Greedy Algorithm

정의: 문제 해결 과정에서 순간마다 최적이라고 생각되는 결정을 하는 방식

동적계획법보다 효율적이지만, 반드시 최적 해를 보장하진 않는다.

1. Greedy choice property: 앞의 선택이 이후 선택에 영향을 주지 않음.
2. Optimal substructure: 문제 전체에 대한 최적해가 부분 문제에 대해서도 최적해가 됨.

두 속성을 만족할 경우, 그리디 알고리즘을 통해 최적해를 얻을 수 있다.

문제 푸는 단계:

1. Select(해 선택): 당시에 최적인 해를 구한 뒤, 부분 해 집합에 추가
2. Feasible(해 검사): 새로운 부분 해 집합이 적절한지 검사
3. Union: 새로운 부분 해 집합이 문제의 해가 되는지 검사

대표 예제:

1. 최소 개수 동전 지불

Select: 가장 큰 동전을 우선으로 거스름돈을 구성하고 부분 해 집합에 추가

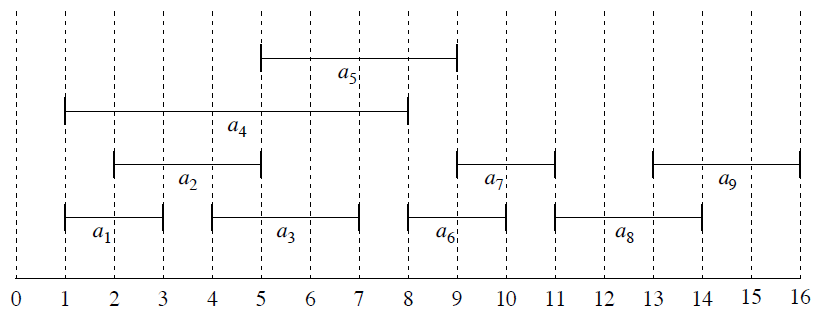
Feasible: 부분 해 집합이 거스름돈 총 금액을 초과하는지 검사

Union: 부분 해 집합이 거스름돈 총액과 일치하는지 검사

1. Activity-Selection Problem

한정된 교실 공간 내에서 최대 수업을 배정하는 문제. 종료 시간을 기준으로 오름차순 정렬하여 문제 해결. 종료 시간이 가장 빠른 수업을 먼저 배정하여 시간이 겹치지 않도록 부분 해를 구성한다. 이 문제는 최적해를 얻기 위한 그리디의 속성을 만족하므로, 항상 최적해를 보장한다.

| i | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| si | 1 | 2 | 4 | 1 | 5 | 8 | 9 | 11 | 13 |
| fi | 3 | 5 | 7 | 8 | 9 | 10 | 11 | 14 | 16 |



1. 최소 스패닝 트리(최소 비용, 프림 알고리즘, 크루스칼 알고리즘): 그래프 내의 모든 정점을 최소 비용으로 연결하는 트리
2. 최단 경로(다익스트라 알고리즘): 한 정점에서 다른 정점으로 가는 최단 경로
3. 배낭 문제(Fractional, 0/1): profit/weight를 기준으로 정렬 후 Greedy하게 부분 해 집합에 추가. Fractional의 경우 최적해를 보장하지만, 0/1의 경우 최적해를 보장하지 않음.
4. 허프만 코드: 데이터, 파일 압축에 사용됨. 특정 형태의 정보를 다른 형태의 정보로 표현