Hence, it suffices to check consecutive pairs and report all combinations of sensors until we find a pair that can't communicate with each other (at which point, we start a new block of sensors that can communicate with each other).

(c) Line 2-4, 6, and 8-12 take $O(1)$ time. Line 13 can be done in $O(n^2)$ time by copying over *result* explicitly (there are at most $O(n^2)$ pairs to report and outputting a single one takes $O(1)$ time), but line can also be done in $O(1)$ time if we pass *result* by reference instead[1]. For the set, we can simply use a linked list and add the new pair at either end in $O(1)$ time. The loop on line 5 is executed n times, as each of the cases increments *current*. The loop on line 7 is executed at most n times per iteration of the while-loop. So the running time of the algorithm is upper bounded by $n \cdot n \cdot O(1) = O(n^2)$ time.

On marking. When marking your solutions, we typically consider the following things: correctness, efficiency, clarity of your description, clarity and correctness of your correctness argument, clarity and correctness of your running time analysis.

If you can't come up with an algorithm that satisfies all requirements, *it's usually better to come up with a slower correct algorithm than a fast incorrect algorithm*. After all, when you're asked to solve a problem, we're interested in correct solutions and giving an incorrect solution very fast is less useful. Note that depending on how much slower than required your approach is, different penalties apply, depending on how easy the problem is to solve in the time you use. For example, if your data structure is supposed to return the maximum of all integers stored in it in $O(1)$ time, you'd get very few points for needing $O(n)$ time to do so, as that running time can be achieved by just iterating through all integers. You'd get more points for an $O(\log n)$ time solution, as this is more difficult to obtain and you're closer to the requirements.

It's also a good idea to avoid mixing in concepts from the lecture that aren't related to the question. Adding unrelated concepts gives the impression that you don't understand what is and isn't related and hence is likely to cost you marks. Also avoid giving multiple answers, since the least correct one shows your level of understanding and thus giving multiple answers is highly likely to cost you marks.

Finally, it's also important to be able to describe your approach and arguments in a concise fashion, so this is taken into consideration as well, though only in a very minor form. We typically consider a solution too long when it's double the length (or more) of the example solutions that we use for marking and that will be released to you afterwards. A general rule of thumb is that the example solutions will be around 2 pages long, so you should aim for at most 4 pages for your description, correctness argument, and analysis.

[1] Either option is fine and this part is not strictly required, but we include it for completeness.

Hence, it suffices to check consecutive pairs and report all combinations of sensors until we find a pair that can't communicate with each other (at which point, we start a new block of sensors that can communicate with each other).

(c) Line 2-4, 6, and 8-12 take $O(1)$ time. Line 13 can be done in $O(n^2)$ time by copying over *result* explicitly (there are at most $O(n^2)$ pairs to report and outputting a single one takes $O(1)$ time), but line can also be done in $O(1)$ time if we pass *result* by reference instead[1]. For the set, we can simply use a linked list and add the new pair at either end in $O(1)$ time. The loop on line 5 is executed n times, as each of the cases increments *current*. The loop on line 7 is executed at most n times per iteration of the while-loop. So the running time of the algorithm is upper bounded by $n \cdot n \cdot O(1) = O(n^2)$ time.

On marking. When marking your solutions, we typically consider the following things: correctness, efficiency, clarity of your description, clarity and correctness of your correctness argument, clarity and correctness of your running time analysis.

If you can't come up with an algorithm that satisfies all requirements, *it's usually better to come up with a slower correct algorithm than a fast incorrect algorithm*. After all, when you're asked to solve a problem, we're interested in correct solutions and giving an incorrect solution very fast is less useful. Note that depending on how much slower than required your approach is, different penalties apply, depending on how easy the problem is to solve in the time you use. For example, if your data structure is supposed to return the maximum of all integers stored in it in $O(1)$ time, you'd get very few points for needing $O(n)$ time to do so, as that running time can be achieved by just iterating through all integers. You'd get more points for an $O(\log n)$ time solution, as this is more difficult to obtain and you're closer to the requirements.

It's also a good idea to avoid mixing in concepts from the lecture that aren't related to the question. Adding unrelated concepts gives the impression that you don't understand what is and isn't related and hence is likely to cost you marks. Also avoid giving multiple answers, since the least correct one shows your level of understanding and thus giving multiple answers is highly likely to cost you marks.

Finally, it's also important to be able to describe your approach and arguments in a concise fashion, so this is taken into consideration as well, though only in a very minor form. We typically consider a solution too long when it's double the length (or more) of the example solutions that we use for marking and that will be released to you afterwards. A general rule of thumb is that the example solutions will be around 2 pages long, so you should aim for at most 4 pages for your description, correctness argument, and analysis.

[1] Either option is fine and this part is not strictly required, but we include it for completeness.