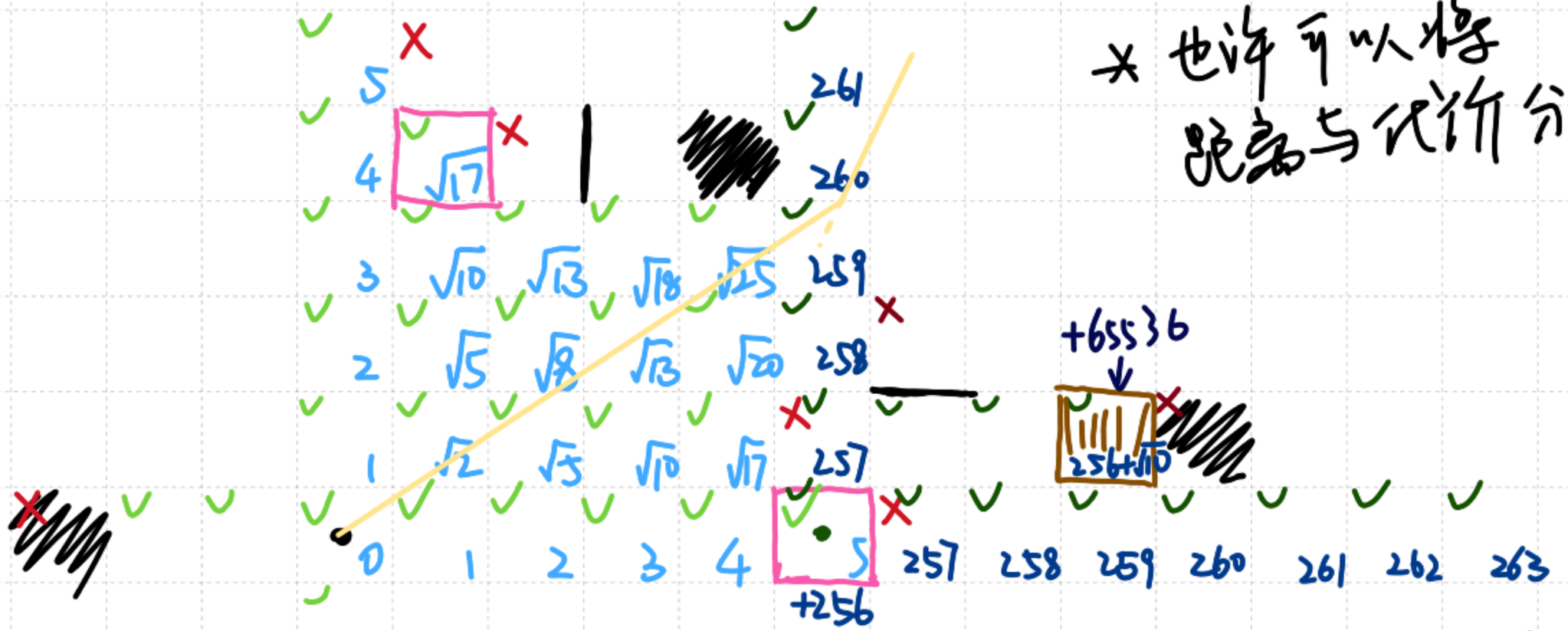



* 也许可以将跳高与代价分离.

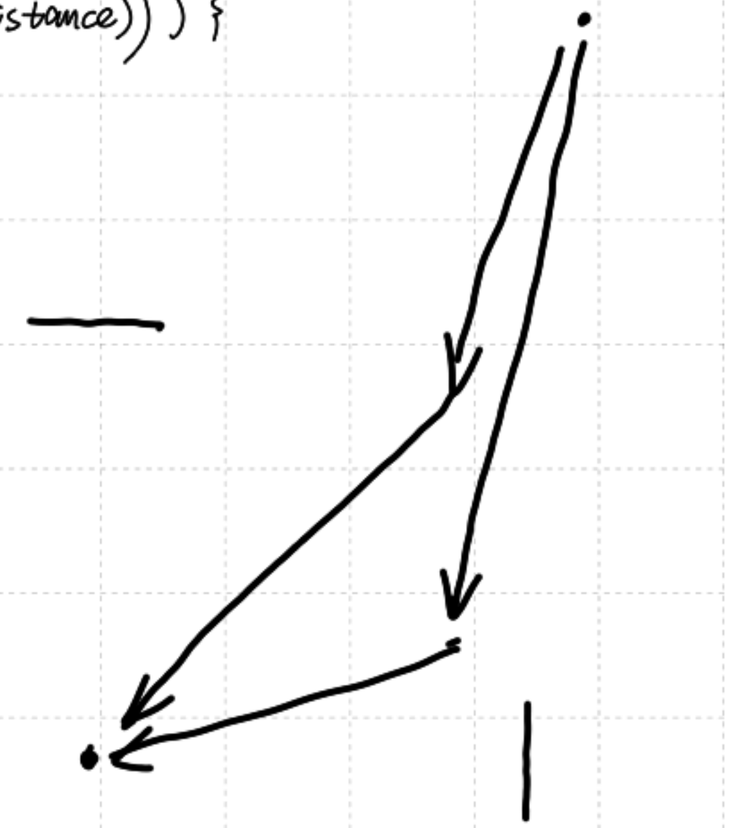
Distance : $d * 2^8$ → uint32_t.

| 0 | 256 | 512 | 768 | ... |
|---|---|---|---|---|
| 1 | 362 | 572 | 810 | ... |
| ⋮ | ⋮ | ⋮ | ⋮ | |

FillQuadrantTile {

```
Tile LT, DT. MT;
TileSearch LS, DS. MS;
bool accessible =
    LS.accessible && DS.accessible && !MT.isWall[Left] && !MT.isWall[Down]&&
    LT.blockLv==0 && DT.blockLv==0 && (MT.obstacleLv==0)
if (!accessible) return false;
uint32_t distance = source.distance + Distance[dx][dy];
ushort cost = source.cost + MT.blockedLv;
if (!MS.isAccessible || (cost<MS.cost) || ((cost==MS.cost)&&(distance<MS.distance))) {
    MS.source = source;
    MS.cost = cost;
    MS.distance = distance;
    MS.isAccessible = true;
    Queue.PushBack…
}
return true;
```

}



```
struct TileSearch{
    bool isAccessible;
    short sourceX;
    short sourceY;
    ushort cost;
    uint distance;
}
enum Direct {
    right = 0,
    up = 1,
    left = 2,
    down = 3,
}
```

```
struct Tile {
    bool isWall[4];
    short obstacleLv;
    short blockedLv;
}
```

只要非0就生效。

↑

← 障碍。(永不可通行)

← 挡路线(极端下可通行)

Lv1. 一般挡路线

Lv2.

⋮

Lv256. 地穴(等)

(作用于cost)。

```
void SpreadFrom(short X, short Y) {
    // →, ↑, ←, ↓ ;  →   maxR / U / L / D;
    SpreadQuadrant(X,Y, ..... ); (→, ↑, ←, ↓),
    ...
}
```

```
void SpreadQuadrant(short SourceX/Y, short MaxX/Y, short Direct){
    short  dx, dy = ... ; // By Direction.
    short  x = SourceX + dx, y;
    while (x <= maxX) {
        y = SourceY + dy;
        while(y<= maxY) {
          y+=dy;
        }
        x+= dx;
    }
```

Fill Quadrant Tile.

```
void Search() {
    ~~Queue.PushBack(Start);~~ SpreadFrom(Start);
    while(!Queue.empty()){
        auto pt = Queue.front();
        Queue.PopFront();
        SpreadFrom(pt);
    }.
}
```