본 내용은 웹 서핑중에 발견한 소프트웨어 직군의 직무 면접에 대한 예상 질문과 답변을 정리한 내용입니다.

[ 문제 ]

- 데이터 베이스

1) DB 무결성과 정합성 설명

2) DB 무결성의 4가지 설명

3) DB 제약 조건의 종류와 무결성을 연관 설명

4) DB 정규화 단계 설명

5) 데이터 베이스 언어 (DDL, DML, DCL, TCL)에 대한 설명

6) 뷰 특징과 장단점 설명

7) 조인 (내부/외부 조인)에 대한 설명

8) JDBC와 ODBC의 차이점 설명

9) statement와 preparestatement의 차이점

- 웹

1) GET과 POST의 차이점

2) session과 cookie 차이점과 사용 용도 설명

3) HTML과 XML의 차이점과 장단점 설명

4) MVC 모델이란, 프레임워크란?

5) Spring Framework에 대한 설명

6) JSP와 Servlet의 차이점

7) JSP와 ASP 그리고 PHP의 차이점

8) AJAX에 대한 설명

- JAVA

1) String과 StringBuffer의 차이점

2) Interface와 Abstract의 차이점

3) Static의 의미와 선언하지 않은 것과의 차이점

4) Heap과 Stack에 대한 설명

5) Call by value와 Call by reference 차이점

6) Java와 Javascript의 차이점

- 기타

1) 정렬 알고리즘중 가장 빠른 방식과 느린 방식을 설명와 이유

2) 탐욕 알고리즘이란?

3) 객체란?

4) 이진 탐색 방식을 시퀀스 서치와 비교하여 이점을 설명

- 자료구조

1) Tree란?

2) Binary Tree란?

3) Binary Tree에서 Non-leaf와 Leaf 노드 개수의 관계식은?

4) Heap Tree란?

5) Heap Tree의 데이터 삭제, 추가 과정을 설명하시오

6) Tree에서 pre, in ,post order를 설명하시오

7) B tree란?

8) B Tree의 데이터 삭제 추가 과정

9) Array List와 Linked List의 차이점은?

10) Graph란?

- 알고리즘

1) Big O란?

2) Divide and conquer를 사용한 알고리즘 예를 들어보라

3) Divide and conquer와 Dynamic Programming의 차이점 및 유사점은?

4) Backtracking과 Branch and Bound의 차이점은?

5) Merge/Quick/Heap sort의 best, average, worst 케이스의 시간 복잡도는?

6) Merge sort와 Quick sort의 차이점과 방식은?

7) Quick sort와 Bubble sort의 차이점은?

8) Floyd 알고리즘이란?

9) Dijkstra 알고리즘이란?

10) Floyd와 Dijkstra 알고리즘의 차이점과 정의는?

11) 프림 알고리즘과 크루스컬 알고리즘의 차이 그리고 시간복잡도는?

12) Knapsack problem과 Traveling Salesman Problem에 대한 설명

13) P와 NP, NP-Completeness에 대한 설명

- 운영체제

1) OS 무엇이며 핵심 기능은?

2) 컴퓨터에서 OS가 하는 일

3) 프로세스와 스레드의 차이점을 Context Switching 관점으로 설명

4) 어떤 경우에 프로세스와 스레드를 사용하는지 예시와 각각의 장단점 설명

5) 리눅스를 사용해보았다면, 자주 사용하는 리눅스 명령어

6) IPC에 대한 설명

7) 데드락에 대한 설명과 이를 해결하기 위한 회피책 설명

8) 임계 영역이란 무엇인지 설명하고, 세마포어와 뮤텍스의 차이점 설명

9) CPU 스케줄링에 정의와 목적 설명

10) 프로세스 스케줄링의 종류에 대한 설명과 해당 예시

11) 가상 메모리에 대한 정의와 구현 기법 설명

12) 페이지 교체 알고리즘에 대한 설명

- 소공

1) 블랙박스 테스트와 화이트 박스 테스트에 대한 설명

2) 코드 결합도와 응집도란?

3) SW공학 생명 주기 모형 종류와 특징 그리고 장단점

4) 리버스 엔지니어링의 특징과 장단점

5) 소프트웨어 공학이 필요한 이유와 좋은 설계란?

6) SW 형상관리란

- 네트워크와 인프라

1) GET과 POST의 차이점

2) TCP/IP, UDP 프로토콜 설명

3) MTTR, MTTF, MTBF에 대한 설명

4) 파티션 테이블이 필요한 이유?

5) OSI 7 계층을 나눈 이유

6) TCP 3 hand shaking이란?

7) L2, L3, L4 차이점

8) 허브와 스위치의 차이점

9) 파이썬과 다른 언어의 차이점

10) 데이터 계층의 PDU는 무엇인가?

11) 시퀀스 넘버에 대한 역할

12) 슬라이딩 윈도우의 역할

13) HTTP 응답코드의 종류를 아는대로 설명

14) MIME 프로토콜 설명

[ 예상 답안 ]

- 데이터 베이스

**1) DB 무결성과 정합성 설명**

**- 무결성이란 데이터 값 자체가 올바른지 여부를 뜻하는 것, 정합성은 동일한 데이터를 나누어 저장하거나, 그를 이용함에 있어서 해당 위치의 값들이 모두 동일한지를 의미한다.**

**= 서로 다른 공간에 잘못된 값이 모두 똑같이 저장되어 있다면, 정합성은 지켜졌으나 무결성은 지켜지지 않은 것**

**2) DB 무결성의 4가지 설명**

**- 개체 무결성 / 참조 무결성 / 도메인 무결성 / 업무 무결성(위키피디아 기준 포함되지 않음)**

**= 개체 무결성 : 모든 엔티티는 고유한 값을 가져야 하며, Null이면 안됨**

**= 참조 무결성 : 참조되는 엔티티의 식별자와 값이 일치하거나 Null이어야 함.**

**= 도메인 무결성 : 테이블 필드의 무결성 보장을 위해 값 자체에 대한 제약을 통해 올바른 값인지를 확인 (ex 주민번호에 알파벳)**

**= 업무 무결성 : 기업이나 업무를 수행하는 방법 또는 데이터 처리 규칙을 의미 (ex 100만원 이상이면 배송비 무료와 같은 조건) / 주로 프로그램에서 확인**

3) DB 제약 조건의 종류와 무결성을 연관 설명 [정확한 내용은 검색 요망]

- DB 제약 조건 : NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK

= 개체 무결성 - NOT NULL, UNIQUE, PRIMARY KEY..?

= 참조 무결성 - FOREIGN KEY

= 도메인 무결성 - NOT NULL, CHECK

= 업무 무결성 - CHECK

4) DB 정규화 단계 설명 [정확한 내용은 검색 요망]

= 1차 정규화 : 각 튜플의 컬럼 값이 1개씩, 원자값을 가져야함.

= 2차 정규화 : 완전 함수 종속을 가져야 함

= 3차 정규화 : 이행적 함수 종속이 없어야 함

= BCNF : Boyce Codd Normal Form, 모든 결정자가 후보키 집합에 속해야 함.

[자세한 내용은 이 페이지를 참고]

\* Anomaly

- Insert Anomaly : 삽입시 원하지 않은 데이터가 추가되는 현상

- Update Anomaly : 삭제시 원하지 않은 데이터가 함께 삭제되는 현상

- Delete Anomaly : 갱신시 일부 튜플의 정보만 갱신되는 현상

5) 데이터 베이스 언어 (DDL, DML, DCL, TCL)에 대한 설명

= DDL : Data Definition Language, 테이블과 같은 데이터 구조 정의하는 명령 (Create, alter, drop, ..)

= DML : Data Manipulation Lang~, 데이터 조회하거나 변형하는 명령 (select, insert, update, delete)

= DCL : Data Control Lang~, 데이터 베이스 접근 권한을 주거나 회수하는 명령 (grant revoke)

= TCL : Transaction Control Lang, 논리적 작업을 단위로 묶어 처리하기 위한 명령 (Commit, Rollback..)

6) 뷰 특징과 장단점 설명

- 뷰 : 데이터를 보여주기 위해 유도된 가상의 테이블

= 장점 : 논리적 독립성을 제어하고, 항상 뷰를 통해 접근하므로 원하지 않는 데이터 접근을 막는 보안 기능을 제공, 서로 다른 뷰를 정의할 수 있음.

= 단점 : 독자적 인덱스를 가질 수 없음, 정의를 변경할 수 없음, 삽입/삭제/갱신 연산에 많은 제약이 따름

**7) 조인 (내부/외부 조인)에 대한 설명**

**- 조인 : 둘 이상의 테이블간 논리적 관계를 기준으로 데이터를 검색해 결과 집합을 만드는 연산**

**= inner join : 두 테이블간에 공통으로 포함되어 있는 값을 결과 집합으로 만든다.**

**= outer join : 한 테이블에만 데이터가 있더라도 값을 가져와 집합으로 만든다.**

**==> Left / Right / Full ( 왼쪽, 오른쪽, 양쪽 조건에 일치하지 않은 데이터를 가져온다. )**

**==> 관련 link (https://rh-cp.tistory.com/44)**

8) JDBC와 ODBC의 차이점 설명

- JDBC : Java Database Connectivity

- ODBC : Open Database Connectivity

= 두가지 모두 언어에서 Database를 사용하기 위한 API, JDBC는 언어의 종속성을 가지나 ODBC는 그러하지 않음, 또한 ODBC의 경우 사용하는 데이터베이스의 차이는 ODBC driver에 의하여 변환되므로 유저는 어떠한 DBMS를 사용하고 있는지 의식할 필요가 없음

9) statement와 preparestatement의 차이점

- 두가지 모두 쿼리이며, 가장 큰 차이점은 캐시 사용여부이다. 쿼리는 문장 분석-> 컴파일 -> 실행 순서대로 진행되는데 preparedstatement를 사용하게 되면 바로 데이터를 담아 실행된다. 또한 SQL 인적션도 방지

- 웹

**1) GET과 POST의 차이점**

**- GET : 주소값에 Query string으로 데이터를 전송**

**- POST : HTTP 바디 안에 담아서 데이터를 전송**

**= GET의 경우 주소값에 담아서 보내는어 데이터가 사용자에게 노출되며, 데이터의 양도 제한되어 있음.**

**= POST의 경우 눈에 보이지 않아 보안성이 조금 더 있으나, 단순히 보이지 않을 뿐 완전한 보안책은 아님.**

**\* HTTP Method : GET / HEAD / POST / PUT / DELETE / OPTIONS 등**

**- GET : 데이터 요청시**

**- HEAD : GET과 동일하지만 본문은 포함하지 않도록 요청**

**- POST : 데이터를 제출할 때**

**- PUT : POST와 유사, 데이터를 담아 보냄**

**- DELETE : 데이터 삭제를 할 때, PUT과 반대**

**- OPTIONS : 서버에서 지원되는 메소드 종류 확인**

**2) session과 cookie 차이점과 사용 용도 설명**

<https://sdevstudy.tistory.com/27>

**- session과 cookie의 가장 큰 차이점은 서버에 저장되는지 쿠키에 저장되는지의 여부이다.**

**= session은 http의 stateless를 보완하며, 서버의 자원을 활용하여 보안과 속도에서 빠르다. 하지만 지나치게 많은 양을 저장할 경우 서버의 리소스가 런아웃될 수 있다.**

**= cookie는 세션을 보완하기 위하여, 사용자 측에 저장하며 다량의 데이터를 저장할 수 있다.**

3) HTML과 XML의 차이점과 장단점 설명

- HTML : Hyper Text Markup Language

- XML : eXtensible Markup Language

= XML은 데이터를 전송과 저장에 목적을 둔 마크업 언어, HTML은 data를 웹상에 표현하기 위함

= HTML은 미리 정의된 태그가 있으며, 의미에 따라서 적절한 태그를 선택해서 사용해야 한다.

= XML은 사용자가 태그를 정의할 수 있고, 환경에 구애받지 않는다.

\* XML의 정의를 위한 DTD와 XSD

- DTD : Document Type Definition

- XSD : XML Schema Definition

\* Well Formed / Valid

- Well Formed : 잘 짜여진 문서를 의미하며, 하나의 루트노드, 시작과 끝태그의 한쌍, 엘리먼트간 겹침이 없음 등, Semantic 느낌(?)

- Valid : DTD나 XSD에 미리 정의된 문서의 구조를 모두 준수하는지 등 Syntax 느낌(?)

**4) MVC 모델이란, 프레임워크란?**

**- MVC : Model View Controller**

**= 프로그래밍 디자인 패턴으로 데이터의 테이블에 대응되는 Model과 이를 조작하기 위한 Controller 그리고 Model의 값을 보여주는 View로 구분된다.**

**= 사용자 인터페이스와 비지니스 로직을 분리해 서로 영향 없이 쉽게 애플리케이션을 수정할 수 있는 패턴**

**- Framework : 소프트웨어의 설계과 구현을 재사용할 수 있도록 협업화되 클래스로 제공되는 것**

**= 장) 바닥부터 설계하여 프로그래밍하는 것보다 효율적이고 생산성이 좋아지며, 유지보수가 편리해진다.**

**= 단) 학습 기간이 필요하며, 유연성에 한계가 있다.**

5) Spring Framework에 대한 설명

- 자바를 기반으로한 오픈소스 애플리케이션 프레임워크

= Java 엔터프라이즈에 비하여 가볍고 단순하다.

6) JSP와 Servlet의 차이점

- Servlet : Server + Applet 톰캣 위에서 동작하는 Java. 전형적인 WAS에서 메모리에 적재된 다음 HTTP 요청을 처리하는 자바프로그램

- JSP : Java Server Page, 서블릿이 자바안에 HTML태그가 섞여있어 분리적 측면에서 떨어지는 효율이 낮은 문제를 하기 위하여 HTML안에 Java코드를 담아 표현함. 다만 톰캣에서는 JSP도 Servlet으로 변환되어 컴파일되고 실행된다.

= 둘을 모두 사용하는 환경에서는 주로 MVC(Model : DB, View : JSP, Controller : Servlet)으로 처리된다.

7) JSP와 ASP 그리고 PHP의 차이점

- JSP : Java Server Page

- ASP : Active Server Page

- PHP : Hypertext Preprocessor

= ASP : MS의 비주얼 베이직 스크립트를 이용하여 MS 운영체제에서 동작

= JSP : Java 기반으로 HTML안에 포함되어 처리

= PHP: C를 바탕으로한 구조를 가지고 다른 언어에 비하여 개발이 쉽고 빠름

8) AJAX에 대한 설명

- AJAX : Asynchronous Javascript And Xml

= 웹에서 XMLHttpRequest 객체를 이용하여 페이지를 다시 로드하지 않아도 데이터를 로드하는 방법. 즉, 페이지와 비동기적으로 데이터를 송수신하는 기술이다.

= 전체 페이지를 로드하지 않아 속도가 빠르고, 요청에 대한 응답을 대기하지 않도록 처리가 가능하다.

- JAVA

1) String과 StringBuffer의 차이점

- Java에서는 String을 상수로 취급하여, 문자열을 합치거나 변경하는 경우 새로운 문자열이 만들어지면서 처리된다. 따라서 GC에 의해 주기적으로 제거되어야 하는등 다양한 오버헤드가 발생한다.

= 반면 StringBuffer나 StringBulider의 경우, String과 달리 변경이 가능하며, 동기화 여부에 따라 동기화가 필요하지 않은 상황에서는 Builder, 필요한 상황에서는 Buffer가 적절하다.

2) Interface와 Abstract의 차이점

- 두가지 모두 선언만 있고 구현이 없는 클래스

- Interface : 이를 구현하는 모든 클래스가 특정 메소드를 구현하도록 강제하는 역할

- Abstract : 공통점을 찾아 추상화시켜 사용하기 위함, 미완성된 클래스라고 보면 적절할 듯.

3) Static의 의미와 선언하지 않은 것과의 차이점

- 클래스의 변수를 static으로 처리하면, 메모리 공간에 할당이되며 그 값은 클래스간 공유할 수 있다.

4) Heap과 Stack에 대한 설명

- 이 부분은 프로세스의 메모리 영역을 설명하는 것이 나아보임.

- 메모리 영역은 [Code / Data / Stack / Heap] 영역으로 나뉨

= code : 실제 프로그램의 코드 및 상수가 적재되는 공간

= data : 프로그램이 실행되면서 필요한 데이터 저장되는 영역, 전역변수(global)와 정적변수(static)가 저장된다.

= stack :함수에 정의된 지역(매개) 변수가 저장된다. (함수 스택)

= heap : 그외 다른 형태의 데이터를 저장하기 위한 빈 공간

= 주로 (낮은 주소) [code, data, heap, stack] (높은 주소)으로 stack은 낮은 방향으로, heap은 높은 방향으로 할당된다.

**5) Call by value와 Call by reference 차이점**

**- 두가지 모두 함수에 매개변수를 어떻게 전달하는지에 대한 방법을 말한다.**

**= Call by Value는 매개변수에 A의 값을 던지면, 매개변수는 A의 값을 복사하여 가져가며 메모리상에 A와 매개변수는 별개로 존재**

**= Call by Reference는 매개변수에 A의 Alias를 전달하는 방법**

6) Java와 Javascript의 차이점

\* 엄밀히 말해서 공통점이라곤 프로그래밍 언어라는 것 밖에 없는듯 생각됨

\* 어느 사람의 말 : 삼성과 삼성부동산, 네이버와 네이버시스템즈와 같은 사이

- 자바는 OOP 언어로 클래스 기반. Compiler언어

- 자바스크립트는 프로토타입 기반, Dynamic Typing 언어, 인터프리터로 처리

- 기타

1) 정렬 알고리즘중 가장 빠른 방식과 느린 방식을 설명와 이유

\* 엄밀히 말해서 누가 빠른지, 느린지는 설명하기 힘듬. MCU와 같은 장치에서는 스택이 없기때문에 쉘 소트를 사용함.

- 일반적인 상황에서 (데이터가 충분히 크고, 적당히 섞여 있어 정렬 알고리즘 로직이 아닌 이를 위한 기타 오버헤드를 충분히 감쇄할 수 있는) 내부 구조가 복잡하지 않은 quick sort가 빠름.

- Bubble이 아닐까.. 정렬되어 있든 되어 있지 않든 모든 원소에 대하여 정렬을 하니까..?

2) 탐욕 알고리즘이란?

- 탐욕법이란 각 단계에서 가장 최선의 선택을 하여 문제를 해결하는 방법

3) 객체란?

- 객체 : 객체 지향 프로그래밍(OOP)에서 객체는 실세계에 존재하는 물체를 추상화를 통하여 모델링한 것

= 객체 지향 프로그래밍에서 객체의 행위는 method로 표현되며, 객체의 특성은 변수로 정의된다.

4) 이진 탐색 방식을 시퀀스 서치와 비교하여 이점을 설명

- 이진탐색 : Binary Search로 정렬되어 있는 데이터의 탐색범위를 1/2씩 줄여나가면서 원하는 데이터를 탐색하는 방법 O(lgn)

- 시퀀스서치 : 모든 데이터를 한번씩 비교해보면서 데이터를 탐색

= 데이터 양이 충분히 많고, 탐색의 빈도가 높을 수록 이진 탐색이 효율적. 다만 정렬하는 것이 필수이기 때문에 각 데이터의 개체 크기가 지나치게 크거나 탐색의 빈도가 극히 적을 때는 시퀀스 서치가 유리할 수도 있음.

- 자료구조

1) Tree란?

- 트리란 노드로 이루어진 자료구조로 Acyclic connected graph로 불리며, 사이클이 없고 둘을 잇는 간선이 하나뿐인 그래프

2) Binary Tree란?

- 바이너리 트리란, 하나의 노드가 최대 두개의 자식 노드를 가질 수 있는 트리를 말함

\* 세부 이진 트리

- Full BT : 포화 이진 트리라고 하며 자식이 0 또는 2개인 BT

- Perfect BT : 포화 이진 트리라고 하며, 모든 리프 노드들의 높이가 같은 BT입니다.

- Complete BT : 완전 이진 트리라고 하며, 마지막 레벨을 제외한 나머지 레벨이 모두 채워진 BT

3) Binary Tree에서 Non-leaf와 Leaf 노드 개수의 관계식은?

- 완전 이진트리 기준 모든 노드의 개수는 높이 h에 대하여 2^(h+1)-1개이며, leaf노드의 개수는 2^h개이다.

4) Heap Tree란?

- Heap Tree란? 일반적으로 말하는 힙이며, 완전 이진 트리로 구성되어 있음.

= 부모 노드와 자식 노드 사이에 일정 규칙을 정의하여 구성됨(크거나 작거나)

5) Heap Tree의 데이터 삭제, 추가 과정을 설명하시오

- 삭제 : 루트를 빼고, 리프 노드중 가장 마지막을 빼서 루트로 이동시킨다. 이후 기존 힙의 규칙에 맞게 자식과 비교해가면서 내려감 (max heap이면 자식중 큰쪽과 변경)

- 삽입 : 리프노드중 가장 마지막 노드위치에 삽입하고, 부모노드와 비교해가면서 값을 교체

6) Tree에서 pre, in ,post order를 설명하시오

- 트리 순회에 있어서 부모를 확인하는 선서에 따라 pre, in, post로 구분.

= 전위 순회 : [부모, 좌, 우]

= 중위 순회 : [좌, 부모, 우]

= 후위 순회 : [좌, 우, 부모]

7) B tree란?

- B tree란? 자식의 개수가 2이상인 트리. 만약 노드안에 데이터를 최대 m개까지 포함할 수 있으면, m차 b tree라고 부름

8) B Tree의 데이터 삭제 추가 과정

- 물어보면 모른다고 답할 거임.

= 상세한 내용은 이 페이지 참고

9) Array List와 Linked List의 차이점은?

\* Array List는 아마도 다수의 Primitive type 데이터를 묶은 Compositive Type을 말하는 듯함. (배열)

- ArrayList : 같은 데이터 타입이 연속으로 할당되어 있는 것.

- Linked List : 데이터 하나 하나가 서로 다른 메모리 영역에 할당되어 링크를 통해서 가르키고 있는 것.

= 랜덤 엑서스, 삭제나 업데이트에 따른 순서 정렬등의 차이가 있음.

10) Graph란?

- 그래프란 데이터 V와 데이터간의 연결선 E으로 구성되어 있는 자료 구조

= 트리와 달리 루트, 부모 자식의 개념이 없음

- 알고리즘

1) Big O란?

- 컴퓨터 알고리즘이 실제 동작함에 있어서 시간과 같은 절대적 수치가 아닌, 입력값에 비례하여 해당 알고리즘의 동작 시간을 나타내는 점근적 접근법에서 알고리즘의 최악의 시간 복잡도를 나타내는 표기법

2) Divide and conquer를 사용한 알고리즘 예를 들어보라

- Divde and conquer는 큰 문제를 소문제로 나누어 해결하는 기법으로 가장 간단한 예로서 merge sort, quick sort등이 있음.

3) Divide and conquer와 Dynamic Programming의 차이점 및 유사점은?

- 두 가지 패러다임 모두 하나의 문제를 소문제로 나누어 처리하는 것을 기반으로 하고 있음. 다만 D&C는 이미 계산한 소문제의 결과를 다시 확인하지 않는 경우에 활용할 수 있음. 반면 DP는 소문제에 대한 결과를 계속 확인함을 효율적으로 처리하기 위하여 이를 저장하여 이용

4) Backtracking과 Branch and Bound의 차이점은?

- Backtracking : DFS, 유망한 노드인지 루트를 이용하여 검사(Pruning)하고 그런 경우에만 하위 자식으로 내려가면서 검사하는 방법

- Branch and Bound : BFS, 가망이 있는 branch와 그렇지 않은 것을 구분하여 경우의 수를 줄임

= 둘 모두 경우의 수를 줄이기 위한 방법

= 다만 두개의 주요 차이점은 백트래킹의 경우 constraint satisfaction 문제에서 가능한 조합의 후보군을 만들어 가는 방법이고, Branch and bound는 최적화 문제에 사용된다.

5) Merge/Quick/Heap sort의 best, average, worst 케이스의 시간 복잡도는?

- 알고리즘 최선 / 평균 / 최악

= Quick : O(nlogn) / O(nlogn) / O(n^2)

= Merge : O(nlogn) / O(nlogn) / O(nlogn)

= Heap : O(n) / O(nlogn) / O(nlogn)

6) Merge sort와 Quick sort의 차이점과 방식은?

- 둘의 가장 큰 차이점은 O(n)이 필요한 동작을 Divide되기 전에 하느냐 Combine할때 하느냐 차이

= 상세한 내용은 이 페이지를 참고

7) Quick sort와 Bubble sort의 차이점은?

- 두 정렬은 정렬이라는 공통점을 빼고는 전혀 같을 게 없는데...

= Quick Sort는 Divide and Conquer

= Bubble Sort는 Brute Force 정도..?

8) Floyd 알고리즘이란?

- 그래프 자료구조에서 하나의 노드에서 서로 다른 노드까지의 최단 거리를 구하는 알고리즘

= i노드에서 j노드로 이동할 때 k노드를 거쳐서 가는것이 빠른가 그렇지 않은 가로 명제를 세워 for(k, i, j) [i, j] = max([i,j], [i,k]+[k,j])로 알 수 있음.

9) Dijkstra 알고리즘이란?

- 다익스트라 알고리즘은 하나의 시작점에서 출발하여 다른 노드까지의 최단 거리를 구하는 알고리즘

10) Floyd와 Dijkstra 알고리즘의 차이점과 정의는?

- 두 알고리즘의 주요한 차이점은 모든 노드에 대한 최단 거리를 구할 수 있냐 없냐

= Floyd는 n^3, 다익스트라는 pq를 이용하여 O(ElogE)로 나타난다. 알고리즘에서 모든 간선이 1회씩 검사되고 각 검사된 결과에 따라 최악의 경우 모든 경우에서 pq에 넣거나 삭제하는 동작을 수행해야한다. 이때 큐에 들어 있을 수 있는 최악의 개수는 E개이며, 따라서 O(ElogE)의 경우의 수가 나온다. log E <= logV^2 = 2logV이므로 O(ElogV) 또는 E가 V^2에 비해 매우 적은 경우 O(VlogV)로 표기하는 경우도 있다.

11) 프림 알고리즘과 크루스컬 알고리즘의 차이 그리고 시간복잡도는?

- 프림과 크루스컬 알고리즘은 최소 스패닝 트리를 구하는데 사용되는 알고리즘이다.

- 두가지 모두 Greedy한 방법으로 현재에서 가장 작은 간선을 선택하는 방식으로 동작한다.

= 크루스컬 알고리즘 : 모든 간선을 가중치의 오름차순으로 정렬한 다음, 간선을 선택하고 해당 간선에 연결된 두 노드가 서로 같은 집합에 포함되어 있는지 여부를 Disjoint set으로 판단한 다음 포함하거나 제외한다. O(ElogE)

= 프림 알고리즘 : 하나의 시작 노드를 바탕으로 인접한 노드들을 연결하는 간선중에 가장 작은 간선을 선택한다. 전체 알고리즘의 시간복잡도는 O(V^2)이며, 만약 밀집 그래프라면 프림이 더 빠르다. pq를 이용하면 프림 알고리즘도 O(ElogV)로 구현할 수 있다.

12) Knapsack problem과 Traveling Salesman Problem에 대한 설명

- 냅색과 TSP문제 모두 완전 탐색으로 풀어야 모든 경우의 수를 구할 수 있다.

= 일반적으로 노드의 개수가 적은 경우 냅색은 DP로 해결한다. TSP는 모르겠음.

= 완전 탐색의 경우 그 크기가 매우 크므로 두 문제 모두 노드의 수가 많으면 Backtracking 또는 Branch and Bound로 해결한다.

13) P와 NP, NP-Completeness에 대한 설명

- P : Polynomial time안에 해결할 수 있는 문제

- NP : Polynomial time안에 답의 존재 여부를 확인할 수 있는 문제

- NP-Hard : 모든 NP문제를 다항시간내 다른 문제 A로 환원할 수 있을 때, 이 문제를 NP-Hard 문제라고 한다.

- NP-Complete : NP이면서 NP-Hard인 문제로 위의 NP-Hard의 환원된 문제 A를 말함. 만약 NP-Complete 문제중 다항시간안에 풀어낼 수 있다면 P=NP임을 증명할 수 있다. (TSP, 해밀토니안)

- 운영체제

1) OS 무엇이며 핵심 기능은?

2) 컴퓨터에서 OS가 하는 일

\* 두 문제 유사해서 같이 답변

- OS : Operating System으로 사용자 대신 컴퓨터 시스템의 자원을 관리, 제어하여, 사용자로 하여금 쉽고 간편하게 컴퓨터 시스템을 이용할 수 있도록 하는 시스템 소프트웨어

= 역할 : 프로세스 관리, 기억장치 관리, 입출력 장치 관리, 자원 관리

\* 더 자세한 내용은 이 페이지 참고

3) 프로세스와 스레드의 차이점을 Context Switching 관점으로 설명

4) 어떤 경우에 프로세스와 스레드를 사용하는지 예시와 각각의 장단점 설명

\* 두 문제 유사해서 같이 답변

- Process : 프로그램의 독립 개체

- Thread : 프로세스 내에서 실행되는 흐름의 단위

= 스레드는 하나의 프로세스의 Code, Data, Heap을 공유한다.

- context switching : CPU 스케줄링에 의하여 실행되는 프로세스가 변경되는 것. 현재의 프로세스 상태를 저장하고 다음에 실행될 프로세스의 상태를 다시 CPU 레지스터와 메모리에 복구하는 것을 의미

= 멀티 프로세스의 경우 멀티 스레드에 비하여 fork하는 시간이 오래 걸리며 컨텍스트 스위칭으로 인한 오버헤드가 존재한다. 반면 Thread의 경우 code, heap, data 영역을 공유하며 그로 인하여 컨텍스트 스위칭의 오버헤드가 적고 각 스레드간 데이터 공유등으로 자원의 효율성을 꾀할 수 있다.

= 멀티 프로세스 사용 예시 : 각 프로세스가 독립적으로 작업을 수행할 수 있는 경우, httpd

= 멀티 스레드 사용 예시 : 서로 정보를 공유함으로써 얻는 이득이 큰 경우, 계산 프로그램(?)

5) 리눅스를 사용해보았다면, 자주 사용하는 리눅스 명령어

\* 질문 자체가 조금 모호하고 어느 수준의 답변을 원하는지 알수 없어서 예상 답안을 작성하지 않음.

6) IPC에 대한 설명

- IPC : inter process communication 프로세스간 통신하는 방법

= 파일, 소켓, 메시지 큐, 공유메모리, 세마포어

7) 데드락에 대한 설명과 이를 해결하기 위한 회피책 설명

- 데드락 : Deadlock, 둘 이상의 작업이 다른 작업이 끝나기만을 기다리고 있는 상태

= 발생 조건 : 점유 대기, 비선점(자원반환 전까지 뺏을 수 없음), 순환대기(A프로세스가 B프로세스의 자원을 필요로 하는 상태)

= 예방법 : 상호 배제 제거, 점유 및 대기 조건 제거, 비선점 조건 제거, 환형 대기 조건 제거

= 회피법 : 자원 할당 그래프 알고리즘, 은행원 할고리즘

8) 임계 영역이란 무엇인지 설명하고, 세마포어와 뮤텍스의 차이점 설명

- 임계 영역 : Critical section, 둘 이상의 스레드가 동시에 접근하면 안되는 자원

\* 레이스 컨디션이 발생하며, 상호 배제를 하지 않아 일어남.

= 뮤텍스 : Mutex : Mutual Exclusive 상호 배제. 한시점에 1개의 스레드만 점유할 수 있는 상호배제

= 세마포어 : Semaphore , 한시점에 다수개의 스레드가 이를 점유할 수 있는 상호배제 (1~N개)

= 둘의 가장 큰 차이점은 개수

9) CPU 스케줄링에 정의와 목적 설명

10) 프로세스 스케줄링의 종류에 대한 설명과 해당 예시

- CPU 스케줄링 : 다수의 프로세스가 시분할 단위로 실행되어 사용자로 하여금, 다수의 프로그램이 실시간으로 돌아갈 수 있도록 함

= 선점형 / 비선점형 스케줄링이 있다.

= 비선점형 : FCFS(First Come, First Served), SJF(Shortest Job First, 둘다 가능)

= 선점형 : MLQ(Multi Level Queue), SJF(Shortest Job First, 둘다 가능)

\* starvation : 우선순위가 낮거나 처리 시간이 길어 프로세스가 자원을 할당 받지 못하여 계속 대기하는 현상

\* aging : 프로세스가 대기한 시간을 나타내는 것

11) 가상 메모리에 대한 정의와 구현 기법 설명

12) 페이지 교체 알고리즘에 대한 설명

- 가상 메모리 : 주 메모리의 주소를 실제가 아닌 가상의 메모리 주소로 부여하는 방식

= 가상 주소로 할당되기 때문에 MMU에서 물리주소로 변경되어 접근된다.

= 프로그래머는 변수의 값이 실제 어디에 존재하는지 알 필요가 없으며, 멀티 태스킹 환경에 적합하다.

= 위와 같이 가상 주소를 할당하게 되면, 실제 메모리 용량보다 더 많은 가상 메모리를 할당 할 수 있는데, 특히 보조 기억 장치를 이용해서 디스크상에 페이지 단위로 데이터를 필요시마다 로드하여 사용할 수 있다.

= 페이지 교체 알고리즘

= FIFO : First in First out

= LRU : Least Recent Used , 최근 사용한 페이지는 계속 사용될 것이라고 생각

= MFU, LFU : Most/Least Frequent Used 최고, 최근 사용빈도를 카운팅하여

\* Thrashing : 페이지 교체 시간이 지나치게 많아지는 현상

\* space / time locality : 시간 공간 지역성, 최근 또는 인접한 부분을 자주 접근하는 특징

- 소공

1) 블랙박스 테스트와 화이트 박스 테스트에 대한 설명

- 화이트 박스 : 코드를 보면서 테스트 진행

= 제어 흐름 테스트, 경로 테스트

- 블랙 박스 : 입력 값에 대한 출력 값을 바탕으로 테스트 진행

= 경계값 분석, 동등분할 테스트

2) 코드 결합도와 응집도란?

- Coupling : 코드 중 일부가 다른 것과 얼마나 의존적인지

- Cohesion : 특정 기능을 위한 코드가 얼마나 뭉쳐있는지

= 결합도는 낮아야하고 응집도는 높아야 코드의 가시성과 재사용성이 높아진다.

3) SW공학 생명 주기 모형 종류와 특징 그리고 장단점

- Waterfall : 요구 분석 -> 설계 -> 개발 -> 테스트 -> 유지보수 : 시제품의 시간이 오래 걸리고, 위험 관리가 힘들며 유연성이 부족하다.

- Prototyping : 프로토타입을 만들고 피드백을 얻어 다시금 시제품을 만드는 모형, 유지 보수가 힘들고 형상관리가 필요

- Spiral : 위의 두가지를 섞어서, 계획 -> 위험 분석 -> 개발 -> 유저 평가로 진행

4) 리버스 엔지니어링의 특징과 장단점

\* 잘 모르겠음

- 소프트웨어의 생명 주기 모델의 반대로 구현 -> 설계 -> 분석 -> 요구분석 형태로 역추적하여 정보를 얻어가는 것

5) 소프트웨어 공학이 필요한 이유와 좋은 설계란?

- 소프트웨어 공학 : 소프트웨어를 설계 개발 및 유지보수하기 위한 체계적인 이론과 기술을 다룸

- 필요 이유 : 소프트웨어를 최소 비용으로 최소 시간에 개발하는 방법

- 좋은 설계란 : 모르겠음

6) SW 형상관리란

- SW 형상 관리 : 개발 및 유지보수 과정에서 발생하는 코드와 문서등에 대한 형상을 만들고 이를 관리하는 것

= 코드의 변경 이력과 롤백 그리고 다수 개발자들의 작업에 의해 발생기는 충돌을 처리하기 위함

- 네트워크와 인프라

1) GET과 POST의 차이점

- 위의 웹 HTTP Request에서 설명함

2) TCP/IP, UDP 프로토콜 설명

\* TCP/IP중 IP는 3계층이므로 TCP / UDP 프로토콜을 설명하고 차이를 말하는 것으로 보임

- 둘 모두 데이터를 전송하기 위한 전송 계층의 프로토콜

- TCP : Transmission Control Protocol

- UDP : User Datagram Protocol

\* TCP는 이름에서 알 수 있듯, 전송을 위한 연결 지향형이며 ACK신호를 이용하여 정상 수신되었는지 그리고 전송에 있어 혼잡 제어를 수행하며, 데이터의 순서를 보장한다.

\* UDP는 연결 설정이 없으며, 혼잡제어와 전송 순서에 대한 보장을 하지 않아 패킷 손실이 발생할 수 있으나, 빠르게 전송할 수 있는 특징이 있다.

3) MTTR, MTTF, MTBF에 대한 설명

- MTTR : Mean time to Repair 평균 수리 시간

- MTTF : Mean time to Failure 평균 고장 시간

- MTBF : Mean time between Failure 고장간 평균 시간

= [1 고장 /4 정상 /5 고장]과 같을 때 MTTR : 1, MTTF : 4, MTBF : 5

4) 파티션 테이블이 필요한 이유?

- 데이터 베이스인거 같은데.. 적합한 내용을 찾지 못함

5) OSI 7 계층을 나눈 이유

- 물리 / 링크 / 네트워크 / 전송 / 세션 / 표현 / 어플리케이션 계층

- 개발에 있어서 단계별 파악이 가능하고 개발에 있어 편리함, 일반적인 계층 구조를 만든 이유와 동일

= 물리 계층 : 데이터를 전송하는 기계적 전기적 방법

= 데이터 링크 계층 : 포인트 투 포인트 정보 전달을 위하여 MAC주소 및 CRC기반 오류 제어, 흐름 제어

= 네트워크 계층 : 데이터를 목적지까지 전달하기 위한 주소와 경로 설정

= 전송 계층 : 엔드 투 엔드 데이터 통신을 위한 계층

= 세션 계층 : 통신을 위한 논리적 연결을 관리하는 계층

= 표현 계층 : 데이터에 대한 MIME 인코딩이나 암호화 작업등으로 데이터의 포맷을 구분하는 계층

= 응용 계층 : 프로그램 의존적인 프로토콜 HTTP등

6) TCP 3 hand shaking이란?

- TCP 통신 연결을 수립할 때 수행하는 통신으로 아래와 같이 수행하며, 서로의 전송 번호를 교환하면서 싱크를 맞추는 과정

= C : SYNC[a] - > S

= C <- SYNC[a+1], ACK[b] : S

= C : ACK[b+1] -> S

\* 4 way Handshake는 TCP 연결을 끝낼 때 아래와 같이 수행한다.

= C : FIN -> S

= C <- ACK : S

= C <- FIN : S

= C : ACK -> S

7) L2, L3, L4 차이점

- 네트워크 장비에서 L2, L3, L4는 장비가 인식할 수 있는 계층 단위를 뜻한다.

= L2는 링크 계층까지 MAC기반 (일반스위치)

= L3는 네트워크 계층까지 IP기반 (허브)

= L4는 전송 계층까지이며 TCP/UDP까지 확인하여 스위칭하는 장비이다. L4의 경우 패킷에 따라서 로드밸런싱을 수행할 수 있다.

8) 허브와 스위치의 차이점

- 가장 큰 차이점은 MAC 주소를 알고 있나의 여부. 허브의 경우 장비간 연결만 수행하며 별도의 MAC을 관리하지 않아 데이터 수신시 모든 장비에게 데이터를 전송한다. 반면 스위치의 경우 MAC 주소 테이블을 바탕으로 해당 컴퓨터에만 전달한다.

9) 파이썬과 다른 언어의 차이점

- 뜬금 없이?

10) 데이터 계층의 PDU는 무엇인가?

- PDU는 프로토콜 데이터 유닛으로 데이터를 전송할 때 데이터에 붙이는 제어 정보이다.

= 물리 계층 : 비트(스트림)

= 링크 계층 : 프레임

= 네트워크 : 패킷 혹은 데이터 그램

= 전송계층 : 세그먼트

= 그 상위 : 메시지, 데이터

11) 시퀀스 넘버에 대한 역할

- TCP 에서 시퀀스 번호는 데이터를 송신한 대상이 세그먼트별로 부여한 번호로, 데이터의 순서를 제어하기 위한 용도로 사용된다.

12) 슬라이딩 윈도우의 역할

- TCP 통신시에 데이터를 전송하고 정상 수신되었다는 ACK 메시지를 대기하는데, 매번 전송 패킷에 대한 응답을 기다리는 stop and wait방식이 아니라, 윈도우 크기만큼 데이터를 전송하고 ACK를 받은 세그먼트가 앞단에 존재시 윈도우를 움직여 다음 세그먼트를 보내거나 재전송한다.

13) HTTP 응답코드의 종류를 아는대로 설명

- 200 : 정상 처리

- 300 : 리다이렉션

- 400 : 클라이언트 오류

- 500 : 서버 오류

= 200 : OK

= 301 : 영구 이동, 302 : 임시이동, 304 : 캐시됨(데이터 변경 없음)

= 400 : 잘못된 요청, 401 : 권한 없음(인증되지 않음), 403 : 금지, 429 : 너무 많은 요청

= 500 : 내부 오류, 503 : 서버 다운

14) MIME 프로토콜 설명

- 전자메일의 경우 7비트 아스키로 전달되어, 다른 데이터를 전송하기 위하여 데이터의 타입을 나태낸 것.

= text/plain등과 같이 다양한 타입들이 있음

출처: https://jcdgods.tistory.com/401 [IT 방랑기]