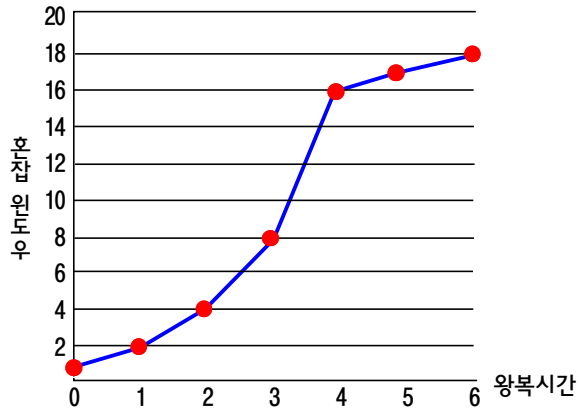
16. 그림은 TCP Tahoe에서 데이터 전송에 따른 혼잡 윈도우(cwnd, 단위: MSS)의 크기 변화를 나타낸다. 혼잡 윈도우값이 18일 때의 전송에서 Time-out이 발생했을 때, 느린 출발(slow-start) 임곗값과 혼잡 윈도우값 변화로 옳은 것은? [2022년 지방 9급]



① 임곗값은 변하지 않고, 혼잡 윈도우값은 1로 감소한다.
② 임곗값이 9가 되고, 혼잡 윈도우값은 1로 감소한다.
③ 임곗값이 9가 되고, 혼잡 윈도우값은 현재 값의 반으로 감소한다.
④ 임곗값은 변하지 않고, 혼잡 윈도우값은 현재 값의 반으로 감소한다.

♤ TCP 혼잡제어 정책

AIMD	· AIMD : Additive Increase / Multicative Decrease - 직역하면 합 증가 / 곱 감소 · 네트워크에 문제가 없으면, 더 빠른 전송을 위해 혼잡 윈도우 크기를 **1**씩 증가시킴 · **혼잡 상태**가 감지되면**(무응답, 데이터 유실)**, 혼잡 윈도우 크기를 **반**(50%)으로 줄인다. · 합 증가 : 늘어날 때는 윈도우 크기 + **1** · 곱 감소 : 줄어들 때는 **0.5** × 윈도우 크기 · AIMD 방식 : 혼잡 윈도우 크기를 그래프로 그려보면 톱니 모양이 된다.
Slow Start	· 혼잡 윈도우 크기를 증가시킬 때, **지수**적으로 증가시킨다. · 혼잡이 감지되면, 혼잡 윈도우 크기를 **1**로 줄인다.**(혼잡 윈도우 크기 = 1)** · ACK가 도착할 때마다 윈도우 크기를 증가시키므로 처음에는 윈도우 크기가 조금 느리게 증가한다. 하지만, 시간이 갈수록 윈도우 크기가 점점 빠르게 증가한다.
TCP Tahoe	· TCP Tahoe는 미국 네바다주에 있는 **타호** 호수의 이름을 딴 것이다. · TCP Tahoe는 **Slow Start**로 시작한다. · 혼잡이 감지되면, 혼잡 윈도우 크기를 **반**(50%)으로 줄인다. - **AIMD 방식 적용** · 그리고, 자신의 혼잡 윈도우 크기는 **1**로 변경 - **Slow Start** 적용

· TCP 혼잡제어 정책 : Tahoe, Reno, New Reno, Cubic, Elastic-TCP 등 다양
· **ssthresh**(slow start threshold) : **임계점**(threshold)을 칭하는 단어이다.

정답 : ②

17. 다중 프로그래밍 환경에서 연속 메모리 할당 방법에 대한 설명으로 옳지 않은 것은? [2022년 지방 9급]

① 가변분할 메모리 할당은 프로세스의 크기에 따라 메모리를 나누는 것으로 단편화 문제가 발생하지 않는다.
② 가변분할 메모리 할당의 메모리 배치방법으로는 최초 적합, 최적 적합, 최악 적합 방법이 있다.
③ 고정분할 메모리 할당은 프로세스의 크기와 상관없이 메모리를 같은 크기로 나누는 것이다.
④ 고정분할 메모리 할당에서는 쓸모없는 공간으로 인해 메모리 낭비가 발생할 수 있다.

♤ **다중 프로그래밍 환경 – 연속 메모리 할당**

• **가변분할** 메모리 할당은 프로세스의 크기에 따라 메모리를 나누는 것으로 **단편화 문제가 발생하지 않는다.(×)**
→ **가변분할** 메모리 할당은 **외부단편화 문제가 발생한다.**
→ **세그먼테이션 기법**은 **가변분할** 메모리 할당이다.

// 페이지와 세그먼트

페이지	블록의 크기를 **동일하게 분할**한 경우에 '페이지'라 한다.**(고정분할)**
세그먼트	블록의 크기를 **서로 다르게 분할**한 경우에 '세그먼트'라 한다.**(가변분할)**

세그먼테이션 기법 | (가변분할)	① **세그먼테이션** 기법에서는 **외부단편화가 발생할 수 있다.** ② 세그먼테이션 기법에서는 메모리를 **동적으로 분할**한다. • 동적 분할은 반입되는 세그먼트 크기에 맞게 메모리를 분할한다는 것이다. • 세그먼테이션 기법에서 분할된 메모리를 **세그먼트**라고 한다. • 분할된 **세그먼트** 영역들은 서로 크기가 다르다. • 운행 중, 분할된 **세그먼트** 영역들은 **자유영역(공백)**으로 되돌려 질 수 있다. • **자유영역**들이 너무 작을 때, **외부단편화**가 발생할 수 있다.**(적재 불가)** ③ 세그먼테이션 기법에서 외부단편화 해결 방법은 • 자유영역들에 대해 **압축**을 수행하여, 더 큰 공간을 만들면 된다. • 압축은 비어있는 **자유영역을 한 곳으로 합치는 작업**이다. ④ **세그먼트**의 최대 크기는 정해져 있는 것은 아니다.

정답 : ①

18. 병렬 프로세서에 대한 설명으로 옳지 않은 것은? [2022년 지방 9급]

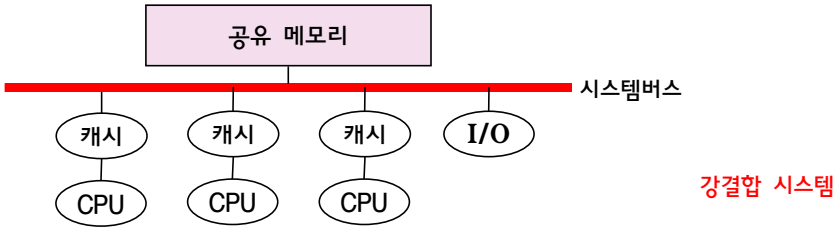
① 프로세스 수준 병렬성은 다수의 프로세서를 이용하여 독립적인 프로그램 여러 개를 동시에 수행한다.

② 클러스터는 근거리 네트워크를 통하여 연결된 컴퓨터들이 하나의 대형 멀티 프로세서로 동작하는 시스템이다.

③ 공유 메모리 프로세서(SMP)는 단일 실제 주소 공간을 갖는 병렬 프로세서를 의미한다.

④ 각 프로세서의 메모리 접근법 분류에 따르면 UMA는 약결합형 다중처리기 시스템, NUMA 및 NORMA는 강결합형 다중처리기 시스템에 해당한다.

♤ **병렬 프로세서 – MIMD**

SMP	• Symmetric MultiProcessing 약어이다. – **대칭형 다중처리** • **2**개 이상의 **프로세서(CPU)**가 **하나**의 **공유 메모리**를 사용하는 **다중 프로세서**이다.  • **대칭형** : 모든 **프로세서**가 **동등한** 위치에서 **공유 메모리**에 접근하므로
UMA	• Uniform Memory Access 약어이다. – **균등 메모리 접근** • 각 프로세서(CPU)는 **동일한** 메모리 **접근시간**을 가진다. • 균형있는 공유 메모리 접근을 제공하므로 SMP 시스템이라고도 한다.(강결합 시스템)
NUMA	• Non-uniform Memory Access 약어이다. – **비균등 메모리 접근** • 각 프로세서(CPU)는 **가변적** 메모리 **접근시간**을 가진다. – **비균등** • NUMA는 UMA 모델을 **확장**한 것이다.(**NUMA**는 **다수**의 **UMA** 모델의 **상호연결**) • NUMA는 다수의 UMA 모델들이 **상호연결망**에 의해 접속된다. • NUMA는 **전역** 및 **공유** 메모리(Global Shared Memory, GSM)도 가질 수 있다. • **접근시간** : 자신 UMA의 메모리 < 전역 공유 메모리 < 다른 UMA의 메모리 • 즉, NUMA는 자신이 속한 로컬 메모리에 접근할 때 가장 빠르다.**(당연)**
NORMA	• No-Remote Memory Access 약어이다. – **무원격 메모리 접근** • 각 프로세서들이 별도의 메모리를 가진다.**(공유 메모리 없음)** • **분산 기억장치 시스템**이라고도 함(**약결합 시스템**) • NORMA은 프로세서가 원격 기억장치는 **직접** 접근할 수 없다.(**무원격 메모리 접근**) • 어떤 노드의 프로세서가 **다른 노드 메모리의 데이터**가 필요하면, 그 노드로 메모리 **접근 요구 메시지**를 보내고, 메시지를 받은 노드는 해당 데이터를 **인출하여 보내줌**

정답 : ④

19. 다음 C 프로그램의 실행 결과로 옳은 것은? [2022년 지방 9급]

```
-----------------------------------------
 #include <stdio.h>
 int star = 10;
 void printStar() { printf("%d \n", star); }
 int main()
 {
   int star = 5;
   printStar();
   printf("%d \n", star);
   return 0;
 }
-----------------------------------------
```

① 5
   5

② 5
   10

③ 10
   5

④ 10
   10

♤ C 프로그램

```
// 외부변수와 자동변수
 #include <stdio.h>
 int star = 10;                              // 외부변수(전역변수)
 void printStar() { printf("%d \n", star); }  // 출력 : 10
 int main()
 {
   int star = 5;                             // 자동변수(지역변수)
   printStar();
   printf("%d \n", star);                    // 출력 : 5
   return 0;
 }
```

정답 : ③

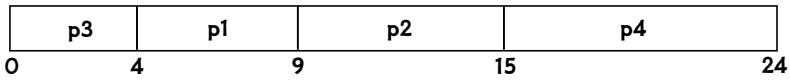
20. 다음과 같이 P1, P2, P3, P4 프로세스가 동시에 준비 상태 큐에 도착했을 때 SJF(Shortest Job First) 스케줄링 알고리즘에서 평균반환시간과 평균대기시간을 바르게 연결한 것은? (단, 프로세스 간 문맥교환에 따른 오버헤드는 무시하며, 주어진 4개의 프로세스 외에 처리할 다른 프로세스는 없다고 가정한다) [2022년 지방 9급]

프로세스	실행시간
P1	5
P2	6
P3	4
P4	9

	평균반환시간	평균대기시간
①	6	6
②	6	7
③	13	6
④	13	7

♤ SJF(Shortest Job First) 스케줄링

// 간트 차트를 그리면 다음과 같다.



프로세스	실행시간	대기시간	반환시간
P1	5	4 - 0 = 4	5 + 4 = 9
P2	6	9 - 0 = 9	6 + 9 = 15
P3	4	0 - 0 = 0	4 + 0 = 4
P4	9	15 - 0 = 15	9 + 15 = 24

- **대기시간 = 다른 프로세스가 작업한 시간 − 도착시간**
- **반환시간 = 실행시간 + 대기시간**

- 평균대기시간 $= \frac{4+9+0+15}{4} = 7$
- 평균반환시간 $= \frac{9+15+4+24}{4} = 13$

정답 : ④