13개이고 버킷당 슬롯(slot)이 1개인 해시 테이블에 키 값 <7, 20, 3, 19, 45>를 차례대로 저장하고자 한다. 다음 <조건>에 따라 아래 <해시 함수>를 이용하여 작동하는 이중 해싱(double hashing)에서, 마지막 키 값 45의 저장 위치는?

— <조 건> —

- 버킷의 인덱스는 0부터 시작한다.
- $h(x)$ 는 첫 번째 조사 위치를 결정하는 기본 해시 함수이다.
- $f(x)$ 는 충돌 발생 시 조사 위치 간격을 결정하는 함수이다.
- 하나의 키 값에 대한 해시 함수 연산 중 충돌이 1번 이상 발생할 경우  $i$ 번째 충돌에 대응하여  $h_i(x)$ 를 적용한다.

— <해시 함수> —

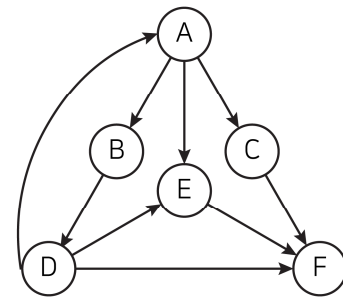
$$h(x) = x \bmod 13$$

$$f(x) = x \bmod 11$$

$$h_i(x) = (h(x) + i \times f(x)) \bmod 13$$

- ① 5  
 ② 7  
 ③ 8  
 ④ 11

23. 다음 그래프에 대한 설명 중 옳은 것만을 모두 고르면?



- ㄱ. 가장 긴 단순 경로의 길이는 4이다.  
 ㄴ. 진입차수가 진출차수보다 큰 정점은 1개이다.  
 ㄷ. 완전 그래프가 되기 위해 추가로 필요한 간선의 수는 10개이다.  
 ㄹ. 간선 <A, B>를 제거한 후 위상정렬을 수행하면 B, D, A, E, C, F 또는 B, D, A, C, E, F 순으로 정점이 나열된다.

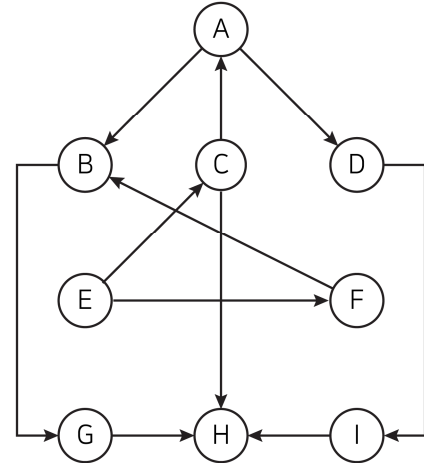
- ① ㄱ, ㄹ  
 ② ㄴ, ㄷ  
 ③ ㄱ, ㄴ, ㄹ  
 ④ ㄴ, ㄷ, ㄹ

24. 무방향 그래프(undirected graph) G의 인접 행렬(adjacent matrix) 표현을 C 언어의 2차원 배열을 이용하여 다음과 같이 구현했다. 그래프 G의 정점(vertex)이 0, 1, 2, 3, 4, 5, 6의 숫자로 표현될 때, 무방향 그래프 G의 그림으로 옳은 것은?

```
int G[7][7] = {
    { 0, 1, 1, 0, 0, 0, 0 },
    { 1, 0, 1, 1, 0, 0, 0 },
    { 1, 1, 0, 1, 0, 1, 0 },
    { 0, 1, 1, 0, 0, 1, 0 },
    { 0, 0, 0, 0, 0, 1, 0 },
    { 0, 0, 1, 1, 1, 0, 1 },
    { 0, 0, 0, 0, 0, 1, 0 }
};
```

- ①
- ②
- ③
- ④

25. 다음 방향 그래프(directed graph)에서 노드 A부터 시작하여 깊이 우선 탐색과 너비 우선 탐색을 수행하는 경우, 여섯 번째로 방문하는 노드의 쌍을 바르게 연결한 것은? (단, 노드 A는 첫 번째 방문 노드이고, 인접 노드가 다수일 경우 방문 순서는 알파벳순으로 하며, 탐색 과정에서 도달하지 못하는 경로는 고려하지 않는다)



깊이 우선 탐색

너비 우선 탐색

- |   |   |   |
|---|---|---|
| ① | C | D |
| ② | D | I |
| ③ | H | H |
| ④ | I | H |