week3

2024-07-18

외판원 순회

간단한 방법부터 생각

n! 로 풀 수 있다 .

모든 경로를 만들어 보고

가능한지 확인한 후 가능하다면 , 총 시간을 더해서 비교한다 .

외판원 순회

n! 은 굉장히 빨리 증가하는 함수이다
15! 는 약 1 조 3 천억 정도 되는 수 .

알고리즘 문제를 풀 때는 1 초에 1 억번 연산이 가능하다고
가정하므로 너무나도 큰 시간복잡도임

문제에선 N <= 16 이므로 n! 로는 풀 수 없다 .

외판원 순회

시작 위치는 중요하지 않다 .
N = 5 이고 , 최적의 경로가 [2, 4, 1, 3, 5] 이라면

[2, 4, 1, 3, 5]
[4, 1, 3, 5, 2]
[1, 3, 5, 2, 4]
...
전부 정답이고 같은 결과가 나온다 . 원형 큐를 생각하면 됨 .
따라서 우리가 시작 위치를 고정할 수 있다 .

외판원 순회

시작 위치를 1 로 고정한다면
경우의 수는
(n – 1)! 이 된다 .

그럼에도 너무 오래 걸리기 때문에

시간을 더 줄일 방법을 생각해야 함

외판원 순회

DP, 메모이제이션을 활용해 필요 없는 계산을 줄일 수 있다 .

=> 불필요한 계산을 줄이기

외판원 순회

(N-1)! 로 계산한다고 해보자 .

[1, 2, 3, 4....]

N = 10 이라면 .... 에는 (5, 6, 7, 8, 9, 10)

-> [1, 3, 2, 4....]

.... 이 똑같음 .

위 첫번째 계산에서 .... 의 최소값을 찾았다면 그냥 넣기만 하면 됨 .

뒷부분을 메모이제이션 해야함

외판원 순회

함수 정의

rec(now, state)

현재까지 state 를 지나 왔고 , 현재 위치가 now 일 때 ,

최적값

외판원 순회

```
//  다음으로 이동할 곳 선택하기
rec(4, 1111b);
dp[now][state] = 1e9;
for(int i = 0; i < n; i++) {
    // state  체크 .  내가 지나온 곳에 포함되지 않는지
    //  이동할 수 있는지 체크
     if(state & (1 << i) || w[now][i] == 0) continue;
    // go(i, state|(1<<i))  를 계산한 적이 있다면 바로 리턴할거임
     dp[now][state] = min(ret, rec(i, state | (1<<i)) + w[now][i]);
}
```

```
ret = min(ret, rec(5, 11111b) + w[4][5]);
ret = min(ret, rec(6, 101111b) + w[4][6]);
..
```

외판원 순회

rec(4, 1111b) 를 들어오는 경우는 여러번 있다 .

rec(3, 111b)
[1, 2, 3] 에서 다음으로 4 를 고르는 경우

rec(2, 111b)
[1, 3, 2] 에서 다음으로 4 를 고르는 경우

외판원 순회

rec(3, 111b) 에서

-> ret = min(ret, rec(4, 1111b) + w[3][4]);

-> ret = min(ret, rec(5, 10111b) + w[3][5]);

-> ret = min(ret, rec(4, 1111b) + w[2][4]);

[1, 2, 3] + w[3][4] + [4 ,,,,,,] <= rec(4, 1111b)

[1, 2, 3] + w[3][5] + [5 ,,,,,,] <= rec(5, 10111b)

[1, 3, 2] + w[2][4] + [4,,,,,,] <= rec(4, 1111b) 히트

rec(4, 1111b) 는 한 번만 구하면 다음부터는 캐시를 사용함

-> [4 ,,,,,,,,,] 의 최적값을 기억하고 있기 때문에

이전 상태랑 연결만 해주면 된다 따라서 w[i][j] 를 더하는것임

```
cout << rec(1, 1b);


if(state ==  가득차면 ) {
                w[now][0] != 0 {
                                return w[now][0];
                } else {}


for(int i = 1; i <= n; i++) {
    if(state & (1 << i) || w[now][i] == 0) continue;
    ret = min(ret, rec(i, state | (1<<i)) + w[now][i]);
   ret = min(ret, 1e9+ w[now][i]);
}
ret = 0;
ret = min(ret, rec(2, 11b) + w[1][2]);
[1] + w[1][2] +[2.....] +
..
[1] + w[1][2] + [2] + w[2][4] + [4] + .....  + rec(n, 1111111);



ret = min(ret, rec(3,101b) + w[1][3]);
```