P7

21700477 윤다은

문제정의

주어진 숫자들을 이웃하는 두 원소끼리 이어나가며 전체를 이었을 때의 최소비용을 구하는 문제

문제해결

<풀이1>

첫 파이프부터 자신의 오른쪽 파이프의 길이를 합한 값을 구한 뒤 그 중에서 가장 작은 값을 합해 나가는 방식 -> 오답

<풀이2>

한 파이프로부터 두개로 쪼개져 나누어 간다는 아이디어를 활용하였다. 쪼개진 두 파이프의 편차가 가장 적게 나는 경우가 정답일 것이라고 생각하여, 쪼개었을 때 편차가 가장 적게 나는 인덱스를 선택하였다. -> 오답

<풀이3>

파이프를 1차원 배열(pipes)에 순서대로 입력받는다. 그리고 각 구간의 최소 합병 비용을 저장할 2차원 배열 cost와 각 구간의 최소 비용이 계산되었는지 저장하는 v를 선언한다.

구간의 양끝의 인덱스를 각각 l과 r이라고 할 때 cost[l][r]은 다음과 같이 정의할 수 있다. S는 해당 구간의 단순 합을 나타낸다.

```
cost[l][r] = min(S[l][r] + cost[l][l] + cost[l+1][r],
                 S[l][r] + cost[l][l+1] + cost[l+2][r],
                 …
                 S[l][r] + cost[l][r-1] + cost[r][r])
```

예를 들어 0부터 3번째 까지의 최소 합병 비용은 아래와 같이 계산된다.

```
cost[l][r] = min(S[0][3] +  cost[0][0] + cost[1][3],
                 S[0][3] + cost[0][1] + cost[2][3],
                 S[0][3] + cost[0][2] + cost[3][3])
```

이 계산 과정을 프로그램으로 작성하기 위해 함수f를 정의한다. 함수의 매개변수로는 합하고자 하는 구간의 왼쪽 인덱스 i와 오른쪽 인덱스 j 그리고 파이프를 저장한 배열이 있다.

0부터 n-1까지의 구간의 합을 구하므로 초기 i와 j의 값은 0과 n-1이다. 이 함수는 재귀 형태로 반복되며, 함수에서는 **두가지 종료 케이스**를 가진다.

1) i와 j가 같은 경우 합칠 파이프가 없다는 의미이므로 0을 반환한다.
2) v[i][j]가 true인 경우 이미 계산된 값이 있다는 의미이므로 cost[i][j]를 반환한다.

두 가지 종료 케이스에 전부 속하지 않은 경우 **cost[i][j]를 계산**한다. 계산하는 과정은 다음과 같다.

1) i와 j값을 계산하기 때문에 v[i][j]의 값을 true로 변경한다.
2) S(i,j)를 계산하기 위해 i부터 j까지의 값을 더해 total에 저장한다
3) 초기 res는 10^9로 지정하고 k를 i+1부터 j까지 반복하며 total+cost[i][k] + cost[k][r]과 res를 비교하여 가장 작은 값으로 res를 업데이트한다.
4) cost[i][k]와 cost[k][r]은 i와 j를 (i,k), (k,r)로 f함수를 재호출하여 구한다.

시간복잡도

cost[i][j]는 일반적으로 N^2/2만큼 호출되며, 0부터 n-1까지 구간합의 경우의 수를 계산하는 데에는 N-1번 반복되므로 N^2/2 * (N-1) 따라서, O(N^3)이다.

공간복잡도

v, cost, pipes 모두 최대 n인 500으로 선언하였으므로, 공간복잡도는 O(1)이다.