7주차. 탐색알고리즘(Search algorithm)

개요

> *" 검색 문제를 해결하는 어떠한 알고리즘이라도 해당되며, 연속 변수나 이산 변수를 사용하여, 일부 데이터 구조 안에 저장된 정보를 검색하거나 문제 도메인의 검색 공간에서 계산을 하기 위해 사용된다 " - 위키백과 검색 알고리즘*

이산변수 vs 연속변수

- 이산변수(discrete variable)란 떨어진 값을 갖게 할 수 있는 변수를 의미.
- 연속변수(continuous variable)란 변수의 각 값 사이에 무수히 많은 또 다른 값들이 존재하는 경우를 의미.

검색 문제 분류

제약만족문제(Constraint Satisfaction Problem)

: 문제의 해가 주어진 제약조건을 만족하는 형태로 정의되는 종류의 문제. 대표적인 문제로 4색 문제나 아인슈타인 문제 등이 이에 포함. 우선순위를 정하여 하나씩 변수에 값을 대입

트리탐색문제(Tree Search Problem)

: 그래프나 트리와 같은 자료구조로 구성되는 환경에서 목적지에 도달하기 위한 경로를 구하기 위한 문제

구성

- 최초시작점
- 해 평가
- 후행함수
- 선택 소요 비용

문제를 푸는 방법

> *F(s)=g(s)+h(s)*

s는 각 상태를 말함.

F는 평가함수

g는 현재까지 지나쳐 온 행동들에 소모된 비용의 총 합

h는 추정치로서, 문제에서 추가적으로 주어진 정보를 통해서 얻을 수 있는 값

F(s)의 값이 제일 작은 s를 우선적으로 탐색하는 전략을 통해서 탁색을 수행하며, 알려져 있는 대부분의 전략은 저 수식의 어떤 방법으로 참조해서 활용하는가로 분류

종류

1. **Uninformed 전략 - 추정치 h(s)를 배제한 전략. 추가적으로 주어진 정보가 없을 때**
    a. DFS
    b. BFS
2. **Informed 전략 - 추가적인 정보를 활용하여 허용 추정치를 산정하고 활용**
    a. 탐욕 알고리즘(Greedy Algorithmn) - g(s)를 무시하고 추정치만을 이용하여 탐색하기 때문에 최적해를 보장하지 않음.
    b. A* 알고리즘
3. **메타 전략 - 여러가지 알고리즘을 동시에 사용하거나 다른 접근방법을 이용**

경쟁 문제

1. α-β Pruning

지역 탐색

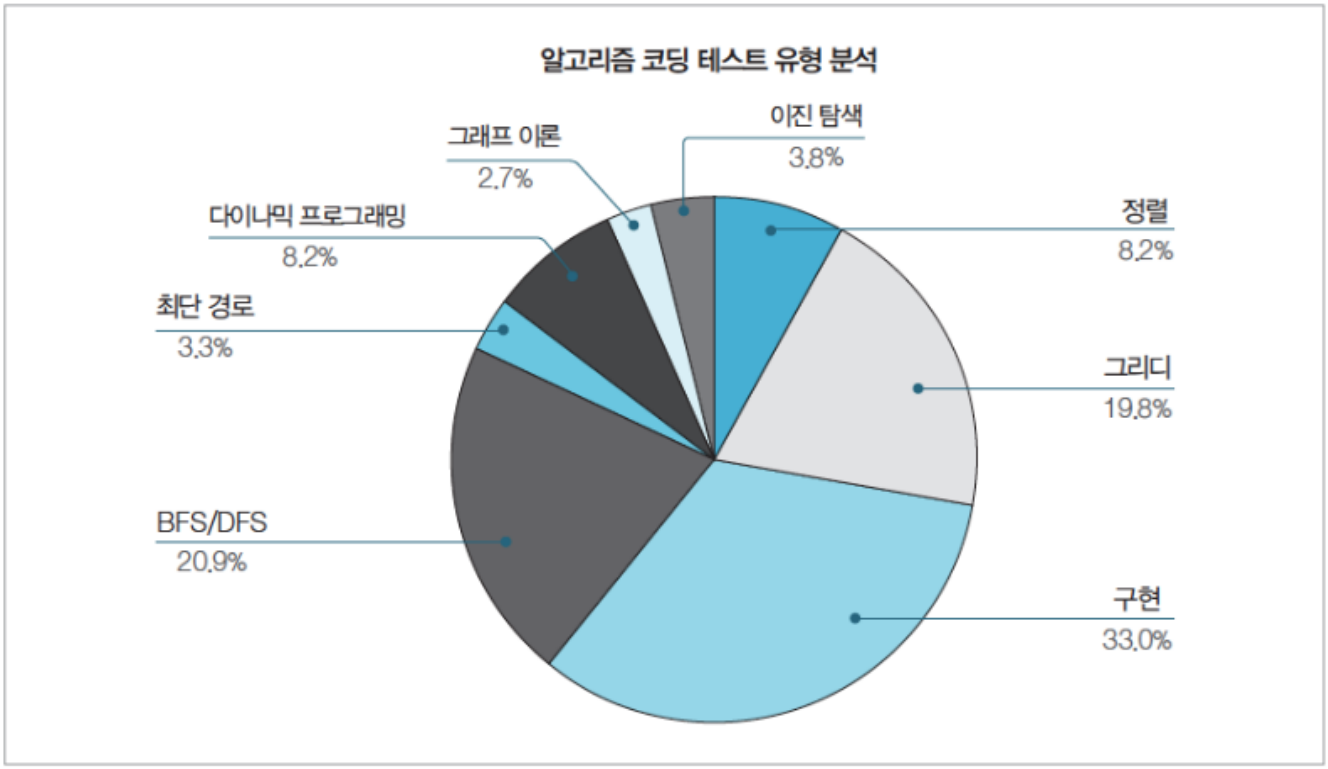
1. CS
2. GPG
3. MADS
4. SPSA

광역 탐색

1. 유전 알고리즘
2. PSO
3. 공분산 행렬 이용 진화 전략

코딩테스트 출제 유형

알고리즘 유형 분석



출처: https://www.hanbit.co.kr/channel/category/category_view.html?cms_code=CMS7793635735

자주 출제되는 탐색알고리즘 유형

- 완전탐색(Brute-Force)
    - 선형구조 - 순차탐색
    - 비선형구조
        - 깊이 우선 탐색(DFS)
            - 사용 예
                - 여행경로
                - 이동 중 가중치 추가/ 제약 존재
        - 너비 우선 탐색(BFS)
            - 특징
                - 최소 비용 문제
                - 간선의 가중치가 1이다.
                - 정점과 간선의 개수가 적다. (시간 제약, 메모리 제한 내에 만족한다.)
            - 사용 예
                - 백준 1303. 전투
                - 네트워크
                - 최단 거리 구하기
                - 웹크롤링
                - 네트워크 브로드캐스트
                - 가비지 컬렉션

샘플 문제 풀이

- 문제 : 백준 1260. DFS와 BFS

풀이

```
import sys
input = sys.stdin.readline
from collections import deque

def dfs(graph, v):
    visited = {}
    stack = [v]

    while stack:
        n = stack.pop()
        if n not in visited:
            visited.setdefault(n)
            stack += reversed(graph[n])
    return visited

def bfs(graph, v):
    visited = {}
    queue = deque([v])

    while queue:
        n = queue.popleft()
        if n not in visited:
            visited.setdefault(n)
            queue += graph[n]
    return visited

n, m, v = map(int, input().split())

graph = {i:[] for i in range(1,n+1)}
for i in range(1, m+1):
    x, y = map(int, input().split())
    graph[x].append(y)
    graph[y].append(x)

for key in graph:
    graph[key].sort()

print(' '.join(list(map(str,dfs(graph, v)))))
print(' '.join(list(map(str,bfs(graph, v)))))



dfs : stack  reverse
bfs : queue

 collection   extend +=   . (https://stackoverflow.com/questions/4176980/is-extend-faster-than)
setdefault(n) :   .
```

출처: https://ebbnflow.tistory.com/236

문제 풀이

- First Bad Version (easy)
- Find First and Last Position of Element in Sorted Array (medium)
- Network Delay Time (medium)

DFS/BFS 추가 문제 (백준)

기초

1. 바이러스 : https://www.acmicpc.net/problem/2606
2. 단지번호 붙이기 : https://www.acmicpc.net/problem/2667
3. 보물섬 : https://www.acmicpc.net/problem/2589
4. 토마토 : https://www.acmicpc.net/problem/7576
5. N-Queen : https://www.acmicpc.net/problem/9663

응용

1. 3차원 토마토 : https://www.acmicpc.net/problem/7569
2. 탈출 : https://www.acmicpc.net/problem/3055
3. 거울 설치 : https://www.acmicpc.net/problem/2151

참고문서

- 위키백과 - 검색 알고리즘
- 나무위키 - 탐색 알고리즘
- [이것이 코딩 테스트다] 1. 코딩 테스트 출제 경향 분석 및 파이썬 부수기
- [알고리즘] BFS/DFS 문제 풀이_파이썬