개발자 기술면접

과목 :알고리즘 , 자료구조 , 운영체제 , 네트워크, 데이터베이스, 프로그래밍 언어

1. 알고리즘

정렬 알고리즘

Name	Best	Worst	Stable	Memory
버블정렬	n	n^2	True	1
선택정렬	n^2	n^2	False	1
삽입정렬	n	n^2	True	1
셸정렬	$n \log n$	(best) $n \log^2 n$	False	1
병합정렬	$n \log n$	$n \log n$	True	n
퀵정렬	$n \log n$	$n \log n \sim n^2$	False	$\log n \sim n$

선택, 삽입, 버블 정렬 코드

```
def buble(a):
    n=len(a)
    for i in range(0,n-1):
        for j in range(i+1,n):
            if a[i]>a[j]:
                a[i],a[j]=a[j],a[i]
    print(a)
def insert(a):
    n=len(a)
    for i in range(0,n-1):
        for j in range(i+1,n):
            if a[i]>a[j]:
                a[i],a[j]=a[j],a[i]
    print(a)
def selction(a):
    n=len(a)
    for i in range(0,n):
        tmp=a[i]
        idx=i
        for j in range(i+1,n):
            if tmp>a[j]:
                idx=j
                tmp=a[j]
        a[i],a[idx]=a[idx],a[i]
    print(a)
```

4. 퀵 정렬

```
def quick(arr, l,r):
        if l<r:
                p=part(arr,l,r)
        quick(arr,l,p)
        quick(arr,p+1,r)
def part(arr, l, r):
        pivot=r-1
        idx=l-1
        for I in range(l,r-1):
                if arr[i]<arr[pivot]:
                        idx++
                        swap(arr[i],arr[idx])
        swap(arr[idx+1],arr[pivot])
        return idx+1
```

5 머지 소트

```
def mergSort(arr, l, r):
        if l<r:
                m=(l+r)//2
                mergeSort(arr,l,m)
                mergeSort(arr,m+1,r)
                merge(arr,l,m,r)
```

과 같은 방식으로 동작함

자료구조

1. 스텍

1) LIFO 마지막에 들어게 처음 방출

2) 선형 자료구조임

2) 후위 표기, 인터럽트 처리루틴, 함수 호출 메모리구조, 올바른 괄호 등

2. Queue

1) 선형 자료구조 FIFO

2) 스케줄링, 프로세스 레디 큐, 대기중인 자료 적합,

3. 트리

1) 사이클이 없는 그래프

2) 계층적 자료를 표현

3) 차수 = 어떤 노드가 가진 자식의 수 (인접리스트 , 인접 배열을 통해 표현가능)

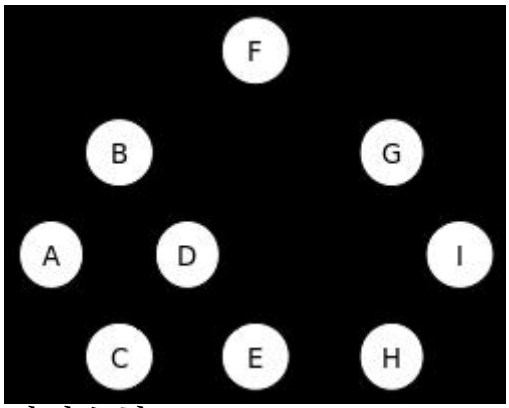
3-1 이진트리

1) 모든 노드가 최대 2개의 서브트리를 가짐

2) preorder : root -left - right

inorder: left-root-right

postorder: left-right-root



전위순회 : fbadcegih

중위순회: abcdefghi

후위순회 : acedbhigf

레벨순회: fbgadiceh

4. 헤시

1) 값을 키와 벨류로 저장

2) 좋은 헤쉬 함수의 조건 => 적은 충돌, 빠른 계산, 해시 테이블 고르게 분포

3) 해시함수 종류=>제산(모드연산) ,폴딩(더하기), 중간 제곱함수, 비트추출, 숫자분석

4) 해시 충돌

개방 주소: 선형 탐색, 제곱 탐색 , 이중해시

체이닝: 버켓 내에 연결리스트를 통해 충돌된 데이터 관리

체이닝=> 연결리스트를 통해 처리 계산 간단, 성능 저하 -> 선형증가

개방주소=> 포인터 필요없음, 추가 저장공간 필요 없음, 삽입 삭제 오버헤드 감소

성능 저하 -> 매우크게증가

5. 힙

1) 최대힙 초소힙

추가연산

힙의 가장 마지막 자리에 추가해 부모와 비교 하면서 타고올라옴

삭제

힙의 루트 삭제 마지막 인덱스 루트에 놓코 재구성

```
      80                  80                        90
   40       30   ->   40          30    ->      40          80
20    10             20  10      90           20   10       30
```

6. 이진 탐색 트리

삽입:

루트에 서부터 작으면 왼 크면 오른 줄기 타고가다 빈공간에 삽입

삭제:

자식이 없는 노드 -> 바로삭제

서브트리가 1개 -> 삭제 노드의 부모와 서브트리 연결 후 삭제

서브트리 2개-> 왼쪽 가장큰값 || 오른쪽 가장 작은 값 선택후 스왑 재구성

3. OS

멀티 프로그래밍: 입출력 작업의 종료를 대기 하는 동안 다른 프로세스 실행
멀티 프로세스: 여러 프로세스가 협력하여 일을 처리
멀티 테스크 : 테스크를 번갈아 가며 수행
멀티 스레딩 : 하나의 프로세스를 여러개의 단위로 실행 하며 여러 스레드 끼리 자원 공유

Program vs process
프로그램을 실행하여 메인 메모리에 올라가 있는 상태->process

Thread
프로그램의 실행 단위로 코드 데이터 메모리 공유
멀티 쓰레드 모델
1:n
병렬 처리 불가능,한번에 하나의 쓰레드만 커널에 접근
한 쓰레드가 봉쇄형 시스템 호출 전체 포로세스 봉쇄

n:n
병렬처리 쌉가능 , 효율적으로 좋지 못함
사용자 쓰레드 생성시 커널 쓰레드 새엇ㅇ

n:m n<m
어느정도 병럴처리 커버 가능 , 효율 적으로 훌륭

n:m + 1:1 모델
가장 진보됨

프로세스 상태 전이도
new - ready - running - terminated - waiting
처음 프로그램을 실행(new)
실행 가능상태 (ready)
스케줄러에 의해 (runing)- 컨텍트 스위치(ready)-종료(terminated)
io이벤트 대기 (waiting)

선점 스케줄링(preemptive) : 서로 경쟁을 허용 RR
비선점 스케줄링(non-preemtive):경쟁없이 FCFS,SJF,HRRN

임계구역 문제 해결 조건
쓰레드는 데이터를 공유하기 때문에 조심히 다루어야한다
이 공유데이터에 접근하는 코드를 critical section이라고 함

임계구역 문제 해결
상호배제(mutual Exclusion): 하나의 한 프로세스만 임계구역에 접근가능
유한대기(bounded waiting): 유한 시간내에 임계구역에 들어가야함
진행(progress): 임계구역을 사용하지 않으면 다른 프로세스가 접근 가능해야함

데드락 발생조건
상호배제, 점유 대기, 비선점, 순환 대기

단편화
외부단편화: 총 메모리 공간은 만족하나 연속된 메모리 공간이 없어 할당 불가
내부단편화: 메모리 할당 최소 크기 4kb 인데 3kb 만 사용하면 1kb 의 단편화

페이징
페이지 크기 = 8KB 메모리 256KB
사상 하려면 32개의 페이지 필요
6비트 페이지 넘버 오프셋 2*13 13 비트 필요

페이지 교채알고리즘: fifo, lfu, lru
쓰레싱:

4. 컴퓨터구조

cpu
제어장치 , 레지스터, 연산장치 로 이루어진다

명령어 파이프 라인
명령어의 실행을 독립적인 모듈로 나누어 동시에 실행
n개의 명령어를 k 개의 독립 묘듈로나눌 때 효율 = n*k/n+k
이론 적으론 BUt
모든 명령어가 모든 단계를 지나지 않음
조건 분기시 미리 인출 한 명령어 불필요
가장 오래걸리는 단계를 기준으로 명령어가 나님
io 기억장치 접근 등은 동시 불가

소수의 표현
ieee 754
1비트 주소 양수 0 음수 1
8비트 0111 1111 + 가수
23비트 소수 표현
1.101101*2^3 = [0][0111 1111+ 0000 0011][1011 0100 0000 0000 0000 000]

리틀 에디안 빅 에디안
Ox 12345678
리틀 에디안 78 56 34 12
빅 에디안 : 12 34 56 78

RISC vs CISC

	CISC	RICS
컴퓨터 구조	복잡	간편
명령어 구성	복잡하고 다양한	간단하고 최소화
길이	다양한 길이	고정된
레지스터	적음	많음
속도	느림	빠름
용도	개인 피시	워크스테이션 , 서버

컴퓨터 네트워크

osi 7 Layer
어플리케이션
프레젠테이션
세션
트렌스포트
네트워크
데이터/링크
물리
에플리케이션
패킷 지연의 종류
nodal processing : 라우터 처리지연(오류검출)
queuing : 큐에서 링크로 전송되기를 기다림
전송지연: 페킷의 모든 비트들이 링크에 올라가는데 걸리는 시간
전파지연: 출력 링크에서 다음 라우터까지 걸리는 시간

네트워크 계층화의 이점
유지보수 유리함, 명확한 구조화를 통해 시스템을 구분하고 명확한 관계설정

http 프로토콜
비 연결지향: 클-> 서 요청 서버 -> 응답 접속 끊음
stateless : 사용자의 상태 정보를 알 수 없음 -> 쿠키와 세션을 통해

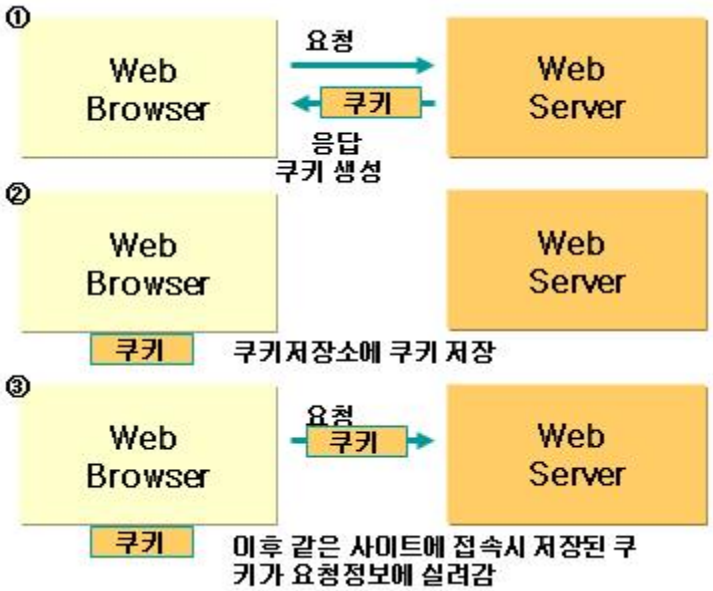
쿠키?
클라이언트 로컬에 저장되는 키와 벨류로 이루어진 데이터
사용자 인증 유효시간 명시가능 -> 만료 해야 여없어짐
사용자가 요청하지 않아도 브라우저가 알아서 전송함
구성
이름 , 값, 유효시간, 도메인 , 경로

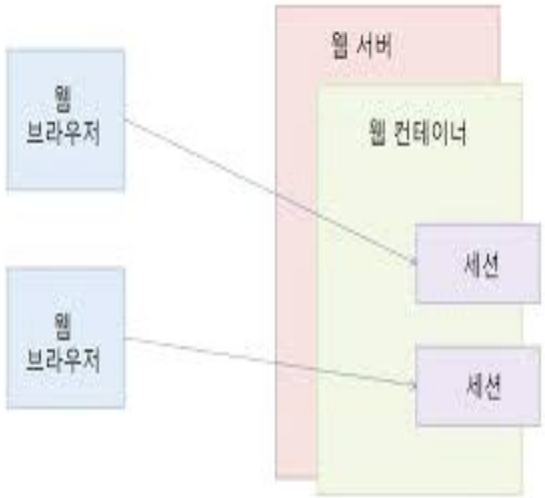
세션

쿠키를 기반으로 세션은 서버에 관리됨

웹 브라우저가 서버에 접속해서(id생성) 브라우저 종료까지 유지

보안성에 좋지만 서버에 부담됨





트렌스포트 레이어

TCP vs UDP

	tcp	udp
신뢰성	신뢰성 ack,checksum	신뢰 없음 체크씀만
연결성	연결 지향형	비 연결성
재전송	오류 및 패킷 손실 재전송	재전송 없음
특징	흐름제어, 전송 제어	속도 뿌름
용도	신로성 필요한곳	동영상 , 음성 멀티미디어

tcp 프로토콜

신뢰성있는 데이터 전송, 흐름제어, 혼잡제어, 연결제어

gbn, sr

연결제어

c ---syn m----> s

c <--syn n ack m+1

anc n+1 --------->

연결종료

fin wait 1 ----finbit=1 seq=x ----> closewait
fin wait 2 <---ackbit=1 acknum=x+1
<--finbit=1 seq=y ----- last_ack
timed wait ----ackbit=1 acknum=y+1--> closed

closed
1. 클라이언트 핀비트에 1을 넣고 시퀀스x 넣고 보냄 (fin wait)
2. server send ackbit=1 acknum=x+1 to cliend (server closewait),(c fin wait2
3. server send finbit=1 seq=y to client (server last ack)
4. 클라이언트에서 서버에 종류 수락 패킷 전송 후 타임웨잇
5. 서버가 수락 패킷을 받으면 연결종료
6. 클라이언트는 서버에서 올 남은 패킷을 기다리다 종료

흐름제어 : 수신 속도가 처리 속도를 넘지 않게 유지함 수신측은 수신버퍼를 rwnd로 송신측에 알려 유지시킴
혼잡제어: 네트워크 처리망이 처리가능한 데이터 량을 넘지 않게 하는 전송 량을 천천히 늘리다 패킷 손실이 늘어나면 반으로 줄이는 방식

네트워크 계층
ip프로토콜
8비트 4개 32비트 ipv4
16비트 8개 128비트 ipv6
터널링을 이

CIDR a.b.c.d/x x값을 유연하게 유지할 수 있음
상위 x 비트가 서브넷임을 명시하고 x 값을 유연하게 관리해 기존의 클래스 주소 체계 보다 유연하게 ip 주소를 나눌 수 있음

DHCP
유동 ip를 제공해주는 프로토콜
1. 광역으로 물어봄 어디 나 아이피좀 줄사람?
2. 어내가줄깨 여기로 연락해
3. 어나아이피좀
4. 어그래 거기로 줄깨

DB

정의 조작 제어

정의: create , alter, drop , turncate

조작: select, insert, update, delete

제어: grant, revoke, commit

정규화
제 1 정규형 : 모든 속성이 원자값 이어야한다.
제 2 정규형: 완전 함수종속 만족 부분종속이 없어야 한다.(기본키가 하나가 되게)
제 3 정규형: 기본키를 제외한 속성들 간의 이행적 함수종속
BCNF: 모든 결정자가 후보키

트렌잭션
원자성(automicity): 모든 연산이 적용되거나, 아무것도 적용되지 않아야함
일관성(consistency): 트렌젝션 수행전 DB가 무결했다면 수행후에도 무결
독립성(isolation): 트랜젝션수행중 다른 트랜잭션은 끼어들 수 없음
지속성(durablility): 트랜젝션의 결과는 시스템이 고장나더라도 반영되야함\
isolation level

	diry read	unrepetable read	phantom read
read uncommited	0	0	0
read commited	x	0	0
repeatable read	x	x	0
serializable	x	x	x

더티리드 : 트랜젝션 작업이 완료되지 않은 시점에서 다른 트렌젝션에서 정보를 읽게 되는 현상
unrepeatable read: 한 트랜젝션 네에서 같은 쿼리인데 결과가 다름
팬텀 리드 : 다른 트렌잭션에 수행하는 변경잡업으로 레코드가 보였다 안보였다함