This examples how long it takes to conduct the greedy algorithm .

In[ ]:=
```
<< KnotTheory`
```

```
Loading KnotTheory` version of February 2, 2020, 10:53:45.2097.
Read more at http://katlas.org/wiki/KnotTheory.
```

In[ ]:=
```
testingKnot = {X[111, 106, 112, 107], X[105, 112, 106, 113], X[102, 114, 103, 113],
   X[114, 102, 115, 101], X[115, 111, 116, 110], X[109, 117, 110, 116], X[59, 109, 60, 108],
   X[107, 61, 108, 60], X[64, 103, 65, 104], X[104, 63, 105, 64], X[58, 55, 59, 56],
   X[62, 57, 63, 58], X[56, 61, 57, 62], X[182, 66, 183, 65], X[54, 184, 55, 183],
   X[66, 182, 67, 181], X[67, 101, 68, 100], X[117, 69, 118, 68], X[99, 181, 100, 180],
   X[126, 70, 127, 69], X[127, 119, 128, 118], X[70, 120, 71, 119], X[71, 129, 72, 128],
   X[120, 126, 121, 125], X[184, 122, 185, 121], X[46, 185, 47, 186], X[124, 45, 125, 46],
   X[186, 47, 187, 48], X[122, 188, 123, 187], X[48, 123, 49, 124], X[188, 50, 189, 49],
   X[53, 50, 54, 51], X[51, 99, 52, 98], X[97, 53, 98, 52], X[189, 154, 190, 155],
   X[44, 156, 45, 155], X[156, 130, 157, 129], X[72, 157, 73, 158], X[96, 154, 97, 153],
   X[158, 179, 159, 180], X[133, 130, 134, 131], X[178, 161, 179, 162], X[73, 163, 74, 162],
   X[163, 133, 164, 132], X[131, 165, 132, 164], X[169, 75, 170, 74], X[75, 171, 76, 170],
   X[177, 77, 178, 76], X[171, 169, 172, 168], X[167, 173, 168, 172], X[173, 167, 174, 166],
   X[165, 175, 166, 174], X[79, 176, 80, 177], X[175, 78, 176, 79], X[77, 80, 78, 81],
   X[136, 81, 137, 82], X[82, 137, 83, 138], X[134, 84, 135, 83], X[138, 135, 139, 136],
   X[0, 160, 1, 159], X[160, 0, 161, 199], X[198, 140, 199, 139], X[84, 197, 85, 198],
   X[43, 197, 44, 196], X[190, 195, 191, 196], X[191, 42, 192, 43], X[194, 42, 195, 41],
   X[140, 2, 141, 1], X[85, 3, 86, 2], X[3, 192, 4, 193], X[193, 4, 194, 5],
   X[141, 86, 142, 87], X[152, 88, 153, 87], X[88, 95, 89, 96], X[94, 89, 95, 90],
   X[90, 93, 91, 94], X[92, 20, 93, 19], X[20, 92, 21, 91], X[40, 11, 41, 12],
   X[5, 149, 6, 148], X[149, 11, 150, 10], X[9, 6, 10, 7], X[147, 8, 148, 9], X[7, 146, 8, 147],
   X[142, 146, 143, 145], X[150, 144, 151, 143], X[144, 152, 145, 151], X[18, 23, 19, 24],
   X[24, 40, 25, 39], X[12, 26, 13, 25], X[13, 38, 14, 39], X[28, 22, 29, 21],
   X[22, 28, 23, 27], X[26, 29, 27, 30], X[17, 30, 18, 31], X[31, 16, 32, 17],
   X[37, 33, 38, 32], X[33, 37, 34, 36], X[35, 14, 36, 15], X[15, 34, 16, 35]};
```

In[ ]:=
```
maxWidthPresentationList[pdList_] :=
 Max[Length /@ FoldList[Complement [#1 ⋃ #2, #1 ⋂ #2] &, {}, List @@@ pdList]]
```

```
In[•]:= greedyRep[pd_PD] := Module[{todo, front, x, v, len, rep},
          todo = List @@ pd;
          front = {};
          len = 0;
          v[x_X] := Length[front ⋂ (List @@ x)];
          rep = {};
          While[Length[todo] > 0,
           x = RandomChoice[MaximalBy[todo, v]];
           rep = Join[rep, {x}];
           todo = DeleteCases[todo, x];
           front = Complement[front ⋃ (List @@ x), front ⋂ (List @@ x)];
           len = Max[len, Length[front]]
          ];
          rep]

In[•]:= Protect[factor, tries];

In[•]:= Options[bestGreedy] = {factor → maxWidthPresentationList, tries → 1000};

In[•]:= bestGreedy[pd_PD, opts : OptionsPattern[bestGreedy]] :=
         SortBy[Table[greedyRep[pd], {i, OptionValue[tries]}], OptionValue[factor][#] &][[1]]

In[•]:= widthPresentationList[crossings_] := widthPresentationList[crossings] =
          Length /@ FoldList[Complement [#1 ⋃ #2, #1 ⋂ #2] &, {}, List @@@ crossings]

In[•]:= JonesResult[pdList_] := Module[{widths, JonesQuantity},
          widths = Delete[Delete[widthPresentationList[pdList], 1], -1];
          JonesQuantity =
           Total[Range[Length[widths]] * widths * (CatalanNumber[# / 2] & /@ widths)];
          JonesQuantity]

In[•]:= Protect[factor, neededSamples, tries];

In[•]:= Options[bestGreedySamples] =
          {factor → maxWidthPresentationList, neededSamples → 5, tries → 1000};

In[•]:= bestGreedySamples[pd_PD, opts : OptionsPattern[bestGreedySamples]] := SortBy[Parallelize[
            Table[greedyRep[pd], {i, OptionValue[tries] * OptionValue[neededSamples]}]],
           OptionValue[factor][#] &][[1 ;; OptionValue[neededSamples]]]

In[•]:= RunTimeConstant = 0.000013;
```

- Now we see the difference between selecting 1000 five times and taking best five of 5000

```
In[•]:= bestThousandFiveTimes =
          Table[bestGreedy[PD @@ testingKnot, {factor → JonesResult, tries → 1000}], {i, 50}];

In[•]:= Mean[JonesResult[#] & /@ bestThousandFiveTimes] * RunTimeConstant
Out[•]= 24.1982

In[•]:= StandardDeviation[JonesResult[#] & /@ bestThousandFiveTimes] * RunTimeConstant
Out[•]= 5.22742
```

```
In[•]:= bestThousandFiveLarge = Flatten[Table[bestGreedySamples[PD @@ testingKnot,
          {factor → JonesResult, neededSamples → 5, tries → 1000}] , {i, 10}], 1];

In[•]:= Mean[JonesResult[#] & /@ bestThousandFiveLarge] * RunTimeConstant
Out[•]= 23.0796

In[•]:= StandardDeviation[JonesResult[#] & /@ bestThousandFiveLarge] * RunTimeConstant
Out[•]= 2.50291
```

- Timing of time generating it

```
In[•]:= ParallelizeBestGreedy[pd_PD, opts : OptionsPattern[bestGreedy]] :=
         SortBy[Parallelize[Table[greedyRep[pd], {i, OptionValue[tries]}]],
           OptionValue[factor][#] &][[1]]

In[•]:= AbsoluteTiming[
          ParallelizeBestGreedy[PD @@ testingKnot, {factor → JonesResult, tries → 1000}]][[1]]
Out[•]= 6.43584

In[•]:= AbsoluteTiming[Table[ParallelizeBestGreedy[
            PD @@ testingKnot, {factor → JonesResult, tries → 1000}], {i, 5}]][[1]]
Out[•]= 33.6935

In[•]:= AbsoluteTiming[bestGreedySamples[PD @@ testingKnot,
           {factor → JonesResult, neededSamples → 5, tries → 1000}]][[1]]
Out[•]= 30.1747
```

- So the alternative method takes less time and will be better
- Also, secondly, notice that we ain't going to cut the time down