The algorithm

```
initialize R to be the set of all availabilities;
sort R by increasing finish time;
initialize ordered list A to be empty;
while (R is not yet empty):
  get i∈R; #notice i ends the earliest out of all of R
  remove i from R;
  if ∃ j∈R such that i and j overlap by some time:
    get the first j∈R; #notice j ends the earliest out of all of R
    add i and j to the end of A;
endwhile
return A as list of all partners (every two consecutive elements are partners)
```

Proof of Correctness

For purposes of comparison, let O be an optimal list of intervals and A be our solution. We need to prove $|A|=|O|$. First we will show inductively that our greedy algorithm solution A "stays ahead" of O and that it is doing better in a step-by-step fashion.

Notation:

- $i_1, \dots, i_k$ is the list of partners' availabilities in A in the order they were added to A.
- $j_1, \dots, j_m$ is the list of partners' availabilities in O.

We need to prove k=m. Assume that the availabilities in O, like A, are also ordered by increasing finish time.

Base Case

Need to prove $f(i_1) \leq f(j_1)$ and $f(i_2) \leq f(j_2)$. This is true because our greedy algorithm chooses people with the earliest possible finish times.

Inductive Case

$\forall$ r>1, we need to prove $f(i_{r+1}) \leq f(j_{r+1})$ assuming the inductive hypothesis $f(i_r) \leq f(j_r)$. We know that our greedy algorithm when attempting to choose a partner i_{r+1} for a student i_r chooses the student with the earliest-ending overlapping availability; and that when it is trying to initiate a new pair on some student i_{r+1} (after a pair – the second student of which is i_r – was just formed, or after a pairing attempt failed for i_r, someone prior), it once again chooses the student with the earliest-ending availability. We thus see that our algorithm always stays ahead of any optimal solution. $f(i_{r+1}) \leq f(j_{r+1})$.

Proof of Optimality

We will prove it by contradiction. If A is not optimal, then an optimal list O must have more pairs, that is, we must have m>k. Applying what we just proved, we get that $f(i_k) \leq f(j_k)$. Since m>k, there must be a j_{k+1} and a j_{k+2} in O. These availabilities end after j_k ends, and hence after i_k ends. So after deleting all eligible partners and everyone whose availability doesn't overlap with anyone else's ($i_1, \ldots, i_k$), the list of possible availabilities still contains j_{k+1} and j_{k+2}. But the algorithm stops with availability i_k, and it is only supposed to stop when R is empty – a contradiction.

Runtime Analysis

Iterating through each element of R is O(n). Finding the first availability that overlaps with each element is also O(n). Thus, The entire algorithm is O(n * n) = **$O(n^2)$**.