1.
greedy algorithm:

sort the jobs that are received from customers in reverse order (so larger first) and we then calculate the weighted sum of completion time.

For this algorithm I am assuming that the input is going to be n number of jobs that have weight of w_x and processing time of t_x
and the output in this case will be the minimized or reduced weighted sum of completion times.

Algorithm_Job_Schedule (w_x , t_x)
{
    for x = 0 to x < n; x++
    vector1[j] ← w_x

    for x = 0 to x < n; x++
    vector2[j] ← t_x

    sort(vector1) // use any algorithm to sort the first array since that has the
                  order of weights

                  // and then we assemble the processing times in the 2nd vector
                    accordingly

    C_x = SUM of (C_x-1 * t_x) from n =1 to x -1

    //now calculate wjcj for each job

    SUM of (w_x * c_x) from x = 1 to n

}

We know C increases x also increases and the greatest weight is kept at the end and the product of largest completion time and weight will be our maximum and the weighted sum of those will not be minimized or reduced so the maximum weight is placed in the first position, so it will be multiplied by the minimum amount of completion time so largest weight is multiplied by the smallest time (or completion time)

So let's say we have a list of weights (9 9 8 7) and times (2 2 1 1)

9 2
9 2 + 2
8 1 + 2 + 2
7 1 + 1 + 2 + 2
if we add everything up we will get 136.

2.
Since swimming is something that can't happen with other events then we can't reduce that, so everyone participate in swimming one by one and total swim time will always be the same, so our total time should just include the total swimming time.
Now since we want to optimize the total time we need to optimize the total biking and running time, we know that two people can do both of those events at the same time

Greedy algorithm:
to optimize the total time the first person has to has to have a
MIN Swim
MAX Bike
MAX Run
and last person should have
MAX Swim
MIN Bike
MIN Run
so we sort according to min swim, max bike and max run (the time of them course) to the opposite which would be max swim, min bike, min run.

Optimize_Alg(S_x,B_x,R_x)
{
$S_x \leftarrow$ swim
$B_x \leftarrow$ Bike
$R_x \leftarrow$ Run

sort ($B_x + R_x$) // any sorting algorithm would work, just do it in decreasing order

i constant
j constant

if($B_i + R_i < B_j + R_j$) $\rightarrow s_i < s_j$
then // swap
swap (i,j) $\rightarrow$ invert pairs (i,j) // just a normal swap no fancy algorithm

//here is j finishes earlier than i then i stops swimming when j got out of the swimming
}

so if we have a person who swims in 30 minutes and bikes and runs in 1 hours .
and we have someone who swims in 15 minutes and bikes and runs in 2 hours.

1^{st} :	2^{nd} :
finish swim in 30 min	start swim at 30, finish swim at 45 min
bike+Run in 1	Bike+Run in 2 hrs
finish 1 hr 30 min	finish in 2 hrs 45 min