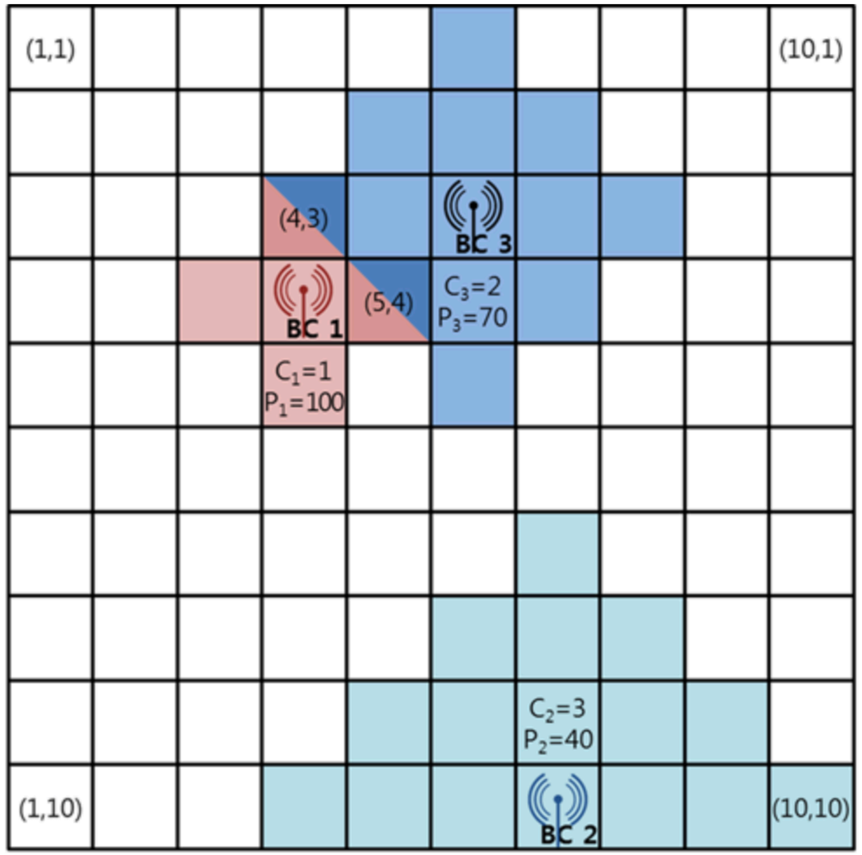
무선 충전

SWEA 5644

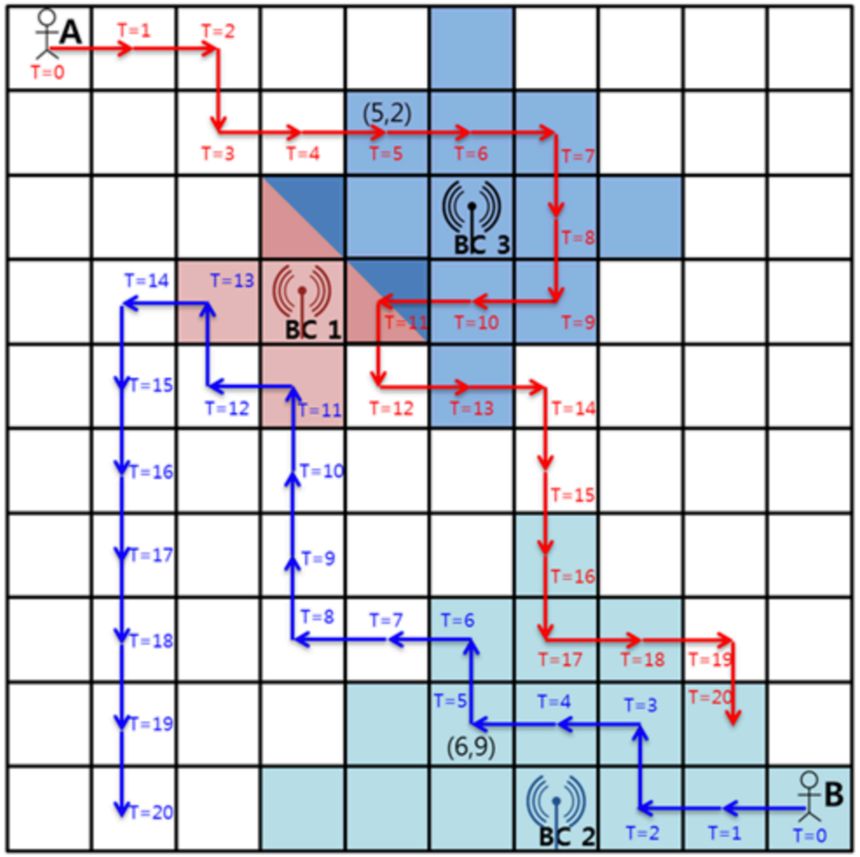
전윤철

문제 요약



- 무선 충전기 BC의 좌표, 범위, 성능 제공
- 사용자 A와 사용자 B가 각각 (1, 1), (10, 10)에서 출발
- 각 사용자의 매 초 이동 방향 제공
- 매 순간 연결된 BC 성능 합의 최대치 계산

문제 요약



- 사용자는 범위 내 연결할 BC 선택 가능
- 같은 BC를 사용자가 공유한다면 충전 양 균등 분배
- 사용자의 초기 위치(0초)부터 충전 가능
- BC는 서로 같은 위치에 설치 불가

제약 사항

- 지도의 크기 10 x 10
- 20 ≤ 총 이동 시간 M ≤ 100
- 1 ≤ BC의 개수 A ≤ 8
- 1 ≤ BC의 충전 범위 C ≤ 4
- 10 ≤ BC의 성능 P ≤ 500. P는 짝수
- 임의 좌표 X, BC Y에 대하여

  거리 D가 DB의 충전 범위 C 이하이면

  좌표 X는 BC Y에 접속할 수 있다
- $D = |X_A - X_B| + |Y_A - Y_B|$

접근 방법

- 매 순간의 선택이 미래에 영향을 주지 않음

  => 그리디

- 매 초 각 사용자가 최대 성능의 BC를 선택

접근 방법

- 모든 좌표에 대해 연결된 BC를 성능순으로 매핑
- 각 사용자가 이동하며 먼저 연결된 BC를 선택
- 해당 BC의 성능을 최종 결과에 합산

코드 - charge (1)

```
57      // 각 플레이어가 움직이며 충전 시작
58      static void charge() {
59          setChargingFeild();
60          int ia = 0, ja = 0, ib = 9, jb = 9;
61          boolean connectA, connectB;
62          for (int s = 0; s <= M; s++) {
63              connectA = map[ia][ja] != null;
64              connectB = map[ib][jb] != null;
65              // 한 사람만 충전기와 연결되어 있다면 해당 충전기의 성능 합산
66              if (connectA && !connectB) {
67                  sum += map[ia][ja].get(0)[1];
68              } else if (!connectA && connectB) {
69                  sum += map[ib][jb].get(0)[1];
70              }
```

코드 - charge (2)

```
71              // 두 사람이 모두 충전기와 연결되어 있다면
72              else if (connectA && connectB) {
73                  // 플레이어 A의 최대 충전기 성능 합산
74                  sum += map[ia][ja].get(0)[1];
75                  // 두 사람이 같은 충전기에 연결 중이라면
76                  if (map[ia][ja].get(0)[0] == map[ib][jb].get(0)[0]) {
77                      // 두 사람 모두 복수의 충전기에 연결중이라면
78                      if (map[ib][jb].size() > 1 && map[ia][ja].size() > 1) {
79                          // 각 플레이어의 2순위 성능 충전기 중 큰 값을 합산
80                          sum += Math.max(map[ia][ja].get(1)[1], map[ib][jb].get(1)[1]);
81                      }
82                      // 한 사람만 복수의 충전기에 연결중이라면 해당 플레이어의 2순위 성능 충전기 합산
83                      else if (map[ia][ja].size() > 1) {
84                          sum += map[ia][ja].get(1)[1];
85                      } else if (map[ib][jb].size() > 1) {
86                          sum += map[ib][jb].get(1)[1];
87                      }
88                  }
89                  // 플레이어 B의 최대 성능 충전기 합산
90                  else {
91                      sum += map[ib][jb].get(0)[1];
92                  }
93              }
```

코드 - charge (3)

```
94              // 최대 시간에 도달했다면 중지
95              if (s==M) {
96                  break;
97              }
98              // 각 플레이어를 다음 위치로 이동
99              ia += delta[playerA[s]][0];
100             ja += delta[playerA[s]][1];
101             ib += delta[playerB[s]][0];
102             jb += delta[playerB[s]][1];
103         }
104     }
```

코드 - setChargingFeild (1)

```java
    // 충전 가능 구역 설정
    static void setChargingFeild() {
        // 무선 충전기를 성능이 좋은 순으로 내림차순 정렬
        Arrays.sort(belkins, new Comparator<int[]>() {

            @Override
            public int compare(int[] o1, int[] o2) {
                return o2[3] - o1[3];
            }
        });
        for (int i = 0; i < belkins.length; i++) {
            int ai = belkins[i][0], aj = belkins[i][1], ar = belkins[i][2];
            for (int j = ai - ar; j <= ai + ar; j++) {
                for (int k = aj - ar; k <= aj + ar; k++) {
                    // 검사 좌표가 배열의 범위 내에 있고 충전기의 범위  내라면
                    if (isIn(j, k, ai, aj, ar)) {
                        // 현 좌표에 충전기 정보가 없을 경우 생성
                        if (map[j][k] == null) {
                            map[j][k] = new ArrayList<>();
                        }
                        // 현재 충전기의 번호와 성능을 좌표에 저장
                        Integer[] temp = { i, belkins[i][3] };
                        map[j][k].add(temp);
                    }
                }
            }
        }
    }
```

코드 - setChargingFeild (2)

```
135     // 충전기의 범위 내인지 확인
136     static boolean isIn(int ni, int nj, int i, int j, int range) {
137         if (ni < 0 || ni >= size || nj < 0 || nj >= size ||
138             (Math.abs(i - ni) + Math.abs(j - nj)) > range) {
139             return false;
140         }
141         return true;
142     }
143 }
```

END