Directed graphs can be represented as lists in ML by representing every directed edge by a 2-element tuple consisting of the source and the destination. For instance, the graph below

is represented as $[(0, 1), (0, 2), (1, 2)]$. Nodes are given values from an equality type and each node is given a distinct value as its label.

Write a boolean function `acyclic` which takes a graph represented as above and returns `true` if it has no cycles and `false` if it has a cycle. Any other function(s) you define must be local to `acyclic` and not visible to the outside environment.

For instance, for the above graph `acyclic` should return `true`.

- `acyclic;`
  val it = fn : ('a * 'a) list -> bool
- `acyclic nil;`
  val it = true : bool
- `acyclic [(0,0)];`
  val it = false : bool
- `acyclic [(1,2), (2,1)];`
  val it = false : bool
- `acyclic [(1,2), (1,3), (2,3)];`
  val it = true : bool
  - `acyclic ["q0", "q1"], ("q0", "q2");`
    val it = true : bool
  - `acyclic [((2),(3)), ((1),(2)), ((4),(1)), ((3),(5)), ((3),(4))];`
    val it = false : bool