1. 동적 계획 알고리즘
   1. 동적 계획 알고리즘: 최적화 문제를 해결.  
      \*최적화 문제: 하나 이상의 해답 후보와 최종적인 해다 하나를 가진 문제
      1. 분할 정복 알고리즘과 차이점: 분할 정복은 하향식 문제풀이로, 부분 문제가 오로지 그 부분 문제의 직계 상위 문제를 풀기 위한 답일 뿐이다. 반면 동적 프로그래밍은 상향식 문제 풀이로, 직속이 아니더라도 상위 문제를 푸는 단초로 사용된다(부분 문제간 의존적 관계, 즉 함축적 순서가 존재).
   2. 모든 쌍 최단 경로
      1. 다익스트라 알고리즘: 시간복잡도
      2. 플로이드-워샬(플로이드) 알고리즘: 시간복잡도 . 다익스트라보다 간단하여 효울적, 다익스트라는 기점을 기반으로 찾지만, 플로이드는 한 번에 쭉 훑어내림.
   3. 연속 행렬 곱셈
      1. A \* B = C: A의 행과 B의 열의 수는 같아야 한다. C는 A의 행과 B의 열의 크기를 가진다. 결합 법칙은 성립되지만, 교환 법칙(이웃하는 행렬끼리만 곱 가능)은 성립하지 않는다. 행렬 곱셈의 연산 횟수는 A의 행(이자 B의 열) \* A의 열 \* B의 행이다.
      2. A \* B \* C \* D에서 (A \* B) \* C \* D와 (A \* B) \* (C \* D), (A \* B \* C) \* D와 같이 인접한 행렬과의 곱을 비교하여 최소값을 찾는다.
   4. 편집 거리 문제: 삽입, 삭제, 대체 연산을 사용하여 문자열을 수정하여 변환하고자 할 때 필요한 최소의 편집 연산 횟수
      1. S와 T의 글자 수만큼의 테이블을 만든다. 첫 줄의 가로 세로는 인덱스와 같은 숫자로 깔아둔다.
      2. 세로는 삭제, 가로는 삽입, 대각선은 대체 연산으로 만약 글자가 같으면 대체 연산을 그대로 가져온다. 그 외의 경우에는 세로, 가로, 대각선 중 가장 작은 값에 +1 한다.
2. 정렬
   1. 정렬의 종류
      1. 내부 정렬: 주기억 장치보다 크지 않은 경우 수행되는 정렬  
         \*알고리즘 성능(좋아지는 순): 버블– 선택 – 삽입(제곱) – 쉘(1.5 제곱) - 힙 – 합병 – 퀵(n logn)   
         (버선발로 삽입 쉘로 힙합은 퀵)
      2. 외부 정렬; 보조 기억 장치에 있는 입력을 여러 번 나누어 정렬하여 저장 (다방향 합병, 다단계 합병)
      3. 안정성: 동일한 데이터에 대해서 입력될 때의 상대적 순서가 보장되는 것 (삽입, 버블, 합병)
   2. 버블 정렬: 이웃하는 숫자를 비교하여 작은 수를 앞쪽으로 이동시킴. 시간복잡도 이다. 정렬이 되고 나서도 반복이 계속된다는 점에서 비효율적이다.
   3. 선택 정렬; 최솟값을 선택하여 배열의 처음부터 채워나가는 방식. 입력에 민감하지 않고, 항상 일정한 시간 복잡도 을 가진다. 원소 간의 자리바꿈 횟수가 최소이다.
   4. 삽입 정렬: 배열을 정렬된 앞 부분과 정렬 안 된 뒷부분으로 나누고, 정렬 0안 된 부분의 가장 왼쪽 원소를 정렬된 부분의 적절한 위치에 삽입하여 정렬. 시간복잡도 , 입력이 이미 정렬된 경우 의 시간복잡도. 거의 정렬된 입력, 입력의 크기가 작을 때 효율적이다.
   5. 쉘 정렬: 간격을 이용한 선택 정렬. 시간복잡도 . 가장 좋은 간격이 확실하지 않기 때문에 쉘 정렬의 시간 복잡도는 절대적이지 않다. 입력 크기가 작을 때 좋은 성능, 임베디드 시스템에서 주로 사용.
   6. 힙 정렬: 시간 복잡도
   7. 비교 정렬: 버선삽쉘힙합퀵과 같이 숫자 대 숫자로 비교 (기수 정렬은 부분 비교)
      1. 정렬 문제의 하한: 시간 복잡도의 하한은 가장 빠른, 최소 시간 풀이
      2. 비교 정렬은 최댓값의 하한은 n-1, 이진 트리 정렬의 하한은
   8. 기수 정렬: 제한된 범위의 숫자를 각 자릿수 별로 정렬 (안정성 보장됨). 시간 복잡도
      1. LSD(Right to Left, RL) 정렬: 1의 자리 – 10의 자리 – 199의 자리 순으로 정렬. 작은 자리 수부터 정렬하는 것
      2. MSD(LR) 정렬: 큰 자리 수부터 정렬하는 것. 100의 자리 – 10의 자리 – 1의 자리
3. NP 완전 문제
   1. 알고리즘 문제의 종류
      1. P 문제: 현실적인 시간동안 풀 수 있음. 7장 이전의 모든 문제. N의 다항식에 비례하는 문제 집합.
      2. NP 문제: 현실적인 시간동안 풀 수 없음. 근사값으로 해를 구함. 해결이 가능한지, 불가능한지도 증명되지 않음. P와 NP-완전 문제를 포함하는 집합.
      3. NP 완전 문제: 현실적인 시간동안 풀 수 없지만, 문제들 사이에 강력한 논리적 연관성을 가진 완전 논리 집합이 있음. 다항식 시간보다 큰 복잡도를 가진 알고리즘으로 해결됨. 같은 집합에 속한 문제는 하나라도 다항식 시간에 풀 수 있다면, 전부 다항식 시간에 풀 수 있다.
      4. 결정 문제: 맞다, 아니다로 답하는 문제. 보통 NP문제의 추측한 해를 다항식 시간 내에 확인