

AI 알고리즘

네트워크 모형(MST, 그리디 알고리즘 등)

그리디알고리즘

그리디알고리즘 개요

그리디알고리즘개요

❖ 그리디 알고리즘이란

- 그리디
 - 탐욕스러운, 욕심많은
 - 문제를 해결하는 과정에서 매 순간 최적이라 생각되는 해답을 찾으며 이를 토대로 최종 문제의 해답에도달하는 문제 해결 방식
- 최소결침나무 알고리즘(Minimum Spanning Tree Algorithm)
 - 최소비용으로 노드와 노드를 연결
- Kruskal 알고리즘
 - 간선이 가장 적은 것부터 연결
- Prim 알고리즘
 - 만들어진 트리에 인접한 가장 가까운 정점을 하나씩 추가하여 최소신장트리를 만들
- 그리디 알고리즘은 다양한 절차와 방향을 포함
- 순간마다 하는 선택은 그 순간에 대해 지역적으로는 최적이나 최종적(전역적)인 해답을 만들었다고 해서 그것이 최적이라는 보장은 없음
- 어떤 문제는 최적해를 찾고, 어떤 문제는 못 찾는 이유?

❖ 그리디 알고리즘 적용 조건

- 탐욕적 선택 속성(Greedy Choice Property)
 - 앞의 선택이 이후의 선택에 영향을 주지 않음
- 최적부분구조(Optimal Substructure)
 - 문제에 대한 최종 해결 방법은 부분 문제에 대한 최적 문제 해결 방법으로 구성됨
 - 조건이 충족되지 않으면 최적해를 구할 수 없음
 - 순간 선택으로 최적해에도달한 것과 그렇지 못한 것이 있음
 - 조건이 성립하지 않더라도 근사해로 사용 가능

그리디알고리즘 개요

❖ 그리디 알고리즘 문제 해결 방법

- 선택 절차(Selection Procedure)
 - 현재상태에서의 최적의 해답을 선택
- 적절성 검사(Feasibility Check)
 - 선택된 해가 문제의 조건을 만족하는지 검사
- 해답 검사(Solution Check)
 - 원래의 문제가 해결되었는지 검사하고, 해결되지 않았다면 선택 절차로 돌아가위의 과정 반복

그리디알고리즘

다양한 문제 해결방법

다양한 문제 해결 방법

❖ 동전 거스름돈 문제

- 동전 거스름돈 문제의 최소 동전 수를 찾는 그리디 알고리즘
 - 남은 액수를 초과하지 않는 조건하에 '욕심내어' 가장 큰 액면의 동전을 취하는 것
 - 최적해 방법이 아니지만, 최적해가 나옴
 - 동전 체계로 인해 항상 최적해 도출
- 200원을 거슬러줘야 하는 경우
 - 현재와 같은 경우
 - 100원짜리 2개
 - 160원짜리 동전이 있는 경우
 - 160원짜리 1개
 - 10원짜리 4개

❖ 0-1 배낭 문제

- 경우에 따라 최적해와 근사해를 모두 구할 수 있음
 - 탐욕 1
 - 무게와 상관없이 가장 비싼 물건부터 넣는 방법
 - 탐욕 2
 - 가장 가벼운 물건부터 넣는 방법
 - 탐욕 3
 - 단위 무게당 가격이 가장 높은 물건부터 넣는 방법
- 무게와 상관없이 가장 비싼 물건부터 넣어보는 방법
 - 그리디 알고리즘으로는 최적해인지 아닌지 알 수 없음

물건	
3kg 60만원	20만원/kg
6kg 20만원	3.3만원/kg
5kg 10만원	2만원/kg
4kg 40만원	10만원/kg

다양한 문제 해결방법

❖ 0-1 배낭 문제 (분할 가능, Fractional Knapsack)

- 무게와 상관없이 가장 비싼 물건부터 넣어보는 방법
- 단위 무게당 가격이 가장 높은 물건부터 넣는 방법
 - 분할 가능한 배낭 문제의 경우 최적해를 구할 수 있음

❖ 집합커버 문제 (Set Cover)

- 집합 F 에서 선택하는 집합들의 수를 최소화하는 문제
 - n 개의 원소를 가진 집합 U 가 있고, U 의 부분집합들을 원소로 하는 집합 F 가 주어질 때, F 의 원소들인 집합들 중에서 어떤 집합들을 선택하여 합집합하면 U 와 같게 되는가?
- 신도시 소방서 배치 사례
- 어느 마을에 소방서를 신설해야 소방서의 수가 최소가 되는가?
 - 신도시의 마을 10개
 - $U = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$
 - S_i 는 마을 i 에 소방서를 배치했을 때 커버되는 마을의 집합
 - $F = \{S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}\}$
 - $S_1 = \{1, 2, 3\},$
 $S_2 = \{1, 2, 3, 4\},$
 $S_3 = \{1, 2, 3, 4, 8\},$
 $S_4 = \{2, 3, 4, 5, 7, 8\},$
 $S_5 = \{4, 5, 6, 7\},$
 $S_6 = \{5, 6, 7, 9, 10\},$
 $S_7 = \{4, 5, 6, 7\},$
 $S_8 = \{1, 2, 4, 8\},$
 $S_9 = \{6, 7, 9\},$
 $S_{10} = \{9, 10\}$

두 집합을 합치면
전체 집합임

다양한 문제 해결 방법

❖ 집합커버 문제 (Set Cover)

- 알고리즘
 - 입력: $U, F = \{S_i\}, i=1, \dots, n$
 - 출력: 집합커버 C
 - 1. $C = \emptyset$
 - 2. while $U \neq \emptyset$
 - 3. U 의 원소를 가장 많이 가진 집합 S_i 를 F 에서 선택
 - 4. $U = U - S_i$
 - 5. S_i 를 F 에서 제거하고, S_i 를 C 에 추가
 - 6. Return C