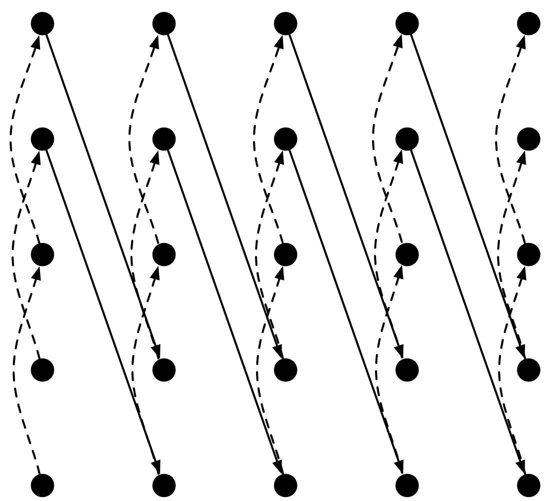
ECE 468 & 573
Problem Set 7: Dependence analysis

1. Draw the iteration space graph for the following loop:

```
for (int i = 0; i < 5; i++) {
   for (int j = 0; j < 5; j++) {
      A[i][j - 1] = A[i - 1][j + 2] + A[i][j + 1];
   }
}
```

Answer: Solid lines = flow dependences, dashed lines = anti dependences, dotted lines = output dependences.



2. Show the distance vector(s) for the loop from the previous problem.
   Answer: Flow dependence: (1, -3), Anti dependence (0, 2)

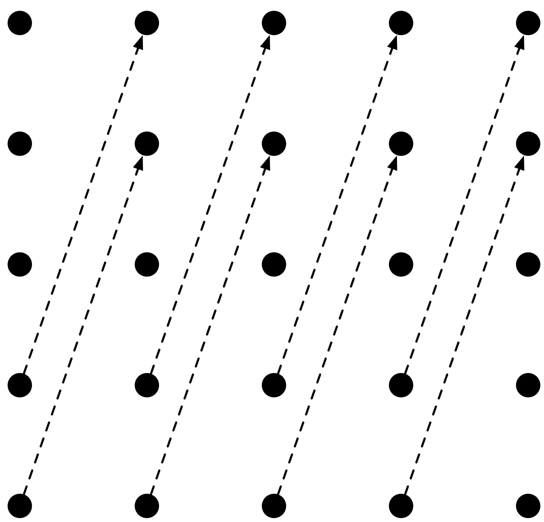
3. Show the direction vector(s) for the loop.
   Answer: Flow dependence: (+, -), Anti dependence (0, +)

4. Can the two loops be interchanged? Why or why not?
   Answer: No, because the flow dependence is a (+, -) dependence, and it will be broken by interchange.

5. Repeat the previous four steps for the following loop:

```
for (int i = 0; i < 5; i++) {
   for (int j = 0; j < 5; j++) {
      A[i][j] = A[i + 1][j + 3];
   }
}
```

Answer: Solid lines = flow dependences, dashed lines = anti dependences, dotted lines = output dependences.



There is an *anti* dependence, with distance (1, 3), and direction (+, +). Thus, interchange is legal.

6. Give an example of a doubly-nested loop *with a single statement in the loop body* that (a) has an infinite number of distance vectors, and (b) can nevertheless be interchanged.

   Answer: In order to get an infinite number of distance vectors, we need the "length" of the dependence to be different for different dependences. This means that we want the length of the dependence to be based on the loop indices, which can go from 1 to N, and hence be unbounded. There are many possible examples, but here is one:

```
for (int i = 0; i < N; i++) {
   for (int j = 0; j < N; j++) {
      A[2i + 1][j] = A[i + 1][j];
   }
}
```

Note that when i is 0, we write to location [1, j], which is read in the same iteration, so does not create a dependence. Then, when i is 1, we write to location [3, j], which will be read when i is 2. When i is 2, we write to location [5, j], which will be read when i is 4. When i is 3, we write to location [7, j], which will be read when i is 6, and so on. Thus, we get distance vectors of (1, 0), (2, 0), (3, 0), etc. However, these vectors are all (+, 0) vectors, and hence loop interchange is always possible.