동적계획법

NEXTERS 알고리즘 스터디

동적 계획법

큰 의미에서 분할 정복과 같은 접근 방식

어떤 부분 문제를 두 개 이상의 문제에서 사용 할 수 있다.

중복되는 부분문제(overlapping subproblems)를 캐시에 저장하고 재 사용하여 중복 계산하지 않는다.

이항 계수 (1)

동적 계획법 알고리즘의 예로 n 개의 서로 다른 원소 중에서 r개의 원소를 순서 없이 골라내는 방법의 수

이항 계수 점화식

$$\binom{n}{r} = \binom{n-1}{r-1} + \binom{n-1}{r}$$

이항 계수 (2)

재귀 호출을 이용한 이항계수의 계산

```
int bino(int n, int r) {
        if (r == 0 || n == r) return 1;
        return bino(n - 1, r - 1) + bino(n - 1, r);
}
```

이항 계수 (3)

함수의 중복 호출 수는 n과 r이 커짐에 따라 기하급수적으로 증가

예) bino(8, 4)를 계산하기 위해 bino(1,0)과 bino(1,1)을 굉장히 여러 번 반복 호출

캐시배열을 만들어서 반복되는 부분을 최적화

Memoization (1)

함수의 결과를 저장하는 장소를 마련해 두고, 한 번 계산한 값을 저장해 뒀다 재 활용하는 최적화 기법

메모이제이션을 이용한 이항 계수의 계산

```
int cache[30][30]

int bino(int n, int r) {
    if (r == 0 || n == r) return 1;
    if (cache[n][r] != -1)
    return cache[n][r];
    return cache[n][r] = bino(n - 1, r - 1) + bino(n - 1, r);
}
```

Memoization (2)

참조적 투명성(referential transparency) : 입력이 고정 되어있을때 그 결과 값이 항상 같은 함수

참조적 투명성이 보장되는 함수에만 메모이제이션 기법을 사용 할 수 있다.

항상 같은 형태를 두고 구현 하는 방법이 추천 됨

Memoization (3)

메모이제이션 구현 예

```
int cache[2500][2500]

int someObscureFunction(int a, int b) {
     if (...) return ...;
     int& ret = cache[a][b];
     if (ret != -1) return ret;
     ...

     return ret;

}

int main() {

          memset(cache, -1, sizeof(cache));

}
```

외발 뛰기 문제 (1)

N x N 의 격자에서 1~9까지의 정수가 있고, 오른쪽이나 아래쪽으로만 이동 가능할 때 게임 판의 끝까지 도달 할 수 있는지를 알아 보는 문제

문제 링크

https://algospot.com/judge/problem/read/JUMPGAME

외발 뛰기 문제 (2)

외발 뛰기 문제를 해결하는 재귀 호출 알고리즘

```
int n, board[100][100];

int jump(int y, int x) {
    if (y >= n || x >= n) return false;
    if (y == n - 1 && x == n - 1) return true;
    int jumpSize = board[y][x];
    return jump(y + jumpSize, x) || jump(y, x + jumpSize);
}
```

외발 뛰기 문제 (3)

외발 뛰기 문제를 해결하는 동적 계획 알고리즘

```
int n, board[100][100];
int cache[100][100];
int jump2(int y, int x) {
    if (y >= n || x >= n) return 0;
    if (y == n - 1 && x == n - 1) return 1;
    int & ret = cache[y][x];
    if (ret != 1) return ret;
    int jumpSize = board[y][x];

return ret = jump2(y + jumpSize, x) || jump2(y, x + jumpSize);
}
```

와일드 카드 (WILDCARD)

와일드 카드 패턴을 주어진 글자와 대응되는지 검사한다.

‘?’ 어떤 한 글자와도 대응된다. ‘*’ 어떤 여러 글자와도 대응된다.

✓문제

https://algospot.com/judge/problem/read/WILDCARD

✓풀이코드

https://github.com/Nexters/algorithmStudy/blob/master/seokjoong/Chapter08/Wildcard.cpp

https://github.com/Nexters/algorithmStudy/blob/master/Hsue/WILDCARD/wild.py

원주율 외우기 (PI)

숫자들의 조각이 주어질 때, 정해지 난이도 표에 따라 최소의 난이도를 계산 하는 문제

3글자, 4글자 ,5글자를 제외한 나머지의 최적해 를 재귀적으로 구한다.

✓문제

https://algospot.com/judge/problem/read/PI

✓풀이코드

https://github.com/Nexters/algorithmStudy/blob/master/seokjoong/Chapter08/PI.cpp

https://github.com/Nexters/algorithmStudy/blob/master/yongseongkim/chapter_8/pi.py

폴리오미노 (POLY)

N개의 정사각형으로 구성된 세로 단조 폴리오미노의 개수를 세는 프로그램을 작성

첫 줄에 first개의 정사각형이 있고, 나머지 사각형으로 만든 폴리오미노의 첫 줄에 second 정사각형 이 있을 때 이들을 붙일 수 있는 방법은 first + second – 1

✓문제

https://algospot.com/judge/problem/read/POLY

✓풀이코드

https://github.com/Nexters/algorithmStudy/blob/master/seokjoong/Chapter08/Poly.cpp

https://github.com/Nexters/algorithmStudy/blob/master/Hsue/POLY/poly.cpp

참고 자료

알고리즘 문제 해결전략

http://algospot.com