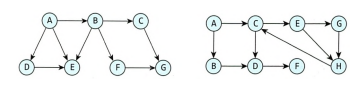
4장 연습문제

7번, 8번, 13번, 18번, 19번, 20번, 21번(인터넷검색)

5장 연습문제

1번, 3번

**7. 다음의 방향 그래프에 대해 DFS 기반 위상 정렬을 수행하라.**



DFS – 깊이 우선 탐색

위상 정렬 – 의존성 그래프

- 사이클이 존재하지 않는 방향 그래프에서만 가능

- 진입 차수가 0인 임의의 정점을 시작 정점으로 선택

- 깊이 우선 탐색으로 그래프의 정점들을 방문

- 더 이상 갈 수 있는 정점이 없으면 다른 진입 차수가 0인 정점

- 위상 정렬의 시간 복잡도는 O(V + E) // 정점 + 간선

1. A-B-C-F-G-D-E

2. A-B

- C를 실행하기 위해선 H의 작업이 끝나야 하는데 그러지 못했다

**8. 위 문제의 방향 그래프에 대해 축소 정복 전략의 위상 정렬 알고리즘 4.4를 수행하라.**

- 진입 차수가 0인 임의의 정점 선택

- 선택된 정점을 삭제하고 진출하는 모든 간선들도 삭제하며 진입 차수를 갱신한다

1. A-B-C-D-E-F-G

2. A-B

- C 정점의 진입 차수가 0이 되지 못함

**13. 1부터 n까지 오름차순으로 이루어진 숫자 n一1개로 이루어진 리스트 A［0..n—2］가 있다. 즉, 이 리스트에는 1부터 n 사이의 하나의 숫자가 빠져 있다. 빠진 숫자를 찾는 효율적인 알고리즘을 작성하고 시간 복잡도를 설명하라.**

- 이진 탐색 알고리즘

- 정렬되어 있는 리스트에서 탐색 범위를 절반씩 좁혀가며(로그 시간의 시간 복잡도) 데이터를 탐색하는 방법

- 시작점, 끝점, 중간점(소수점 이하 제거)을 이용하여 탐색 범위를 설정

- 반드시 리스트가 정렬되어 있어야 하며 데이터의 삽입이나 삭제가 빈번한 응용에는 적합하지 않음

- 시간 복잡도 = O(logN)

**18. 알고리즘 4.11의 quick\_select()를 이용해 리스트 A=[12, 5, 7, 9, 18, 3, 8]에서 중앙값(median)을 찾는 과정을 보여라.**

4번째로 큰 값이 중간 값 🡪 k = 4

임의의 숫자를 피벗으로 정하고 피벗보다 큰(오른쪽) 리스트와 작은(왼쪽) 리스트로 파티션한다

피벗의 인덱스가 k와 같은 경우 그대로 그 인덱스의 값을 리턴

피벗의 인덱스가 k보다 작은 경우 피벗의 인덱스+1부터 마지막 인덱스까지 다시 파티션

피벗의 인덱스가 k보다 큰 경우 첫 번째 인덱스부터 피벗의 인덱스-1까지 다시 파티션

**19. 알고리즘 4.11의 quick\_select() 알고리즘을 반복구조로 기술하라.**

def quick\_select\_iter(a, left, right, k):

    while left <= right:

        pos = partition(a, left, right)

        if (pos == k-1):

            return a[pos]

        elif (pos > k-1):

            right = pos-1

        else:

            left = pos+1

**20. 알고리즘 4.11에 대한 최악의 입력 상황을 구체적으로 만들어 보고, 왜 최악의 입력인지를 설명하라.**

정렬이 되어 있는 리스트에서 가장 작거나 큰 수를 찾아야 할 때

피벗을 계속해서 최악으로 설정하게 되면 퀵 셀렉트를 n번 진행해야 해서

O(n^2)의 시간 복잡도를 지니는 최악의 입력이 된다

**21. 피벗을 이용해 리스트를 나누는 다른 알고리즘들도 있다. 알고리즘 4.12는 호어(Hoare) 분할인데, 로무토(Lomuto) 분할이라는 방법도 있다. 이 방법을 조사하고 두 방법의 장단점을 비교하라. Lomuto 분할을 구현하고, 이를 이용해 알고리즘 4.11을 테스트해 보라.**

호어 분할이 로무토 분할보다 스왑을 3배 이상 덜 발생시키기 때문에 호어 분할이 더 유리하다

로무토 분할은 구현이 상대적으로 쉽지만 이미 정렬된 배열에서도 스왑을 끝까지 발생시켜 시간복잡도가 O(n^2)이 된다

def partition(A, left, right):

    x = A[right]

    i = left - 1

    for j in range(left, right):

        if A[j] <= x:

            i += 1

            A[i], A[j] = A[j], A[i]

    A[i+1], A[right] = A[right], A[i+1]

    return i + 1

**1. n개의 항목으로 이루어진 리스트에서 가장 작은 항목을 찾는 문제를 해결하려고 한다. 물음에 답하라.**

(1) 이 문제에 대한 분할 정복 알고리즘을 작성하라.

전체 리스트를 같은 크기인 2개의 부분 리스트로 분할한다 – 항목이 하나가 될 때까지 반복

항목이 하나가 남은 후 병합을 하는 과정에서 비교연산을 이용해 더 작은 값을 해답으로 내놓는다

최종적으로는 가장 작은 값이 남는다

(2) 이 알고리즘의 복잡도를 순환 관계식으로 나타내라. 단, 킷값의 비교 연산을 기본 연산으로 사용하라.

T(n) = 2T(n/2)+1

(3) 이 식을 연속 대치법과 마스터 정리를 이용해 각각 풀어라.

a = 2, b = 2, d = 0

2 > 2^0

= O(nlog2) = O(n)

(4) 이 알고리즘과 억지 기법 알고리즘을 비교하라.

억지 기법은 되는대로 이리저리 해보는, 몸으로 때우는 알고리즘이고 분할 정복은 Divide, Conquer, Combine 세 가지 과정을 거쳐야 하는 알고리즘이다

분할 정복 알고리즘은 정렬이나 탐색과 같은 많은 중요한 문제에서 상당히 효과적이지만, 항상 억지 기법보다 더 효율적인 건 아니다

억지 기법 외에는 아직까지 해결책이 알려지지 않은 문제도 많다

**3. 마스터 정리를 이용해 다음 복잡도 함수의 점근적 표기를 구하라.**

(1) T(n) = 4T(n/2)+n, T(1) = 1

a = 4, b = 2, d = 1

a > b^d

= O(n^log4) = O(n^2)

(2) T(n) = 8T(n/4)+n^2, T(1) = 1

a = 8, b = 4, d = 2

a < b^d

= O(n^2)

(3) T(n) = T(n/2)+n^3, T(1) = 1

a = 1, b = 2, d = 3

a < b^d

= O(n^3)