1 Algorytm

Algorytm jest prawie identyczny do algorytmu przedstawionego na wykładzie tylko dla każdej kolumny mamy dodatkowo dwie pętle, które sprawdzają ruchy w góre i w dół, a przy odzyskiwaniu wyniku mamy dodatkowe dwa przypadik ruchów.

```
listapunktow trasa(int i, int j){
    int k, l = j-1;
    if(j == 1)
        return i;
    if(d[i-1][j-1] < d[i][j-1])
        k = i-1;
    else
        k = i;
    if(d[k][j-1] > d[i+1][j-1])
        k = i+1;
    if(d[k][l] > d[i+1][j]){
        l = j;
        k = i+1;
    }
    if(d[k][l] > d[i-1][j]){
        l = j;
        k = i+1;
    }
    return concat(trasa(k, l), i, j);
}
```

Rysunek 1: echo reply

```
for(j = 1; j <= m; j++)
    d[0][j] = d[n+1][j] = INFINITY;
for(i = 1; i <= n; i++)
    d[i][1] = a[i][1];
for(j = 2; j <= m; j++)
    for(i = 1; i <= n; i++)
        d[i][j] = a[i][j] + min(d[i-1][j-1], d[i][j-1], d[i-_][j-1]); // z poprzedniej
kolumny
    for(i = 1; i <= n; i++)
        d[i][j] = min(d[i][j], a[i][j] + d[i-1][j]); // z dołu
    for(i = 1; i <= n; i++)
        d[i][j] = min(d[i][j], a[i][j] + d[i+1][j]); //z góry
```

Rysunek 2: echo reply

2 Dowód poprawności

Załóżmy nie wprost, że otrzymaliśmy nie optymalną ścieżkę więc nasz wynik jest większy niż optymalny.To oznacza, że punkt, w którym się kończy optymalna ścieżka ma u nas większy koszt dojścia, wybierzmy pierwszy taki punkt, w którym koszt dojścia do tego pola jest większy niż w ścieżce optymalnej. Skoro poprzednik ma obliczony dobry koszt oznacza, że nie rozpatrzyliśmy przypadku dojścia do tego elementu (nieoptymalnego) po obliczeniu kosztu dojścia do poprzednika. Rozważmy przypadki gdzie leży poprzedni element:

1. Jeśli poprzedni element leży w innej kolumnie (z mniejszym indeksem) na pewno rozważyliśmy taki przypadek, ponieważ po przejściu do następnej kolumny nie zmieniamy już wyniku w poprzednich kolumnach. Więc mamy sprzeczność

2. Podobnie przy elemencie z "dołu", ponieważ jest to ostatnia pętla.

3. Poprzedni element leży powyżej nas, mogliśmy zmienić jego wartość przy sprawdzaniu czy nie ma lepszej drogi przychodząc z dołu, ale z prostej obserwacji można zauważyć, że jeśli byśmy zaktualizowali element, który jest powyżej to oznacza, że nasza wartość jest mniejsza bądź równa wartości powyżej (wagi są nieujemne), więc nie jesteśmy w stanie otrzymać lepszego wyniku w tym punkcie. Otrzymujemy sprzeczność.

3 Złożoność

Obliczanie tablicy najmniejszego kosztu dojścia do pola X wynosi $\Theta(n*m)$ ponieważ dla każdej kolumny (m) wykonujemy n*3 obliczeń. Koszt odzyskiwania trasy nie ma większej złożoności niż $\Theta(n*m)$, ponieważ jak wcześniej pokazaliśmy nic nie zyskujemy przechodząc wielokrotnie jakieś pole. Więc złożoność naszego algorytmu to $\Theta(n * m)$.