1\. (a) A small example: skiers = [60 62 64]

skis = [56 62 64]

(skier, ski) pairs derived from Prof. I.M. Rong's algorithm:

(60, 62), (62, 64), (64, 56)

$$\text{disparity} = (62-60) + (64-62) + (64-56)$$
$$= 2+2+8$$
$$= 12$$

A better solution:

(skier, ski) pairs: (60, 56), (62, 62), (64, 64)

$$\text{disparity} = 60-56 = 4$$

Since $4 < 12$, it is a better solution.

(b) There are 6 cases in total:

$$S_y < S_k < k < y$$
$$S_y < k < S_k < y$$
$$S_y < k < y < S_k$$
$$k < S_y < S_k < y$$
$$k < S_y < y < S_k$$
$$k < y < S_y < S_k$$

Here we choose to analyze the fourth case:

$$k < S_y < S_k < y$$

(skier, ski) pairs from the original condition: $(k, S_k), (y, S_y)$

$$\text{disparity} = S_k - k + y - S_y$$
$$= (S_k - S_y) + (y - k)$$

(skier, ski) pairs after switching: $(x, S_y), (y, S_x)$

$$\text{disparity\_switch} = S_y - x + y - S_x$$

$$= (S_y - S_x) + (y - x)$$

$\because\ y > x \ \&\ S_x > S_y$

$\therefore\ S_y - S_x < 0 < S_x - S_y$

$\therefore\ (S_y - S_x) + (y - x) < (S_x - S_y) + (y - x)$

$\therefore$ disparity_switch < disparity

In this case, we have proved that removing the inversion does not make the solution any worse.

(c) Here the number of skiers and skis is the same, and according to the conclusion from part (b), removing the inversion does not make the solution any worse, we could simply give skis to skiers that have the same order. To be specific, we would assign the shortest pair of skis to the shortest skier, the second shortest pair of skis to the second shortest skier, repeat this assignment until assign the tallest pair of skis to the tallest skier.

(skier, ski) pairs:

(skier[0], ski[0]), (skier[1], ski[1]), ......, ~~(skier[n], ski[n])~~

(skier[n-1], ski[n-1])

To finish assignment, we need to go through two arrays of length n in linear time simultaneously, so the running time is $O(n)$.

(d) i. recurDisparity ( skiers , skis ) :

n is the length of the array of skiers
m is the length of the array of skis
if n is equal to 0 :
return 0

else if n is equal to m :
return partC ( skiers , skis )
else :
choice 1 : recurDisparity ( skiers , skis [0 : m-1] )

choice 2 : recurDisparity ( skiers[0 : n-1] , skis [0 : m-1] )
plus the absolute value of the disparity
between skiers[n-1] and skis [m-1]

return the smaller one between choice 1 and choice 2

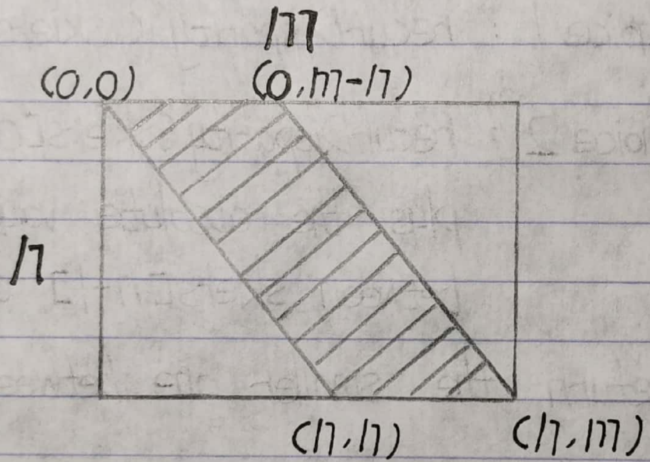
ii. dpDisparity ( skiers , skis ) :

n is the length of the array of skiers
m is the length of the array of skis
construct the dp table with n rows and m columns
for every element j in range (0, m+1) :
fill dp [0][j] = 0
for every element i in range (1, n+1) :
fill dp [i][i] = partC ( skiers [0 : i] , skis[0 : i] )
for every element i in range (1, n+1) :
for every element j in range ( i+1, m-n+i+1) :
choice 1 : dp [i][j-1]
choice 2 : dp [i-1][j-1] + the absolute value
of the disparity between skiers[i-1]
and skis [j-1]

dp[i][j] = the smaller value between choice1 and choice2

return dp[n][m]

iii. We filled the dp table from top left to bottom right, and we only filled a parallelogram inside the dp table since all possible choices locate in it:




Besides, we also filled the first row to be zero.
So the running time : $(m-n) \times n + m$

$$= mn - n^2 + m$$

$$\in O(n \cdot m)$$

iv. We first have an empty solution array
Then we start at dp[n][m] and go until arrive at dp[0][0] in the dp table :

```
n is the length of the array of skiers
m is the length of the array of skis
while n is not equal to 0 and m is not equal to 0 :
    if n is equal to 0 :
        m = m - 1
    else if n is equal to m :
        append m into our solution array
        n = n - 1
        m = m - 1
    else :
        if dp[n][m] = dp[n][m-1] :
            m = m-1
        else :
            append m into our solution array
            n = n-1
            m = m-1
```

after arriving at dp[0][0], reverse our solution array and it should have the same length as the array of skiers. Then we could assign skis in order, give the first pair of skis to the first skier, the second pair of skis to the second skier, repeat this process until give the last pair of skis to the last skier, and it is our optimal solution.