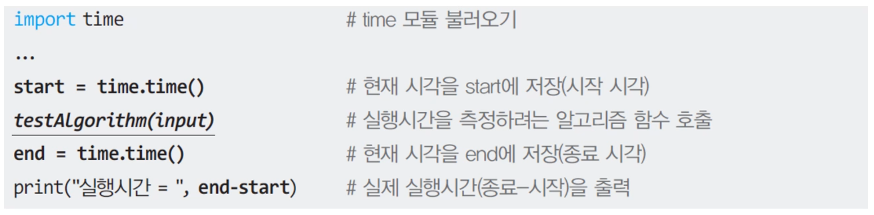
**좋은 알고리즘은?**

・시간 효율성(Time efficiency)

・공간 효율성(Space efficiency)

**2.1 효율성 분석의 기초**

・실제 실행시간 측정 방법 (파이썬)  


* 문제점은?  
  ・반드시 구현?  
  ・같은 조건의 실행시간?  
  ・소프트웨어 환경, 사용한 언어?  
  ・모든 데이터에 대해?

・절대적인 시간 측정 -> 이론적인 복잡도 분석

**알고리즘 복잡도 분석에서 중요한 점**

・알고리즘에서 입력의 크기는 무엇인가?

・복잡도에 영향을 미치는 가장 핵심적인 기본 연산은 무엇인가?

・입력의 크기가 증가함에 따라 처리시간은 어떤 형태로 증가하는가?

・입력의 특성에 따라 알고리즘 효율성에는 어떤 차이가 있는가?

**입력의 크기**

・알고리즘의 효율성은 입력 크기의 함수 형태로 표현

・무엇이 입력의 크기를 나타내는지를 먼저 명확히 결정

・예

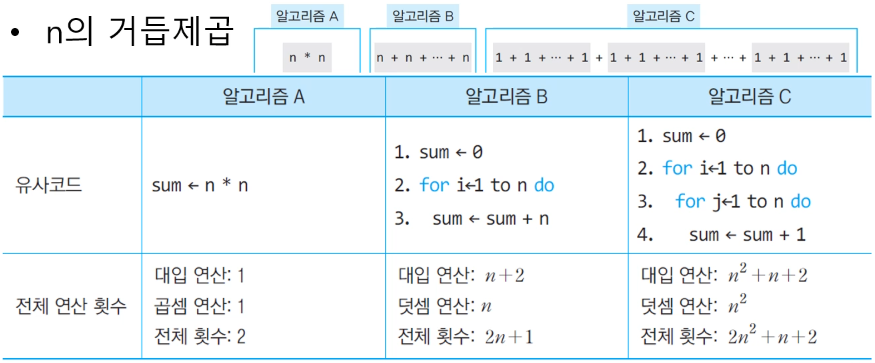
* 리스트에서 어떤 값을 찾는 문제
* x의 n 거듭제곱
* 다항식의 연산
* Row x Col의 행렬 연산
* 그래프 연산: 인접 행렬 표현/인접 리스트 표현

**실행시간 측정의 단위(기본 연산)**

・기본 연산(basic operation)

* 알고리즘에서 가장 중요한 연산
* 이 연산이 실행되는 횟수만을 계산
* 예) 다중 루프의 경우 가장 안쪽 루프에 있는 연산

・n의 거듭제곱



**복잡도 함수와 증가 속도** Time

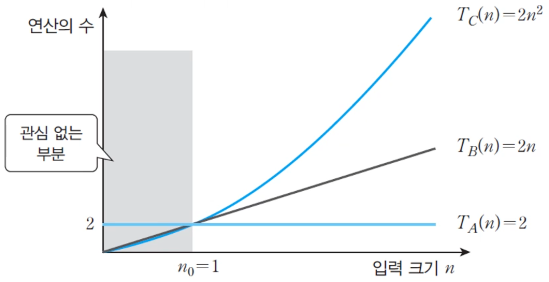
・복잡도 함수



・n이 작은 경우: 예 n=1

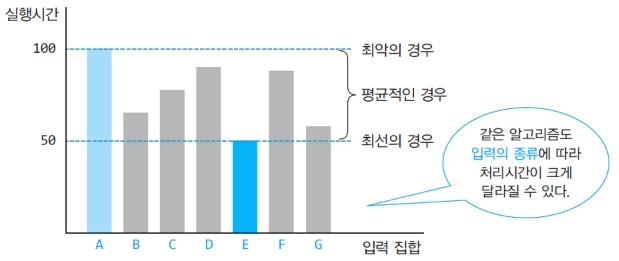


・n이 충분히 큰 경우에만 관심 있음



**최선, 최악, 평균적인 효율성**

・입력의 종류 또는 구성에 따라 다른 특성의 실행시간

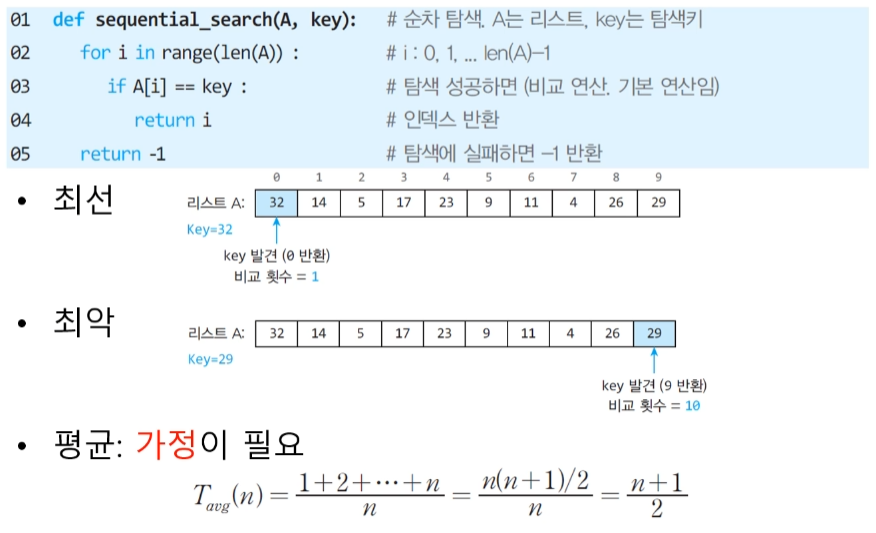


・최선의 경우(best case): 실행시간이 가장 적은 경우를 말하는데, 알고리즘 분석에서는 큰 의미가 없다.

・평균적인 경우(average case): 알고리즘의 모든 입력을 고려하고 각 입력이 발생할 확률을 고려한 평균적인 실행시간을 의미하는데 정확히 계산하기가 어렵다.

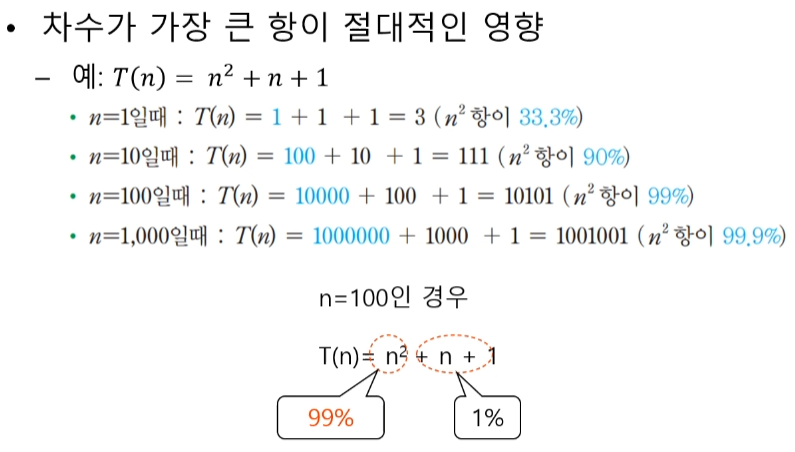
・최악의 경우(worst case): 입력의 구성이 알고리즘의 실행시간을 가장 많이 요구하는 경우를 말하는데, 가장 중요하게 사용된다.

**예) 순차 탐색**



**2.2 점근적 성능 분석 방법**

・차수가 가장 큰 항이 절대적인 영향

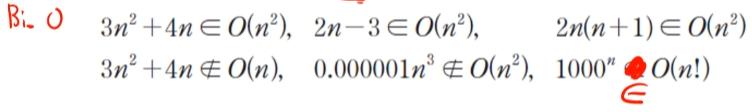


**점근적 표기(asymptotic notation)**

・n이 무한대로 커질 때의 복잡도를 간단히 표현

* 복잡도 함수를 최고차항만을 계수 없이 취해 단순화 함

・빅오: 복잡도 함수의 상한



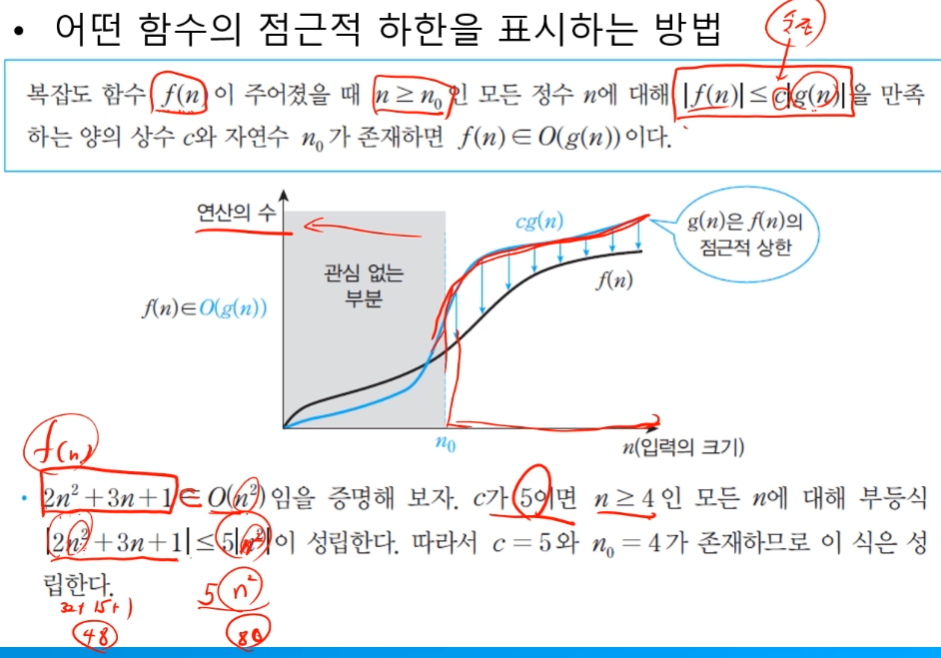
・빅오메가: 하한



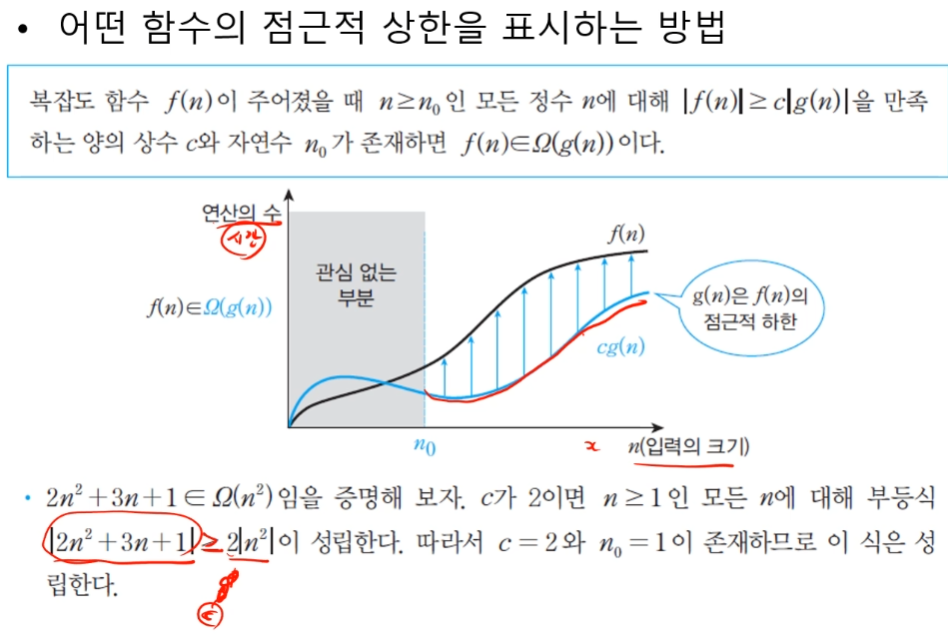
・빅세타: 상한인 동시에 하한



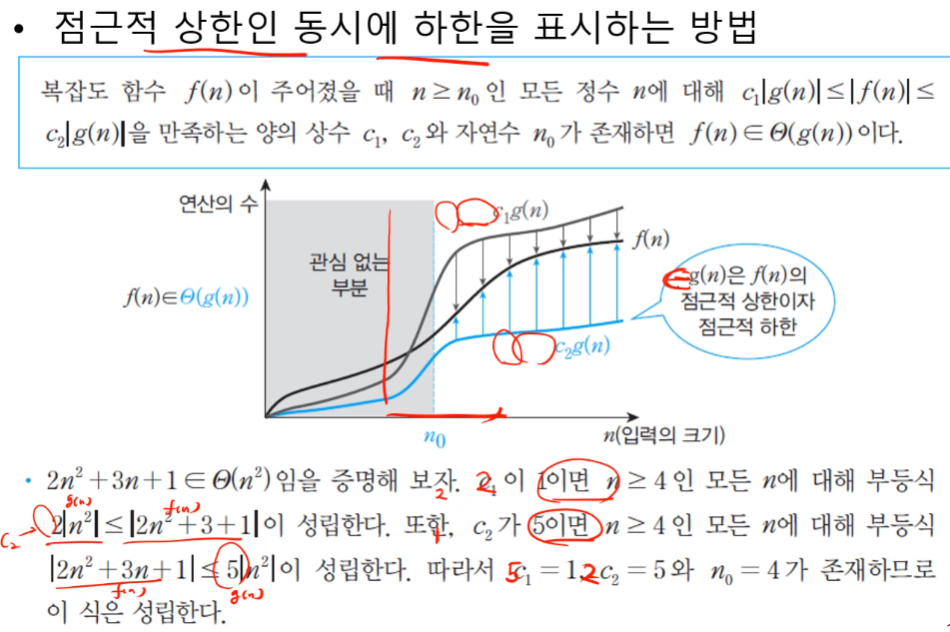
**빅오 표기법**

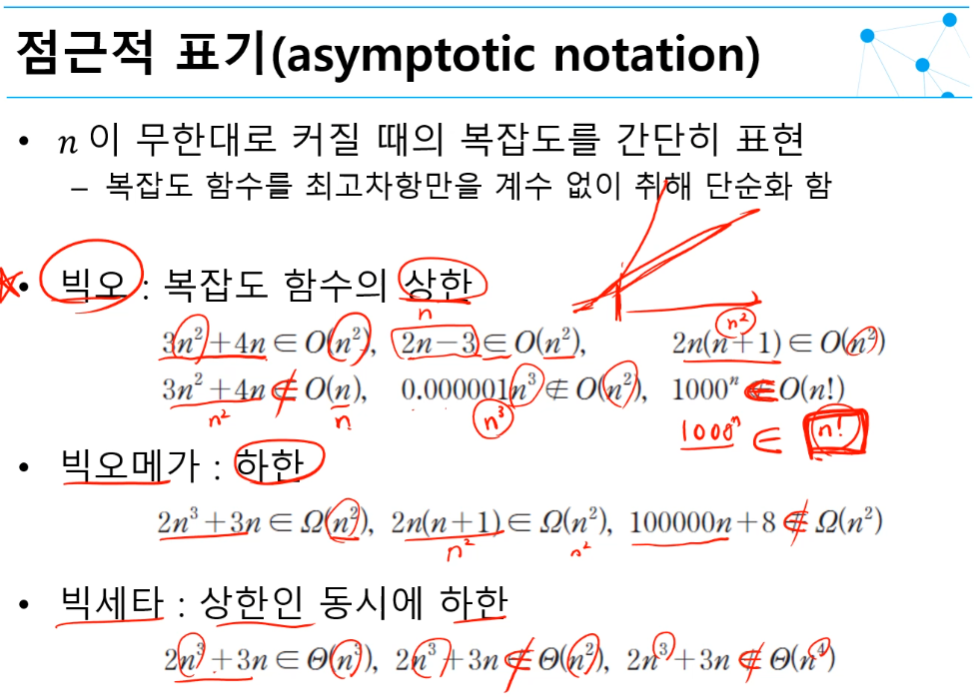


**빅오메가 표기법**

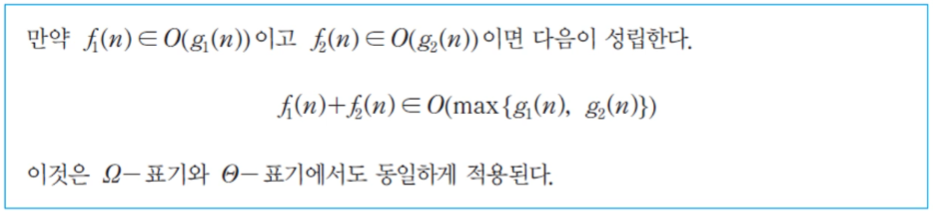


**빅세타 표기법**



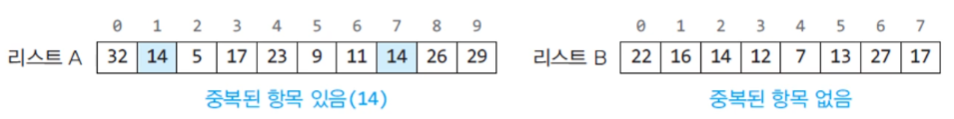


**다단계 알고리즘의 복잡도**



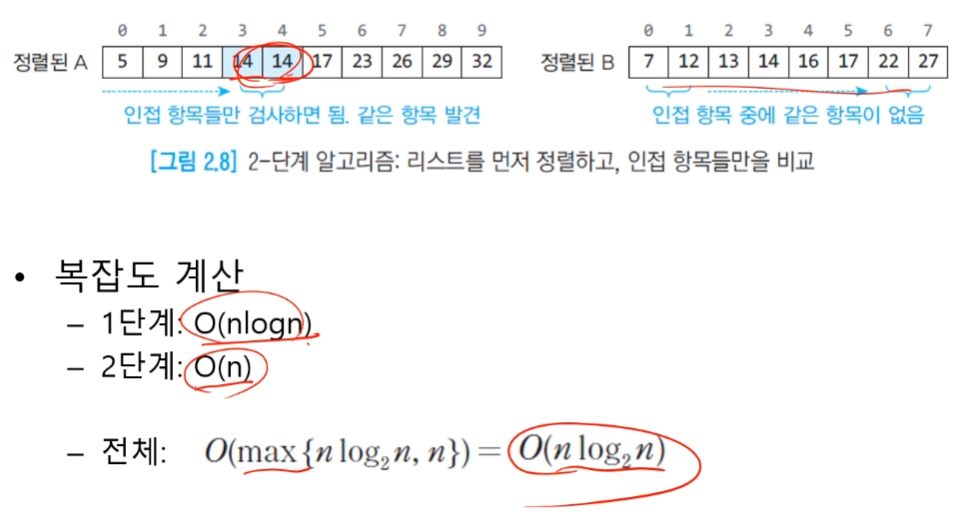
* 증명? 심화학습

・예) 리스트의 중복 항목 검사



* 알고리즘 A: 이중 루프 사용
* 알고리즘 B: 2-단계 알고리즘 -> 정렬 + 단일 루프

**2단계 알고리즘**



**점근적 성능 클래스들**



**2.3 복잡도 분석 예: 반복 알고리즘**

(1) 입력의 크 기를 나타내는 파라미터를 결정한다.

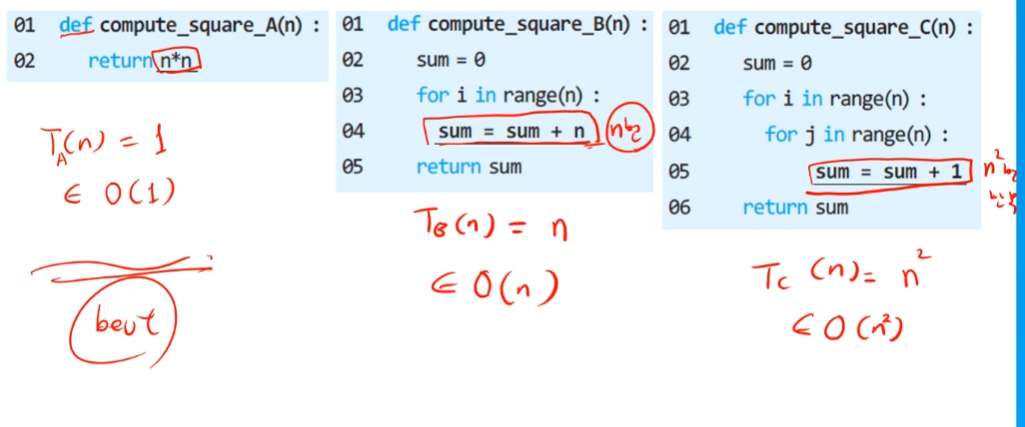
(2) 기본 연산을 찾는다. 보통 반복 루프의 가장 안쪽에 있다.

(3) 연산의 횟수가 입력 크기에 의해서만 결정되는지 살핀다. 만약 입력의 종류에 따라서도 다를 수 있다면 최선, 최악, 평균의 경우에 대해 독립적으로 복잡도를 분석해야 한다.

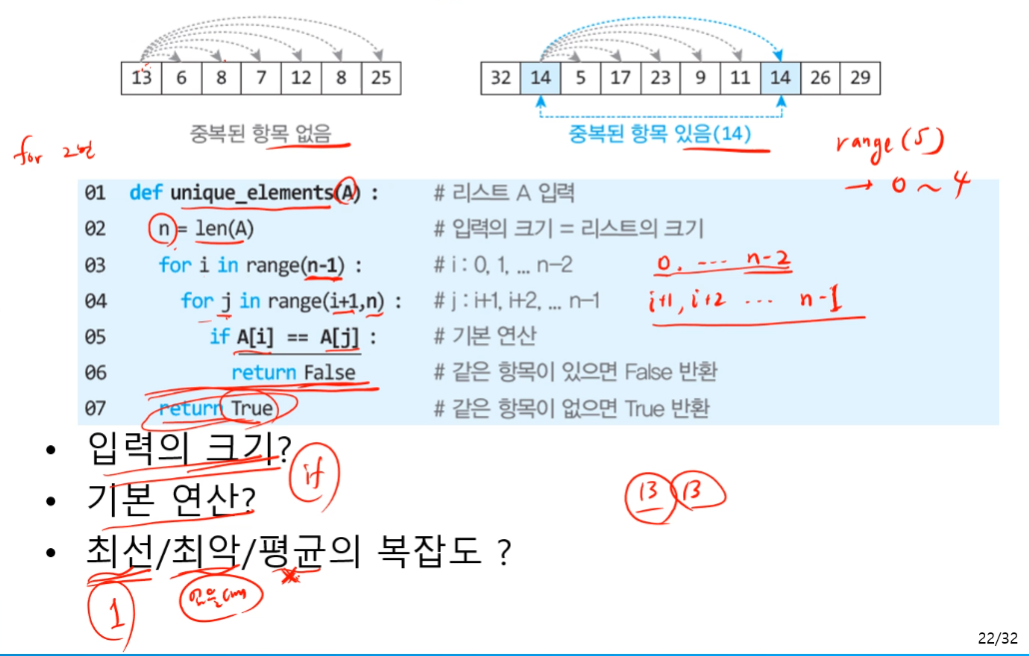
(4) 기본 연산의 전체 실행 횟수를 구하는 복잡도 함수 T(n)을 구한다.

(5) 알려진 공식 등을 이용해 T(n)을 풀고, 증가 속도를 계산한다.

**자연수의 제곱 계산**



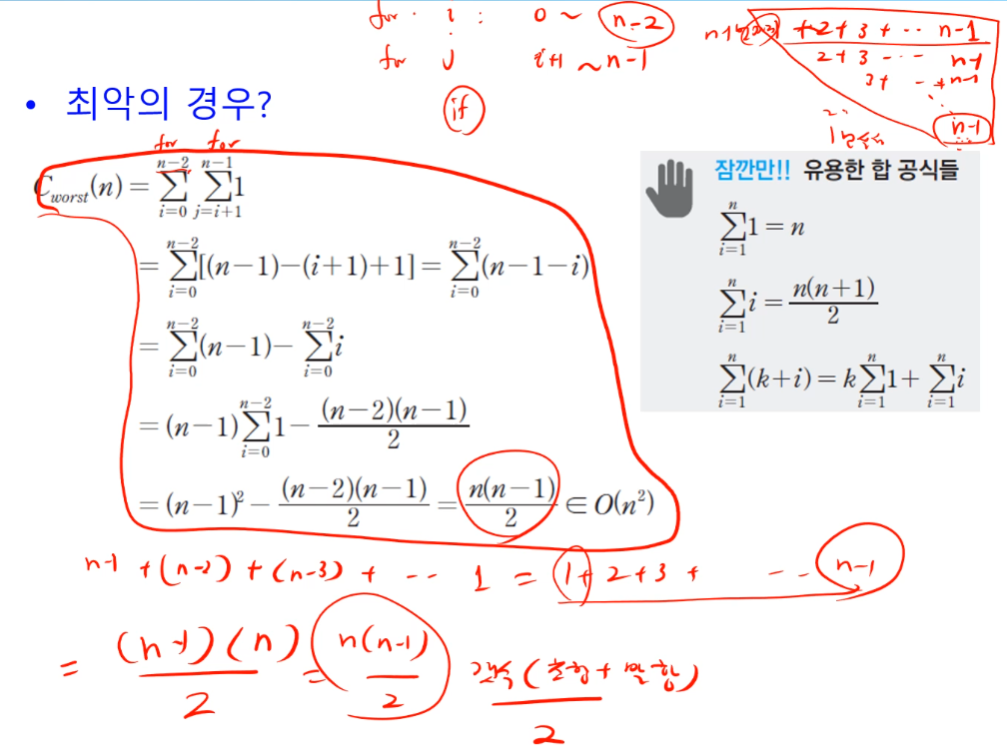
**리스트의 중복 항목 탐색**



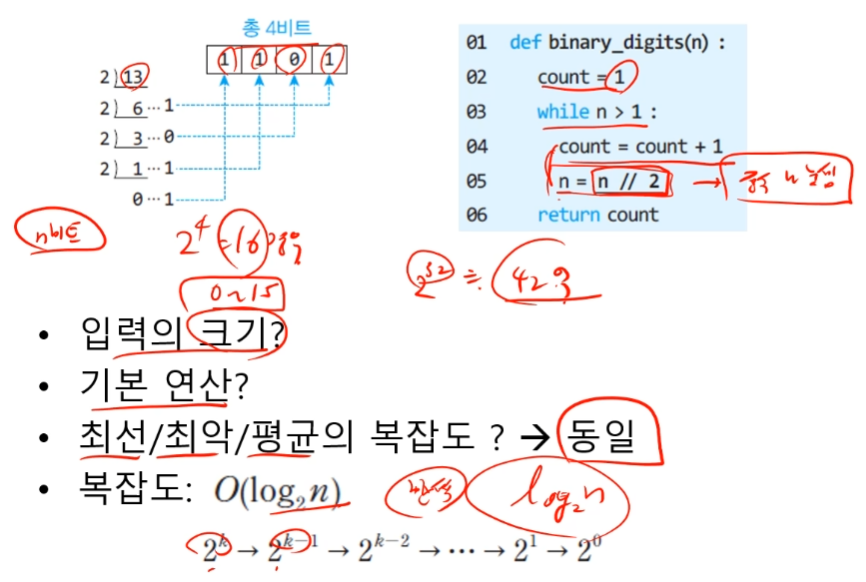
・입력의 크기?

・기본 연산?

・최선/최악/평균의 복잡도?



**자연수의 2진수 변환 시 비트 수 (반복 구조)**



**2.4 복잡도 분석 예: 순환 알고리즘**



