AtCoder ABC194 レポート

AtCoder 楽しい。。。

A - I Scream

省略

B - Job Assignment

問題文

あなたの会社には従業員 1 から従業員 N までの N 人の従業員がいます。
今あなたは仕事 A と仕事 B の 2 つの仕事を受注したので、これらを完了しなければなりません。
従業員 i は仕事 A を A_i 分、仕事 B を B_i 分でこなすことができます。

あなたは仕事 A と仕事 B にそれぞれ従業員を 1 人割り当てます。
同じ従業員を両方の仕事に割り当てても構いませんが、その場合 2 つの仕事が終わるのにかかる時間は、それぞれの仕事が終わるのにかかる時間の和となります。
仕事 A と仕事 B に異なる従業員を割り当てた場合、2 つの仕事が終わるのにかかる時間は、各仕事が終わるのにかかる時間の長い方となります。
2 つの仕事が終わるのにかかる時間として考えられる最小の値を求めてください。

制約

- $2 \leq N \leq 1000$
- $1 \leq A_i \leq 10^5$
- $1 \leq B_i \leq 10^5$
- 入力に含まれる値は全て整数

まず、出力として出さなければいけない状況を書き出していきます

手順

・A・B それぞれ一番早く終わらせられる人を見つける。
・上で選んだ二人の従業員が違う人であれば、長い方の時間を出力
・選んだ二人が違う従業員であれば、その人以外で一番早く終わらせられる人をそれぞれ A と B で調べる
・A と B で早く終わらせられる方に仕事を割り当てて、最初に選んだ人との時間を比べる
・ここで、最初に選んだ人が一人でやった方がいいのか、分担した方がいいのかを比較し、時間が短い方を選択し出力する

この考え方をコードにしていきます。

まず、仕事 A を終わらせる時間（以降時間 A とする）と仕事 B を終わらせる時間の最小値を求めるにはそれぞれ与えられた時間をリストに入れる必要があります。

今回の入力はこんな感じ。

$$
\begin{array}{ll}
N & \\
A_1 & B_1 \\
A_2 & B_2 \\
A_3 & B_3 \\
\vdots & \\
A_N & B_N
\end{array}
$$

スペース区切りの二つの数値が、N 行あります。ということは、map()での入力受け取りを N 回行えば良いということなので、コードはこうなります。

```
A_list = []
B_list = []

N = int(input())
for i in range(n):
    a,b = map(int, input().split())
    A_list.append(a)
    B_list.append(b)
```

これで、リストに時間 A と時間 B を入れることができました。

そしてそれぞれの最小値を出すには、

```
A_min = min(A_list)
B_min = min(B_list)
```

min(リスト) や、min(要素 1 , 要素 2 , … , 要素 N) で最小値を求めることができます。

これで最小値がわかりました。次に選ばれた従業員が同じ人かどうかを調べていきます。これには list.index()を使います。使い方は以下の通り。これをすると要素がリストの前から何番目(index)にあるかを調べることができます。

```
A_index = A_list.index(A_min)
B_index = B_list.index(B_min)
```

リスト.index(要素)とすることで、要素が何番目にあるかを調べることができます。
例)

```
X_list = [1,2,3,4,5]
print( X_list.(3) )
```

↓出力

2

のようになります。(リストの一番最初は 0 番目であることに気をつける)
A_min, B_min は上で求めた最小値です。これでどの従業員が選ばれたか分かりましたね。以上のコードで、時間 A と時間 B のリスト、一番仕事が早い従業員の index と時間が分かりました。

ここで場合わけを行います。

もし、違う従業員が選ばれていた場合、そのまま大きい方の時間を出力します。

```
if A_index != B_index:
    print(max(A_min, B_min))
```

max()は先ほどの min()の最大値バージョンです。

もし、同じ従業員が選ばれていた場合、最後に比較するために一人で仕事をこなした場合の時間を記憶しておきます。

```
time_alone = A_min + B_min
```

次に、選ばれた従業員と違う従業員の中から一番仕事を終わらせるのが早い人を探します。

A_list と B_list から探したいのですが、すでに最小値はわかっているのでなんとかしてその最小値だけリストから消して、もう一回 min()で最小値を行えば求める値が得られそうです。

では、リストから最小値を消してみましょう。ここでは、list.pop()を使います

A_list.pop(A_index)
B_list.pop(B_index)

リスト.pop(削除したい要素の index) でリストから削除できます。.pop()はただ単に値をリストから削除するだけではないのですが、その他の機能はここでは割愛します。リストの要素の削除の方法は色々あるので一通り調べておきましょう。

削除できたので、再び min()を使って最小値を求めます

a_min = min(A_list)
b_min = min(B_list)

どちらの値が小さいか比較して小さい方を選び、その際の出力を考えます。

time = min(max(A_min, b_min), max(B_min, a_min))

少し複雑かもしれませんがどのような処理が行われているか頑張って読みとってみてください。

これで、分担したときの時間と一人のときの時間が分かりました。最後にその２つを比較して短い方を出力しましょう。

Print(min(time, time_alone))

場合わけの作業は選んだ従業員が同じかどうかを判定するための if 文の else:の中に書いてください。これでコードは完成です。

正解コード（コピペできます）

```
n = int(input())
A_list = []
```

```
B_list = []
for i in range(n):
    a,b = map(int, input().split())
    A_list.append(a)
    B_list.append(b)

A_min = min(A_list)
B_min = min(B_list)

A_index = A_list.index(A_min)
B_index = B_list.index(B_min)

if A_index != B_index:
    print(max(A_min, B_min))
else:
    time_alone = A_min + B_min

    A_list.pop(A_index)
    B_list.pop(B_index)

    a_min = min(A_list)
    b_min = min(B_list)

    time = min( max( A_min, b_min ), max( B_min, a_min ) )

    print( min(time, time_alone) )
```

復習

・min() … 最小値を返す
・max() … 最大値を返す
・.index() … 要素のインデックスを返す
・.pop() … 要素をリストから削除する

今回紹介した考え方は Tyama 自身の考え方で、もっとスマートな考え方もあります。リストの操作のいい勉強になると思って今回このレポートを書きました。今回の B 問題は難し

かったですが、数をこなしていくうちに時間をかければ解けるようになってくるはずです。
毎週参加して実力を上げていきましょう！