SAMSUNG SW AI ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

삼성청년 SW·AI아카데미

SWEA2115 벌꿀채취

SAMSUNG SW AI ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

벌꿀채취

목표 : 두 일꾼이 꿀을 채취하여 얻을 수 있는 수익의 합이 최대가 되는 경우를 찾고, 그 때의 최대 수익을 출력하는 프로그램을 작성

조건

1. **벌통은 NxN 정사각형 격자**
2. **일꾼은 가로로 연속된 M개의 벌통만 선택 가능**
3. **선택한 M개의 벌통에서 꿀을 일부 벌통만 골라서 채취 가능**
   - 채취한 꿀의 총량은 C 이하
   - 벌통의 꿀 중 일부분만 채취 불가
4. **수익은 채취한 각각의 벌통 꿀의 양의 제곱 합으로 계산**
   - 6,1 → 6*6 + 1*1 = 37
5. **두 일꾼이 선택한 M개의 벌통은 겹쳐선 안됨**
6. **각 일꾼은 자신이 선택한 M개의 벌통 중에서 꿀을 채취할 조합을 따로 결정함**

제약 사항

1. 시간제한 : 3초.
2. 벌통들의 크기 N (3 ≤ N ≤ 10)
3. 선택할 수 있는 벌통의 개수 M (1 ≤ M ≤ 5)
4. 반드시 M ≤ N 으로 주어진다.
5. 꿀을 채취할 수 있는 최대 양 C (10 ≤ C ≤ 30)
6. 벌통 하나의 꿀의 양 (1 ≤ 입력 data ≤ 9)
7. 벌통에 있는 모든 꿀은 한번에 채취해야 한다.

벌꿀 채취 방법 이해하기 1

벌통크기 N : 4, 벌통개수 M : 2, 꿀의 양 C : 13

- **벌통 정보가 다음과 같을 때, 두 일꾼이 꿀을 채취한다.**

6	1	9	7
9	8	5	8
3	4	5	3
8	2	6	7

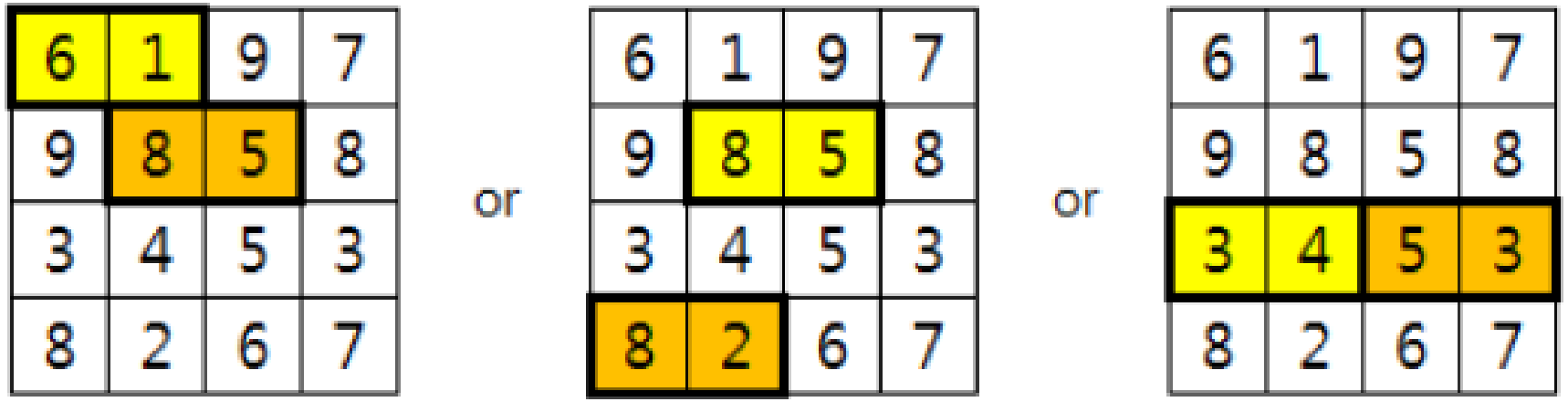
[Fig. 1]

벌꿀 채취 방법 이해하기 2

조건

- 두 명의 일꾼이 있다.
- 꿀을 채취할 수 있는 벌통의 수 M이 주어질 때, 각각의 일꾼은 가로로 연속되도록 M개의 벌통을 선택하고, 선택한 벌통에서 꿀을 채취할 수 있다.
- 단, 두 명의 일꾼이 선택한 벌통은 서로 겹치면 안 된다.

M이 2인 경우,



[Fig. 2]

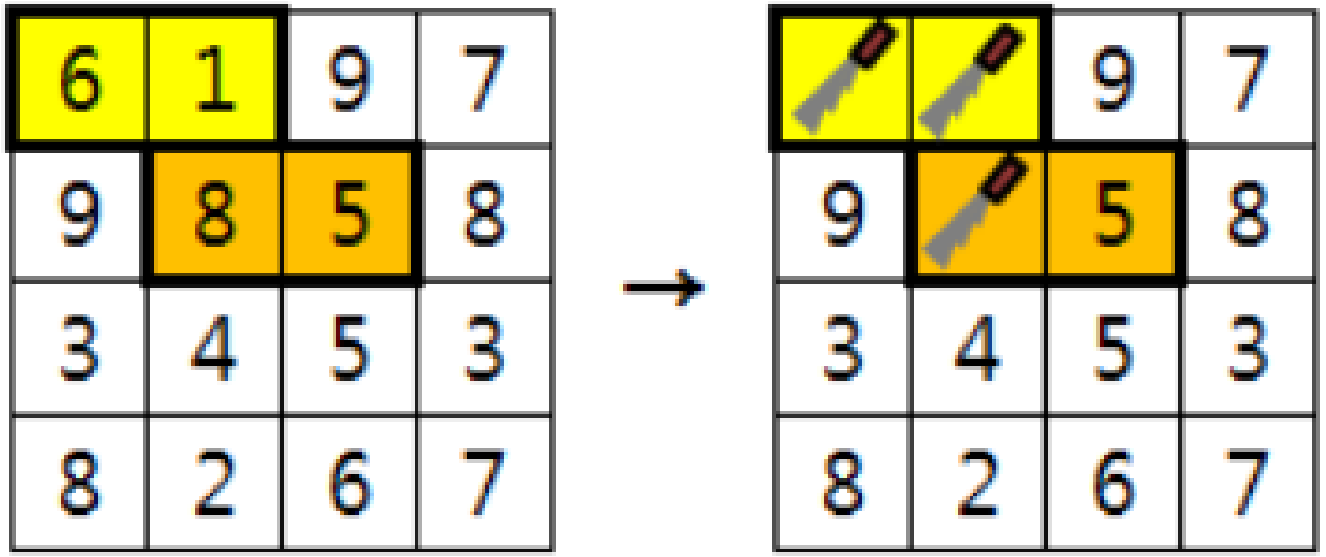
벌꿀 채취 방법 이해하기 3

조건

- 두 명의 일꾼은 선택한 벌통에서 꿀을 채취하여 용기에 담아야 한다.
- 단, 하나의 벌통에서 채취한 꿀은 하나의 용기에 담아야 한다.
- 하나의 벌통에서 꿀을 채취할 때, 일부분만 채취할 수 없고 벌통에 있는 모든 꿀을 한번에 채취해야 한다.
- 두 일꾼이 채취할 수 있는 꿀의 최대 양은 C 이다.

C가 10일 때

- 수익 계산 방법
- 각 통의 제곱의 합
  - 6*6 + 1*1 + 8*8 = 101



[Fig. 3]

데이터 입력 받기

N : 벌통 크기

M : 선택할 수 있는 벌통 개수

C : 채취할 꿀의 최대 양

map[] : 벌통 정보

```
//input
cin >> N >> M >> C;
for (int y = 0; y < N; y++) {
        for (int x = 0; x < N; x++) {
                cin >> map[y][x];
        }
}
```

최적화를 위한 빌드업

각각의 벌통에서 시작하는

M 개의 벌통 조합(!!!) 에서 최대 이익을 계산하는 문제

- 입력 받은 벌통 정보를 제곱해서 보관하는 것은 최적화에 유리하다.
- 조합이기 때문에 매우 많은 연산이 필요하다.
- 코드의 가독성을 위해서라도 미리 계산해 두는 것이 유리

```
for (int y = 0; y < N; y++) {
    for (int x = 0; x < N; x++) {
        cin >> map[y][x];
        square[y][x] = map[y][x] * map[y][x];
    }
}
```

1. 각각의 벌통(y,x) 에서 연속된 M개의 벌통을 선택하는 모든 경우의 수 탐색해서 얻을 수 있는 이익을 계산한다.

- **부분집합을 위한 DFS를 사용해서 시작 좌표(y,x) 에서 M 개의 벌통 선택 시 얻을 수 있는 이익 계산해서 기록**
  - cost[y][x] : y,x 에서 얻을 수 있는 이익 기록용 배열
- **연속된 벌통을 선택하거나 선택을 하지 않거나**
  - 채취할 수 있는 벌꿀의 양 C 초과 시
  - 벌통의 크기 N 초과 시
- **종료 시점 : M 칸 모두 검사 완료 시**

M = 3, C = 13 일 경우,

- 6 (O)
- 1 (O)
- 9 (O)
- 6 + 1 = 7 (O)
- 1 + 9 = 10 (O)
- 6 + 1 + 9 = 16 (X)

최대 이익이므로 10!

6	1	9	7
9	8	5	8
3	4	5	3
8	2	6	7

각각의 벌통(y,x) 에서

→ sy,sx 좌표 필요

연속된 M개의 벌통을 선택하는 모든 경우의 수 탐색해서 얻을 수 있는 이익을 계산

→ 해당 좌표에서 각각의 선택지(now) 마다 얻을 수 있는 이익들 중에서 최대 이익을 비교해야 함

→ 매 분기마다 누적된 합(sum) 과 매 분기가 끝날 때 비교할 최대 이익(profit)을 기록해야 함

```
void dfs(int sy, int sx, int now, int sum, int profit)
```

각 좌표(y,x) 에서 M개의 선택 가능한 모든 조합 중 최대 이익을 계산해서 cost에 기록

```
for (int y = 0; y < N; y++) {
    for (int x = 0; x < N; x++) {
        dfs(y, x, x, 0, 0);
    }
}
```

```
//y,x 에서 M개의 벌통 중 선택 가능한 이익 기록용 배열
int cost[12][12];

void dfs(int sy, int sx, int now, int sum, int profit) {

    //종료 조건

    //0 or 1로 현재 칸의 꿀을 선택 할 지 말지 결정 (분기)
    for (int pick = 0; pick < 2; pick++) {
        int value = square[sy][sx] * pick;

        //현재까지 채취한 꿀 + 현재 칸의 꿀이 최대 채취량 C를 넘기면?

        //다음 칸으로 이동할 때, 벌통의 크기 N을 벗어나면?

        //다음 칸으로 이동하자.

    }
}
```

2. 각 좌표(y,x) 에서 첫 번째 일꾼이 채취했을 때, 그 이후 칸들 중에서 두번째 일꾼이 선택할 수 있는 최대 이익을 계산해야 한다.

- **두 번째 일꾼이 이 후 선택을 또 다시 dfs로 매 번 계산하면 속도가 느려진다.**

→ 미리 해당 좌표 이후 선택 가능한 최대 COST를 기록해야 한다! (배열 사용)

→ 이 후 위치에서 최대 COST를 기록해야 하므로 역순으로 탐색

각 좌표(y,x)에서 채취 시,

이 후, 시작 가능한 모든 구간 중 최대 이익 기록

```
for (int y = 0; y < N; y++) {
    for (int x = 0; x < N; x++) {
        dfs(y, x, x, 0, 0);
    }
}
calc();
```

```
int Mcost[12][12];
void calc() {
    int last = 0;
    for (int y = N - 1; y >= 0; y--){
        for (int x = N - 1; x >= 0; x--) {
            last = max(last, cost[y][x]);
            Mcost[y][x] = last;
        }
    }
}
```

3. 이제 두 일꾼이 채취할 구간의 조합이 겹치지 않게 선택한다.

- **첫번째 일꾼이 (y,x) 에서 작업 시, 두 번째 일꾼의 선택은 (y, x+M) 이거나 N을 벗어나서 (y+1, 0) 이 된다.**

ny,nx 의 값과 범위에 주의한다.

```
void solve() {
    //각 좌표에서 일꾼의 선택 구하기
    for (int y = 0; y < N; y++) {
        for (int x = 0; x < N; x++) {

            //두 번째 일꾼 시작 좌표
            int ny;
            int nx;

            //같은 행 범위 벗어날 경우 다음 열

            //벌통의 크기 벗어날 경우 종료

            result = max(result, cost[y][x] + Mcost[ny][nx]);
        }
    }
}
```

SAMSUNG SW AI ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

내일 방송에서 만나요!

삼성청년SW·AI아카데미