```
using BenchmarkTools    , MolecularGraph
```

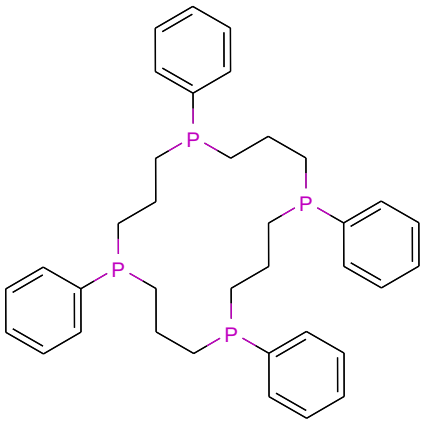
Example molecule that haunts me from my days as a chemist...

```
mol =
```

```
GraphMol([Dict(44 ⇒ 40, 8 ⇒ 8, 1 ⇒ 2), Dict(7 ⇒ 7, 2 ⇒ 3, 1 ⇒ 1), Dict(2 ⇒ 2, 3 ⇒
```

```
mol =
smilestomol("P1(C2=CC=CC=C2)CCCP(C3=CC=CC=C3)CCCP(C4=CC=CC=C4)CCCP(C5=CC=CC=C5)CCC1")
```

Definitely contains aromatic bonds:




```
HTML(drawsvg(mol, 250, 250))
```

Since `mol.edgeattrs[i].isaromatic::Bool` exists, it would be reasonable to retrieve bond aromaticity like so:

```
access_arom_from_struct (generic function with 1 method)
```

```
function access_arom_from_struct(mol::GraphMol)::BitVector
    return [bond.isaromatic for bond in mol.edgeattrs]
end
```

It is **very** fast:

```
BitVector: [false, false, false, false, false, false, false, false, false, false, false, fa
```

```
@btime access_arom_from_struct(mol)
```

```
  154.677 ns (3 allocations: 192 bytes)
```

But it does not work, because the `GraphMol` is initialized with all `isaromatic` values set to `false`:

```
AssertionError: any(access_arom_from_struct(mol))

  1. top-level scope @ Local: 1
```

```
@assert any(access_arom_from_struct(mol))
```

`isaromaticbond` returns the correct result, but it is quite slow:

```
BitVector: [false, true, true, true, true, true, true, false, false, false, false, false, t
```

```
@btime isaromaticbond(mol)
```

```
  6.440 ms (20117 allocations: 4.93 MiB)
```

Using `precalculate!` pays this speed penalty up front, allowing fast access later.

```
begin
    mol2 = deepcopy(mol)
    precalculate!(mol2)
end;
```

But this is still an order of magnitude slower than reading the `BitVector` as shown above:

```
BitVector: [false, true, true, true, true, true, true, false, false, false, false, false, t
```

```
    @btime isaromaticbond(mol2)
```

```
  4.457 µs (79 allocations: 6.86 KiB)
```