**2022학년도 1학기 중간과제물(온라인제출용)**

**교과목명 : 알 고 리 즘**

**학 번 : 202234 - 363659**

**성 명 : 이 민 규**

**연 락 처 : 010 - 6246 - 1458**

**과제유형(공통형/지정형) : 공통형(과제물 A형)**

-목차-

1. 알고리즘 설계기법 설명

2. 성능 관점 나열

3. 점화식과 폐쇄형

4. 퀵 정렬의 배열

5. 합병 정렬의 수행 과정

6. 힙 정렬 과정의 결과

**1. 대표적인 3가지의 알고리즘 설계 기법이 적용된 문제들을 모두 나열하고, 해당 문제의**

**정의 / 개념에 대해서 간단히 설명하시오.**

-> **분할정복방법 (divide-and conquer)**

문제를 자체 형태로 해결할 수 없을 때 문제를 분할하여 해결하는 방법

이진 탐색

정렬이 돼어있는 데이터를 효과적으로 서치할 수 는 방법 (정렬이 되어있는 데이터만 가능)

분할 : 배열의 중간값을 기준으로 두개로 나눔, 탐색키와 중간값이 같으면 분할을 종료

정복 : 탐색키가 중간값보다 작으면 왼쪽배열을 대상으로 이진탐색, 크면 오른쪽배열을 대상으로

이진탐색 호출

결합 : 직접 반환

합병 정렬

하나로 뭉쳐진 리스트를 두개의 똑같은 크기로 나누고 나누어진 리스트를 정렬 후 똑같은 크기로

다시 나누고 정렬하는 것은 반복하는 방법

분할 : 입력 배열을 똑같은 크기로 나눈다

정복 : 부분 배열을 정렬하고, 부분 배열의 왼쪽이 오른쪽보다 크기가 작지않으면 순환 호출을하여 분할 정복방법을 반복

결합 : 정렬된 나누어진 배열들을 하나의 배열로 결합

퀵 정렬

원소 피봇을 기준으로 주어진 배열을 두 개로 나누고 나누어진 배열에 대해 퀵 정렬을 반복하는 방법

분할 : 원소 피봇을 기준으로 두개로 분할

정복 : 나누어진 배열에 대해 퀵정렬 적용

결합 : 불필요

선택 문제

숫자들을 오름차순으로 정렬한 후, K번째 숫자를 찾음

분할 : 숫자는 정렬돼있지 않기 때문에 임의의 피봇을 선택을하고 피봇보다 작은 숫자는 왼쪽, 큰.

숫자는 오른쪽으로 분할

정복 : 나누어진 숫자를 통하여 속해있지 않은 배열은 제외하고 숫자가 포함된 배열에서만 작업을

반복하여 K번째 숫자를 찾음

결합 : 불필요

(다음페이지에 이어서...)

**동적프로그래밍방법(dynamic programming)**

큰 문제를 작은 문제로 나누어가면서 풀어가는 방법,

분할정복과 비슷하다고 생각할 수 있지만 작은 문제가 중복이 발생하는지 안하는지의 가장 큰

차이점을 갖고있음

피보나치 수열 : 피보나치수열이란 0번째 항을 0으로 1번째 항을 1로 두고 2항은 앞의 0과 1을 더한

수를 2항으로 둔다. 3항은 1항과 2항을 더한 수를 3항으로 두는 방법을 반복하는 수열

기본적으로 재귀적인 피보나치 수열이 있지만 동적프로그래밍으로 피보나치 수열을 구하게 되면

실제값을 구하게 되는 방법을 사용을 안해도 되는 방법이므로 성능 향상에 도움이 됨

연쇄 행렬 곱셈 문제 : n개의 행렬을 연쇄적으로 곱셈을 하는 방법으로써 여러가지의 곱셈이 존재,

여러가지의 곱셈이 존재하므로서 곰셈의 횟수가 틀려짐

n개의 행렬을 최적으로 곱하는 순서는 n개의 행렬의 임의의 부분집합을 곱하게되는 최적의 순서를

포함하고, 부분수열을 이용한다는 가장 중요한 핵심을 갖음

스트링 편집 거리 문제 : 논문, 보고서 등의 표절검사, DNA 염기 서열의 유사도 검사 등에 사용

두 문자열의 유사도는 원래 스트링과 비교스트링이 있을 때 원래 스트링을 비교 스트링으로

변환하는데 필요한 삽입, 삭제, 대치 연산의 최소 비용을 구함으로써 판단

다익스트라 : 최단 경로를 검색해주는 알고리즘으로써 1차원으로 사용

모든 장점 간의 최단 경로(플로이드) : 모든 정점에서 다른 정점으로 가는 최단 경로를 모두 구해주는

방법이기 때문에 2차원으로 사용, 다익스트라는 1차원으로 사용하기 때문에 한계점이 있지만

플로이드는 2차원으로 사용을 하기 때문에 다익스트라에서의 한계점을 보완

저울 문제 : 물체의 무게와 추의 무게가 정수라고 가정했을 때 물체를 양팔 저울에 올릴 수 있는지

확인하는 문제

**욕심쟁이방법(greedy)**

최적성의 원리, 최적화 문제, 전체적인 최적해를 구하지 못할 수 있음

동전 거스름돈 문제 : 거스름돈의 액수를 초과하지 않는다는 조건으로 고객에게 거스름돈을 줄 때

최소한의 동전으로 거슬러주는 방법

배낭 문제 : 배낭의 정해진 용량이 초과되지 않는다는 조건에서 가방에 물건을 넣었을 때 물건의

이익의 합이 최대가 되도록 넣는 방법

최소 신장 트리(프림) : 임의의 한 정점으로부터 시작을 하여 가중치가 가장 작은 간선을 하나씩

선택하는 방법

(다음페이지에 이어서...)

최소 신장 트리(크루스칼) : 간선이 하나도 없는 상태로 시작을 하여 가중치가 가장 작은 간선부터

하나씩 사이클을 만들지 않으면서 추가하는 방법을 반복

최단 경로(데이크스트라) : 가중치의 방향 그래프에서 두 정점을 연결하는 경로 중 가중치의 합이

가장 작은 경로

작업 스케줄링 문제 : 작업 스케줄링 시간을 오름차순으로 정렬 후 기계에 최초 작업을 할당, 작업이

중복되면 새로운 기계를 만들어서 작업을 시작,마지막으로 몇 개의 기계가 사용됐는지 뽑아주는 방법

허프만 코딩(허프만 트리 생성) : 문장에서 각 단어가 출현하는 빈도수를 구하여 서로 다른 길이의

부호를 정해주는 방법, 단어늬 빈도나 확률을 이용해서 통계적으로 나타내는 방법

**2. 다음은 입력 크기 n에 대한 빅오 함수들이다.**

**알고리즘의 성능 관점에서 가장 나쁜 것부터 차례대로 나열하시오.**

**<O(logn) O(2^n) O(n) O(n^3) O(1) O(n^2) O(nlogn)>**

-> O(2^n) O(n^3) O(n^2) O(nlogn) O(n) O(logn) O(1)

**3. 다음 4가지 경우에 해당하는 점화식과 폐쇄형을 쓰시오.**

**점화식 폐쇄형**

1) 이진탐색 Θ(1), n = 1

T(n) T(n) = Θ(logn)

T(n / 2) + Θ(1), n ≥ 2

2) 퀵 정렬의 최악의 경우

Θ(1), n = 1

T(n) T(n) = Θ(n^2)

T(n - 1) + Θ(n), n ≥ 2

3) 합병 정렬

Θ(1), n = 1

T(n) T(n) = Θ(nlogn)

2T( n / 2) + Θ(n), n ≥ 2

4) 퀵 정렬의 최선의 경우

Θ(1), n = 1

T(n) T(n) = Θ(nlogn)

2T( n / 2) + Θ(n), n ≥ 2

**4. 주어진 배열에 대해서 퀵 정렬의 분할 함수 Partition()을 한 번 적용한 후의 결과**

**배열을 구하시오.(단, A[0]이 피벗이다.)**

**A[] = {35 50 25 40 70 20 45 55 30 10}**

Left -> <- Right

피벗 35 50 25 40 70 20 45 55 30 10

- 왼쪽에서 피벗보다 큰 값을 선택(50)

- 오른쪽에서 피벗보다 작은 값을 선택(10)

피벗 35 10 25 40 70 20 45 55 30 50

- 선택한 값의 위치를 서로 변경(50 <-> 10)

피벗 35 10 25 40 70 20 45 55 30 50

- 왼쪽에서 피벗보다 큰 값을 선택(40)

- 오른쪽에서 피벗보다 작은 값을 선택(30)

피벗 35 10 25 30 70 20 45 55 40 50

- 선택한 값의 위치를 서로 변경(40 <-> 30)

피벗 35 10 25 30 70 20 45 55 40 50

- 왼쪽에서 피벗보다 큰 값을 선택(70)

- 오른쪽에서 피벗보다 작은 값을 선택(20)

피벗 35 10 25 30 20 70 45 55 40 50

- 선택한 값의 위치를 서로 변경(70 <-> 20)

20 10 25 30 35 70 45 55 40 50

- 피벗(35)을 기준으로 왼쪽은 피벗보다 작고 오른쪽은 피벗보다 큰 수인 것을 확인하고 종료

(5번은 다음페이지에...)

**5. 주어진 데이터에 대해서 합병 정렬의 전체적인 수행 과정을 나타내시오.**

**합병정렬이란 일정한 크기로 배열을 나눠주고 오름차순으로 정렬을 하면서 합병을 해주는 방법**

**A[] = {35 50 25 40 70 20 45 55}**

35 50 25 40 70 20 45 55

(1)

35 50 25 40 70 20 45 55 - 개수를 맞춰서 두개의 배열로 나눠줌

(2) (8)

35 50 25 40 70 20 45 55 - 개수를 맞춰서 4개의 배열로 나눠줌

(3) (5) (9) (11)

35 50 25 40 70 20 45 55 - 각각으로 나눠줌

(4) (6) (10) (12)

35 50 25 40 20 70 45 55 - 오름차순 정렬을 하고 4개 배열로 합병

(7) (13)

25 35 40 50 20 45 55 70 - 오름차순 정렬을 하고 2개의 배욜로 합병

(14)

20 25 35 40 45 50 55 70 - 최종적으로 오름차순 정렬을 한 후에 합병

**6.주어진 데이터에 대해서 다음 조건데 따라 힙 정렬 과정의 결과를 표현하시오. (다음 페이지에 답)**

**A[] = {35 50 25 40 70 20 45 55}**

**\* 주어진 입력 배열의 각 원소에 대해 힙에서의 삽입 과정을 반복하는 알고리즘을 통해 초기 힙을 구축하고 표현한다.**

35와 50을 비교한 후 큰 값을 올림 =>50

50

35

25 추가후 50과 비교 => 50이 더 크기때문에 그대로 진행

50

35 25

35의 왼쪽 자식노드로 40을 추가 후 35와 비교 => 40이 더 크기때문에 35와 40을 변경

50

40 25

35

(다음페이지에 이어서...)

70을 40의 오른쪽 자식노드로 추가 후 40과 비교 => 70이 더 크기 때문에 40과 70을 변경

50

70 25

35 40

70과 50을 비교 => 70이 더 크기 때문에 50과 70을 변경

70

50 25

35 40

20을 25의 왼쪽자식 노드로 추가 후 25와 비교 => 25가 더 크기때문에 그대로 진행

70

50 25

35 40 20

25의 오른쪽 자식노드로 45를 추가 후 25와 비교 => 45가 더 크기 때문에 25와 변경

70

50 45

35 40 20 25

마지막 노드로 55를 35의 왼쪽자식노드로 추가 후 35와 비교 => 55가 더 크기 때문에 35와 변경

70

50 45

55 40 20 25

35

(다음페이지에 이어서...)

55와 50 비교 => 55가 크기 때문에 변경

70

55 45

50 40 20 25

35

**\* 힙 정렬의 두 번째 과정(최댓값 삭제 + 힙의 재구성)을 한 번 그리고 두 번 수행한 후의 힙을 각각 표현한다**

**<첫 번째>**

최댓값 삭제

- 가장 큰 값인 70과 마지막 노드인 35를 변경한다

35

55 45

50 40 20 25

70 (삭제된노드)

힙의 재구성

35

55 45

50 40 20 25

70 (삭제된노드)

- 35와 다음 줄에서의 큰 값을 비교하여 위치를 변경(35 -> 55)

55

35 45

50 40 20 25

70 (삭제된노드)

(다음페이지에 이어서...)

- 35와 다음 줄에서의 큰 값을 비교하여 위치를 변경 (35 -> 50)

55

50 45

35 40 20 25

70(삭제된 노드)

**<두 번째>**

최댓값 삭제

첫 번째와 동일한 방법으로 한 번더 실행

55

50 45

35 40 20 25

70(삭제된 노드)

힙의 재구성

- 최댓값인 55를 삭제하고 마지막 노드 25과 위치를 변경

25

50 45

35 40 20 55(삭제된 노드)

70(삭제된 노드)

(다음페이지에 이어서...)

- 25와 자식 노드의 가장 큰 값과 위치를 변경(25 <-> 50)

25

50 45

35 40 20 55(삭제된 노드)

70(삭제된 노드)

- 25와 자식 노드의 가장 큰 값과 위치를 변경(25 <-> 40)

50

25 45

35 40 20 55(삭제된 노드)

70(삭제된 노드)

50

40 45

35 25 20 55(삭제된 노드)

70(삭제된 노드)