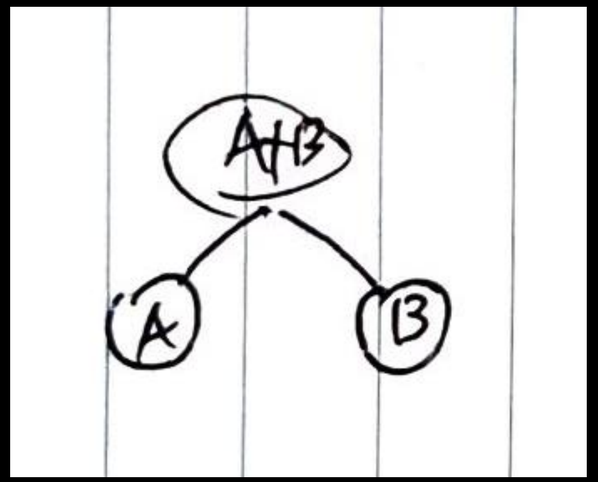
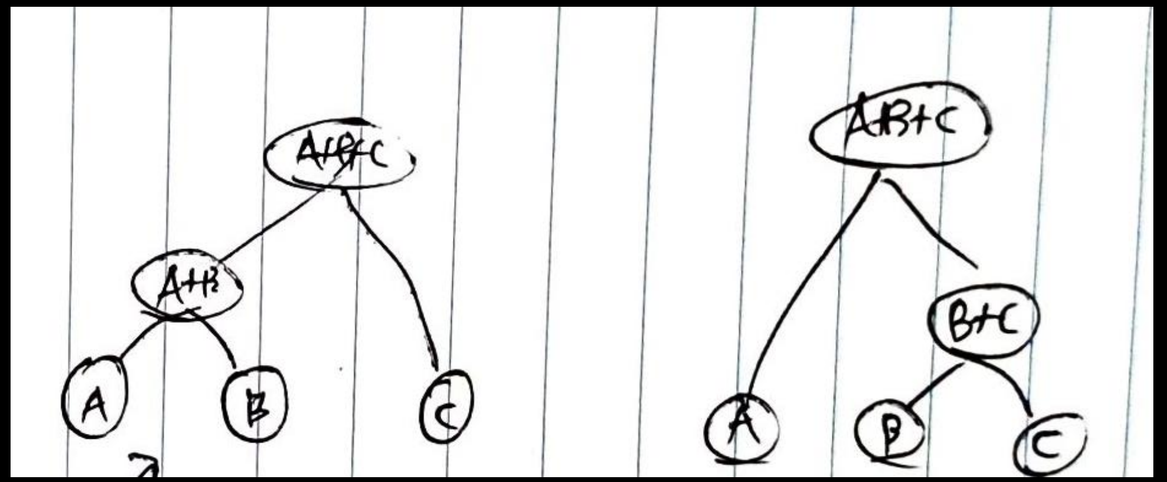
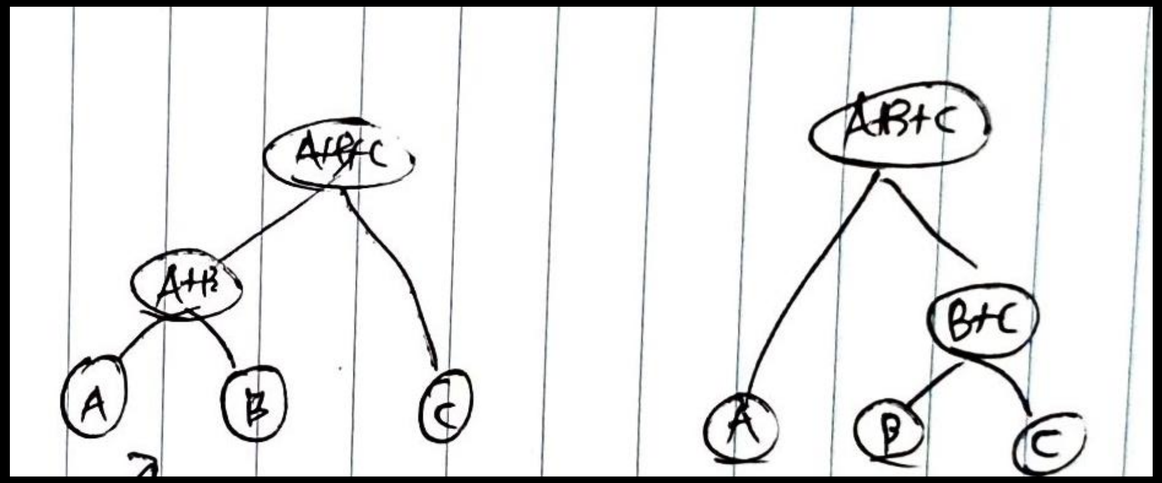
1. 중복되는 부분 문제 찾기

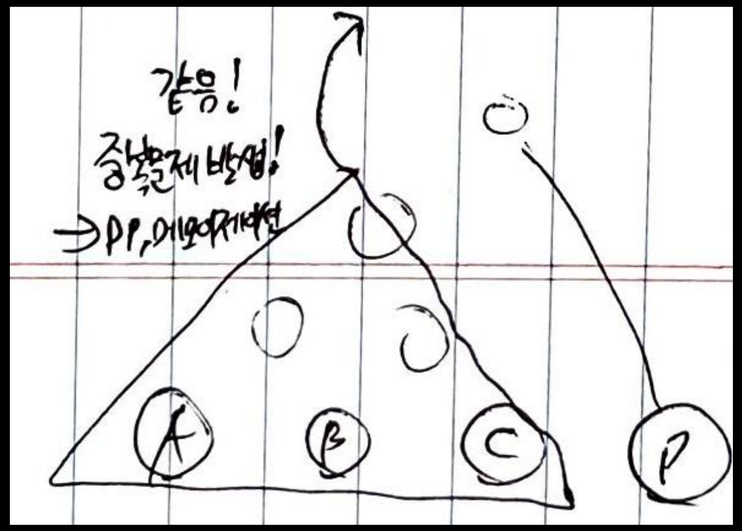




전체 파일이 2개, 3개 일 때 파일을 합치는 경우의 수 는 이미 정해져 있다

1. 중복되는 부분 문제 찾기

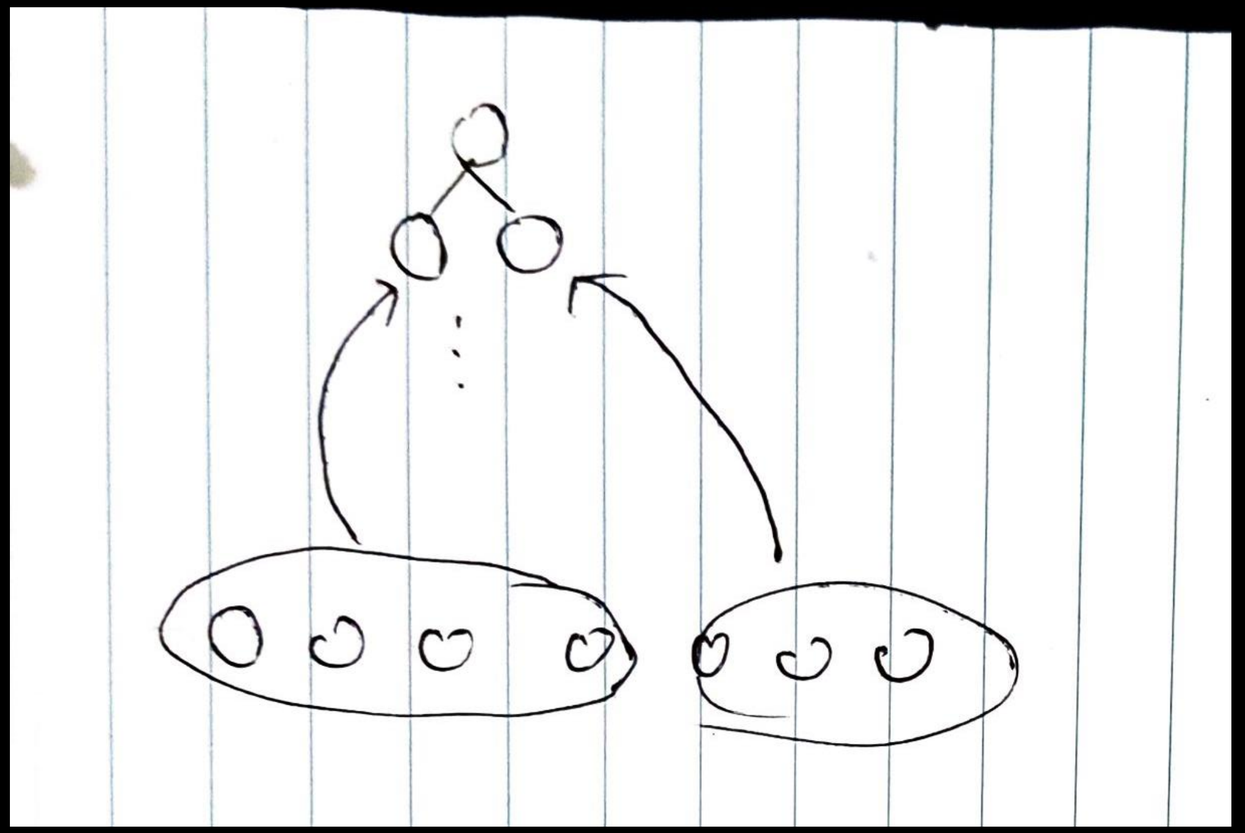




전체 파일이 4개 일 때, 파일 A, B, C 를 합치는 방법은 3개의 파일을 합치는 경우의 수와 같다

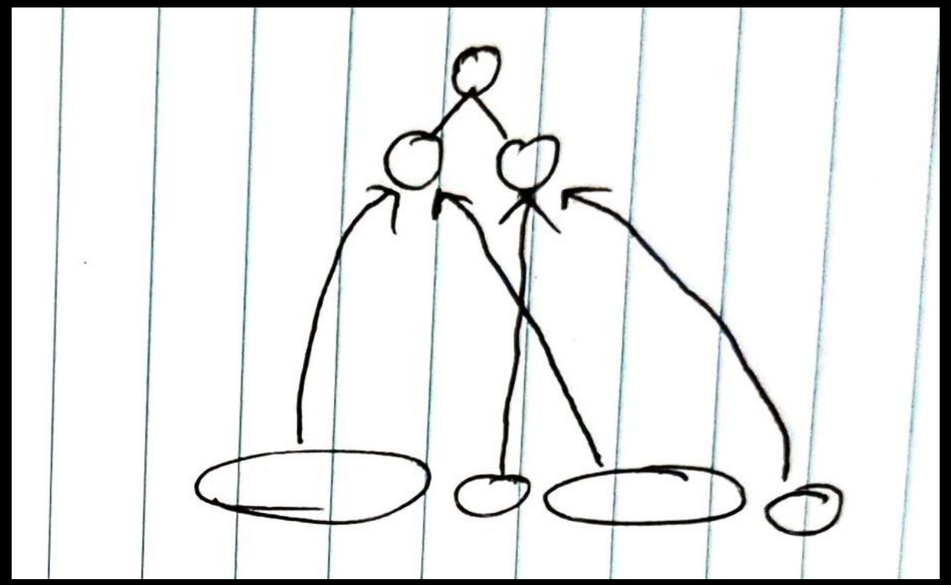
-> 파일을 합치는 중복되는 부분문제가 존재한다

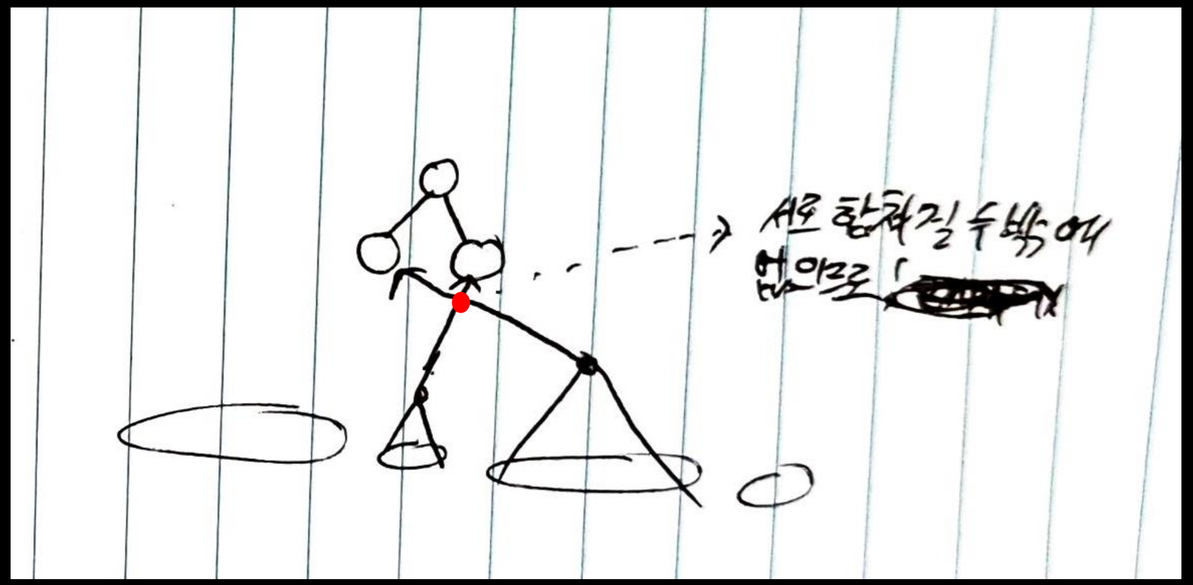
2. 증명: 파일을 합치는 경우는 결국 전체 파일을 2부분으로 나눈 후 각 부분을 합치는 것 과 같다



전체 파일을 합치는 것은 결국 전체 파일을 2부분으로 나눈 후 각 부분을 합쳐가는 것과 같다는 것을 증명해 보자.

2. 귀류법: 2부분이 아닌 여러 부분으로 나뉠 수 있다고 가정



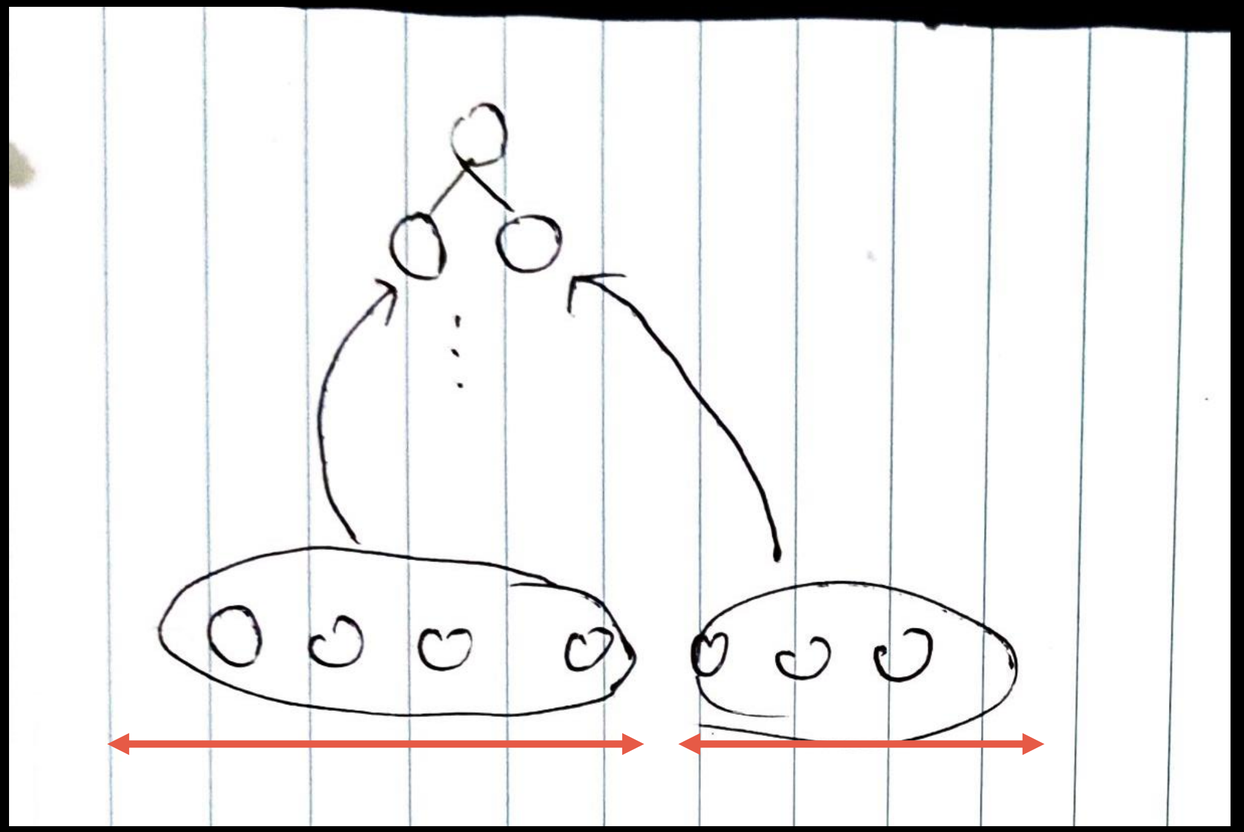


2부분이 아닌 여러 부분으로 나뉘어 진 경우

이때, 나누어진 각 부분이 합쳐지는 시점이 무조건적으로 존재하며

이때 왼쪽 부분이 오른쪽으로, 오른쪽 부분이 왼쪽으로 진행하는 경우 서로 합쳐질 수 밖에 없으므로, 2부분 이상으로 나뉘어 질 수 없다.

3. 메모이제이션 적용



전체 파일은 항상 2부분으로 나뉘어지며 이때 좌측 부분과 우측 부분의 [시작점, 끝점] 좌표로 메모이제이션이 가능하다

4. 구현

```
const int INF = 987654321;
int N;
int cache[501][501];
int cost[501];
int preSum[501];

// a, b: 현재 구간의 [시작점, 끝점] 좌표
int dp(int a, int b) {
    if (a == b)
        return cost[a];


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

    ret = INF;

    // 양쪽 파일을 모두 합쳤을 때의 값을 더해줘야 한다
    int sum = preSum[b+1] - preSum[a];

    // 현재 파일들을 2 부분으로 나눈다
    for (int i = a; i < b; i++)
        // 양쪽 파일을 합친 결과 값
        ret = min(ret, dp(a, i) + dp(i + 1, b) + sum);


    return ret;
}

void solve(int num) {
    int ret = INF;

    for (int i = 0; i < num - 1; i++) {
        ret = min(ret, dp(0, i) + dp(i + 1, num - 1));
    }

    cout << ret << endl;
}
```

코드 원본: https://gist.github.com/injae-kim/2b419af2a077ae6b24007a8458850df7