KnotTheory`SmallGirth` package

A subpackage for KnotTheory`, which reorders PD presentations to minimise girth.
January 18, 2009, Scott Morrison

```
BeginPackage["KnotTheory`SmallGirth`", {"KnotTheory`"}];
```

```
FindSmallGirthOrdering::about =
  "FindSmallGirthOrdering[K] tries to reorder the crossings in a PD presentation
    of K so that, when the crossings are read in order, the girth is
    minimised. It does this by repeatedly running a greedy algorithm, making
    random choices when they are available. FindSmallGirthOrdering[K,
    n] returns the best of n attempts; n defaults to 1000.";
```

```
Begin["`Private`"]
```

```
randomOrderPD[K_] := Module[{pd = PD[K], maxConnection, availableCrossings,
   nextCrossing, inside = {}, result = PD[], girth = 0, girths = {}},
  While[Length[pd] > 0,
   maxConnection = Max[(4 - Length[Complement[List @@ #1, inside]] &) /@ List @@ pd];
   availableCrossings =
    Cases[List @@ pd, x_ /; (4 - Length[Complement[List @@ x, inside]]) == maxConnection];
   nextCrossing = availableCrossings[[RandomInteger[{1, Length[availableCrossings]}]]];
   girth += 4 - 2 Length[Intersection[List @@ nextCrossing, inside]];
   AppendTo[girths, girth];
   inside = Union[inside, List @@ nextCrossing];
   AppendTo[result, nextCrossing];
   pd = DeleteCases[pd, nextCrossing];
  ];
  {girths, result}
 ]
```

```
FindSmallGirthOrdering[K_] := FindSmallGirthOrdering[K, 1000]
```

```
FindSmallGirthOrdering[K_, k_] := Module[{i = 0, bestSoFar = randomOrderPD[K], next},
  While[(++i) ≤ k,
   next = randomOrderPD[K];
   If[Max[next[[1]]] ≤ Max[bestSoFar[[1]]],
    If[Count[next[[1]], Max[next[[1]]]] < Count[bestSoFar[[1]], Max[bestSoFar[[1]]]],
     bestSoFar = next;
    ]
   ];
  ];
  bestSoFar[[2]]
 ]
```

```
End[]
```

```
EndPackage[]
```