```
SetDirectory@NotebookDirectory[];

Build@k_MDT :=
  Build@k = EDT[2 Range@Length@k - 1, 2 List @@ k];

Compactify@k_EDT :=
  Compactify@k = MDT @@ (({1, Sign[#[[1]] #[[2]]]}
              Mod[Abs@If[OddQ@#[[1]], #, #[[{2, 1}]]],
                2 Length@k[[1]], 1] &
             /@ (List @@ k^T)
            // Sort)^T[[2]]
       // Sign@#[[1]] # / 2 &);

KnotGraph@o_List := KnotGraph@o = (Join @@ Table[
        Array[{v, # - 1} ⟼ {v, Mod[#, 4]} &, 4] ⋃
         {{v, 0} ⟼ {o[[v]], 1},
          {v, 2} ⟼ {o[[Mod[v - 1, Length@o, 1]]], 3}},
        {v, Length@o}] // Graph);

KnotSort@l_List :=
  KnotSort@l = Sort[l, If[Length@#1 == Length@#2,
      If[#1 === Abs /@ #1,
       #2 =!= Abs /@ #2 || Order[#1, #2] == 1,
       #2 =!= Abs /@ #2 && If[Abs /@ #1 === Abs /@ #2,
         Order[#1, #2] == 1,
         Order[Abs /@ #1, Abs /@ #2] == 1]],
      Length@#1 < Length@#2] &];

PermutationConjugate[a_List, g_List] :=
  PermutationConjugate[a, g] =
   PermutationProduct[g, a, InversePermutation@g];

Convert@k_EDT := Convert@k = Minimal@Compactify@k;

IsSorted@l_List := IsSorted@l = l == KnotSort@l;
```

```
CandidateKnots@n_Integer := CandidateKnots@n =
   If[n == 0, {MDT[]}, Block[{k, l, p, y = {}},
     For[p = 0, p < n!, p++, k = MDT[];
      Do[Delete[Range@n, {List @@ k}ᵀ]
          ⟦⌊Mod[p, (n - i + 1)!] / (n - i)!⌋ + 1⟧
         // AppendTo[k, #] &;
        If[2 k⟦1⟧ - 1 > (Abs[2 i - 1 - 2 k⟦i⟧]
            // Min[#, 2 n - #] &), p += (n - i)! - 1;
         Goto@l];
        If[k⟦-1⟧ ≤ i, Do[List @@ k⟦j ;;⟧ ⋃ {}
           // If[# == Range[j, i] || # == Range[j, i] - 1,
             p += (n - i)! - 1;
             Goto@l] &, {j, If[i == n && n > 1, 2, 1], i}]],
        {i, n}];
       If[IsPlanar@k && k === Minimal@k, AppendTo[y, k]];
       Label@l];
      y]];

AlternatingKnots@n_Integer := AlternatingKnots@n =
   If[n == 0, {MDT[]}, First /@ KnotSort /@
       (Join @@ Table[Sort[k ⟷ #] & /@ Flype@k,
            {k, CandidateKnots@n}]
          // Graph
         // ConnectedComponents)
     // KnotSort];

ValidKnots@n_Integer := ValidKnots@n = If[n == 0, {MDT[]},
    Select[
     Join @@ Table[MDT @@ (c List @@ #) & /@
        AlternatingKnots@n,
       {c, Tuples[{1, -1}, n]⟦;; 2^(n-1)⟧}],
     ¬ PassReducible@# && Minimal@# == # &]
    // KnotSort];
```

```
KnotAssociation@n_Integer := KnotAssociation@n =
   Table[k → Select[ValidKnots@n, Abs /@ # == k &],
     {k, CandidateKnots@n}] // Association;

Data@n_Integer := Data@n = ToString@n <> ".m";

CreateGraph@n_Integer := CreateGraph@n = (If[n == 0, {},
      Block[
       {r, y = Join[Reverse@KnotSort@#[[ ;; 2]], {#[[3]]}] &
           /@ (Join[#, {"Flype"}] &
              /@ Union @@ AllFlypes
                /@ CandidateKnots@n ⋃
             Flatten[
              Table[{{List @@ k, List @@ #,
                   "Reidemeister 3"} & /@ R3@k,
                 {List @@ k, List @@ #, "2-Pass"} & /@ P2@k},
                {k, ValidKnots@n}], 3]) ⋃ {}},
        r = Join @@ Select[ConnectedComponents@
            Graph[#[[1]] ⟷ #[[2]] & /@ y],
           Or @@ PassReducible /@ # &];
        Sort[Select[y, ¬ MemberQ[r, #[[1]]] &],
         If[#1[[1]] == #2[[1]],
           If[#1[[2]] == #2[[2]],
            Order[#1[[3]], #2[[3]]] ≥ 0,
            IsSorted@{#1[[2]], #2[[2]]}],
           IsSorted@{#1[[1]], #2[[1]]}] &]]] >> Data@n);

DrawGraph@n_Integer :=
  DrawGraph@n = Graph @@ If[n == 0, {{{}}, {}},
     {MDT @@ #[[1]] ⟷ MDT @@ #[[2]] & /@ << Data@n}];

SetAttributes[Strand, Orderless];
Strand /: Strand[i_, j_] Strand[j_, k_] := Strand[i, k];
Strand /: Strand[i_, i_] := -q^(1/2) - q^(-1/2);
Strand /: Strand[__]^2 := -q^(1/2) - q^(-1/2);
```

```
ToPD@k_MDT := ToPD@k =
   Block[{a = Abs /@ k, n = Length@k, o, r}, o = Ordering@a;
    Do[If[PlanarGraphQ@Graph[Join @@ Table[
          Array[{v, # - 1} ⟼ {v, #} &, 3] ⋃
           {{v, 0} ⟼ {v, 3}} ⋃
           Join @@
            ({{v, #[[1]]} ⟼ {#[[2]], #[[1]] +
                    (1 - c[[v]] c[[#[[2]]]]) / 2},
                {v, 3 - #[[1]]} ⟼ {#[[2]],
                  #[[1]] + (1 + c[[v]] c[[#[[2]]]]) / 2}} &
              /@ ({#, o[[Mod[v - # / 2, n, 1]]]} & /@ {0, 2})),
          {v, n}]], r = c;
      Break[]], {c, Tuples[{1, -1}, n]}];
    PD @@
     (X## & @@@
       Array[{2 # - 1, 2 a[[#]], 2 #, Mod[2 a[[#]] + 1, 2 n, 1]}
           [[If[Sign@k[[#]] == 1,  ;; , {2, 3, 4, 1}]]]
          [[If[r[[#]] == 1,  ;; , {1, 4, 3, 2}]]] &, n])];

Writhe@k_PD :=
  Writhe@k =
   Total[If[#[[3]] == Mod[#[[5]] + 1, 2 Length@k, 1], 1, -1] &
     /@ List @@@ List @@ k];

JonesSubstitution = X_a_,b_,c_,d_ :>
   Strand[a, b] Strand[c, d] q^(-1/4) +
    Strand[a, d] Strand[b, c] q^(1/4);
```

```
JonesPolynomial@k_MDT := JonesPolynomial@k = (k
          // ToPD
         // (-q^(1/4))^(3 Writhe@#)
           (Times @@ # /. JonesSubstitution
              // Expand)
            / Strand[0, 0] &
        // Apart
       // {#, # /. q → q^-1} &
      // Sort)[[1]];

Homomorphisms[k_MDT, o_Integer] :=
  Homomorphisms[k, o] = If[k === MDT[],
    Length /@
     (Sort[Length /@ (List @@ PermutationCycles@#)[[1]]] &
        // GroupBy[Permutations@Range@o, #] &
       // Values),
    Block[{e, s, v, w = List @@@ List @@ ToPD@k},
     s = (w //. (Max@# → Min@# & /@ w[[ ;; , {3, 5}]]))^T[[
        2 ;; 4]]^T;
     e = Complement @@@
       (Table[{#⋃i[[ ;; 2]] ⟞ #⋃i, #⋃i[[2 ;;]] ⟞ #⋃i} &
              /@ Subsets[Complement[Union @@ s, i]],
            {i, s}]
           // Flatten
          // Graph
         // FindShortestPath[#,
           Sort[ConnectedComponents[#, {Union @@ s}][[1]]][[
            1]],
           Union @@ s] &
        // {#, {{}}⋃#[[ ;; -2]]}^T &);
     s = SortBy[If[Order[Position[Join @@ e, #[[1]]],
           Position[Join @@ e, #[[3]]]] == 1,
         #, Reverse@#] & /@ s,
       Max@Table[Position[Join @@ e, #[[j]]], {j, 2}] &];
```

```
        Total /@ Table[v = Array[0 &, 2 Length@k];
          v[[e[[1]]]] = g;
          If[And @@ Table[PermutationConjugate[v[[c[[1]]]],
                SortBy[w, Length[c ⋂ #] &][[-1]]
                 // If[#[[3]] == Mod[#[[5]] + 1, 2 Max@e, 1],
                   v[[c[[2]]]],
                   InversePermutation@v[[c[[2]]]]] &]
               // If[v[[c[[3]]]] === 0, (v[[c[[3]]]] = #) || True,
                 v[[c[[3]]]] == #] &,
              {c, s}], 1, 0],
          {p, (Length /@ (List @@ PermutationCycles@#)[[1]]
                 // Sort) &
             // GroupBy[Permutations@Range@o, #] &
            // Values},
          {g, Tuples[p, Length@e[[1]]]}]]];

P2@k_MDT := P2@k = Block[{a, c, n = Length@k,
      p = List @@ Build@k //
        (#ᵀ ⋃ (Abs@Reverse@# Sign@#)ᵀ)ᵀ[[2]] &, v, y = {}},
     Do[v = Abs@p[[Mod[{i, i + 1}, 2 n, 1]]];
      If[Sort@Sign@p[[Mod[{i, i + 1}, 2 n, 1]]] == {-1, 1},
       Do[If[Total@Mod[l, 2] == 2,
         c = Range @@@ Partition[l + {1, -1, 1, -1}, 2];
         If[¬ MemberQ[Join @@ c, i], l = RotateLeft@l;
          c = Mod[Range @@@ Partition[l + {1, -1, 1, 2 n - 1},
              2], 2 n, 1]];
         If[Length[Join @@ c] < 2 n - 4
           && v ⋃ Join @@ c == Abs@p[[Join @@ c]] ⋃
             Mod[{i, i + 1}, 2 n, 1],
          AppendTo[y, Convert[Build@k /. x_Integer :>
              If[a = Abs@x;
               Length[v ⋂ l[[ ;; 2]]] == 1,
               Mod[If[MemberQ[v ⋃ Join @@ c, a],
                  If[MemberQ[c[[1]], a],
```

```
                a + If[Mod[Abs@p[[l[[1]]]] - i, 2 n] > 1,
                  1, -1],
                If[MemberQ[c[[2]], a],
                 a + If[Mod[Abs@p[[l[[3]]]] - i, 2 n] > 1,
                   1, -1],
                 (l[[{2, 1, 4, 3}]] + {-1, 1, -1, 1})
                  [[Position[l, If[OddQ[l[[1]] + l[[2]]],
                     Total@v - a, a]][[1, 1]]]]],
               a], 2 n, 1]
             If[MemberQ[v ⋃ Join @@ c, a] || EvenQ@a,
              If[Mod[a - i, 2 n] ≤ 1 ||
                OddQ@a && ¬ MemberQ[l, a], -Sign@p[[a]],
               1],
              Sign@p[[a]]],
            Mod[If[MemberQ[v, a],
               SortBy[Delete[l, FirstPosition[l, #] & /@
                    v],
                 Mod[#, 2] &][[Mod[x, 2] + 1]],
               a + If[MemberQ[Join @@ c, a], 0,
                 If[MemberQ[v,
                   l[[Ordering[Mod[a - l, 2 n, 1]][[1]]]],
                  -1, 1]]], 2 n, 1]
             If[OddQ@x, Sign@p[[a]]
               If[MemberQ[v ⋃ Join @@ c, a], 1, -1],
              1]]]]]],
   {l, Sort@Join[#, v] & /@
     Subsets[Delete[Range[2 n],
       Mod[{{i, i + 1}}^T, 2 n, 1]], {2}]}]],
 {i, 2 n}];
KnotSort[Minimal /@ y ⋃ {}]];
```

```
R1@k_MDT :=
  R1@k =
   Table[EDT @@ ((Map[# + If[Abs@# ≥ 2 i, 2 Sign@#, 0] &,
              List @@ Build@k^T, {2}]
            ⋃ {{2 i, 2 i + 1}})^T)
       // Convert, {i, Length@k}] ⋃ {} // KnotSort;

R3@k_MDT := R3@k = Block[{b, f, n = Length@k,
     p = List @@ Build@k //
       (#^T ⋃ (Abs@Reverse@# Sign@#)^T)^T[[2]] &, v, y = {}},
    b = Abs@p // #[[Mod[#[[1]] + {1, -1}, 2 n, 1]]] &;
    Do[f = Mod[Abs@p[[i]] + {1, -1}, 2 n, 1]
       // If[OddQ@i, Abs@p[[#]], #] &;
     Do[If[{c, i - 1, i}
       // Total[Sign@p[[#]]]^2 == 1 &&
         MemberQ[(Abs@p[[#]] ⋃ #)[[2 ;; 3]], i] &,
       v = p[[{Abs@p[[c]], Abs@p[[i - Mod[i, 2]]],
         i - Mod[i + 1, 2]}]] / 2;
       If[DuplicateFreeQ@v,
        AppendTo[y, k /.
          (v[[#1]] → -Abs@v[[#2]] Sign@v[[#3]] & @@@
           {{1, 2, 3}, {2, 3, 1}, {3, 1, 2}})]]],
      {c, b ⋂ f}];
     b = f, {i, 2, 2 n}];
    KnotSort[Minimal /@ y ⋃ {}]];

Flype@k_MDT :=
  Flype@k = KnotSort[AllFlypes[Abs /@ k, {k}]^T[[2]]];
```

```
AllFlypes[k_MDT, l_List] :=
  AllFlypes[k, l] = Block[{a, c, e, n = Length@k,
     p = List @@ Build@k // (#ᵀ ⋃ Reverse@#ᵀ)ᵀ⟦2⟧ &,
     y = {}},
    Do[c = Mod[2 i - 1 + s⟦1⟧ Range@o, 2 n, 1];
     For[
      e = Max@Mod[Complement[p⟦c⟧, c] - s⟦2⟧ 2 k⟦i⟧, 2 n, 1],
      e < Mod[s⟦2⟧ (2 i - 1 - 2 k⟦i⟧), 2 n, 1],
      e++,
      c = c ⋃ Mod[2 k⟦i⟧ + s⟦2⟧ Range@e, 2 n, 1];
      If[Sort@p⟦c⟧ == c,
       y = Join[y, {l, Convert /@ (Mod[a = Abs@#; a +
                   If[Mod[s⟦1⟧ (a - 2 i + 1), 2 n, 1] ≤ o,
                    -s⟦1⟧,
                    If[Mod[s⟦2⟧ (a - 2 k⟦i⟧), 2 n, 1] ≤ e,
                     -s⟦2⟧,
                     If[a == 2 i - 1, s⟦1⟧ o,
                      If[a == 2 k⟦i⟧, s⟦2⟧ e, 0]]]], 2 n, 1]
                  Sign@# &
                 // Map[#, Build /@ l, {3}] &)}ᵀ]]],
     {i, n},
     {s, {{1, 1}, {1, -1}, {-1, 1}}},
     {o, 2, Mod[s⟦1⟧ (2 k⟦i⟧ - 2 i + 1), 2 n, 1] - 1}];
    KnotSort /@ (y ⋃ {})];

IsPlanar@k_MDT :=
  IsPlanar@k =
   PlanarGraphQ@KnotGraph@Ordering@Abs[List @@ k];

Shift[k_MDT, o_Integer, s_Integer] :=
  Compactify[EDT @@ (Sign[List @@ Build@k]
      (o + Abs[List @@ Build@k] s + 2 Length@k))];
```

```
Minimal@k_MDT :=
  Minimal@k =
   (Array[Shift[k, #1, (-1)^#2] &, {2 Length@k, 2}]
       // Join @@ # &
      // KnotSort)[[1]];

PassReducible@k_MDT := PassReducible@k =
   Block[{l, n = Length@k, p = List @@ Build@k
       // (#^T ⋃ (Abs@Reverse@#
                 Sign /@ List @@ Build[MDT @@ (-List @@ k)])^T)^T[[2
     v = True},
    Do[If[(Sign@p[[Mod[i + j (Range@o - 1), 2 n, 1]]] ⋃ {})^2 ==
       {1},
      If[o == 3, Goto@l];
      Do[If[Total@Mod[e, 2] == 2,
        If[(Mod[e, 2 n, e[[1]]]
                 // Partition[#, 2] &
                // Range @@@ # &)[[ ;; , 2 ;; -2]]
              // Mod[#, 2 n, 1] &
             // Union @@ # &
            // # == Abs@p[[#]] ⋃ {} &, Goto@l]],
       {e, Select[Table[SortBy[{c, i} ⋃
                 Abs@p[[Mod[{i, i + j}, 2 n, 1]]],
               Mod[#, 2 n, i] &], {c, 2 n}],
           Length@# == 4 &]^T[[
         If[j == 1, {2, 3, 4, 1}, ;;]]^T}]],
     {o, {3, 2}}, {i, 2 n}, {j, {1, -1}}];
    v = False;
    Label@l;
    v];
```

```
IsReducible@k_MDT := IsReducible@k =
   Or @@ Array[Mod[# - Abs@k[[#]], Length@k] ≤ 1 &,
      Length@k] || PassReducible@k;

Invariants[a_List, t_List, o_Integer] :=
  Invariants[a, t, o] =
   Block[{l = a, r = t, i}, Do[If[(i = i /@ l) ≠ {},
      r = Select[{l, i}ᵀ, Count[i, #[[2]]] == 1 &]ᵀ[[1]] ⋃ r;
      l = Complement[l, r]],
     {i, {If[# === Abs /@ #, #, 0] &, JonesPolynomial,
       Homomorphisms[#, o] &}}];
    {l, r}];
```

```
RolfsenTable[n_Integer, o_Integer] :=
  RolfsenTable[n, o] = If[n == 0, {MDT[]},
    Block[{a = ValidKnots@n, e, l, r, t = {}, v},
     For[l = 0, a ≠ {}, l++,
      e =
       Join @@ Table[Sort[k ⟷ #] & /@
           If[l == 0, R3@k ⋃ P2@k, R1@k], {k, a}] ⋃
        If[l == 0, Sort[#[[1]] ⟷ #[[2]] ] &
          /@ Union @@ (AllFlypes[#, KnotAssociation[n]@#] &
             /@ AlternatingKnots@n), {}];
      r = Join @@ Select[ConnectedComponents@Graph@e,
         Length /@ # ⋃ {} == {n}
          && Or @@ PassReducible /@ # &];
      e = Select[e, r ⋂ List @@ # == {} &];
      While[v = Join @@ List @@@ e;
       a = Complement[Select[v, Length@# ≠ n &], a];
       a ≠ {},
       e = e ⋃ Join @@ Table[Sort[k ⟷ #] & /@ R3@k, {k, a}];
       a = v];
      {a, t} = Invariants[First /@ KnotSort /@ Select[
           ConnectedComponents@Graph@e,
           Length /@ # ⋃ {} ≠ {n} ||
            Nor @@ PassReducible /@ # &], t, o]];
     KnotSort@t]];

Timing[{Length@#, #} &[
  Join @@ Array[RolfsenTable[#, 5] &, 11, 0]]]
```

```
{926.734375, {250, {MDT[], MDT[2, 3, 1],
   MDT[2, 3, 4, 1], MDT[2, 4, 5, 1, 3], MDT[3, 4, 5, 1, 2],
   MDT[2, 4, 5, 1, 6, 3], MDT[2, 4, 5, 6, 1, 3],
   MDT[2, 4, 6, 5, 1, 3], MDT[2, 4, 5, 6, 1, 7, 3],
   MDT[2, 4, 6, 1, 7, 3, 5], MDT[2, 5, 6, 7, 1, 4, 3],
   MDT[2, 5, 7, 6, 1, 4, 3], MDT[3, 5, 6, 7, 1, 2, 4],
   MDT[3, 5, 6, 7, 2, 1, 4], MDT[4, 5, 6, 7, 1, 2, 3],
```

```
MDT[2, 4, 5, 7, 1, 8, 3, 6], MDT[2, 4, 6, 1, 7, 3, 8, 5],
MDT[2, 4, 6, 1, 7, 8, 3, 5], MDT[2, 4, 6, 1, 8, 7, 3, 5],
MDT[2, 4, 7, 5, 1, 8, 3, 6], MDT[2, 5, 6, 7, 1, 8, 3, 4],
MDT[2, 5, 6, 7, 1, 8, 4, 3], MDT[2, 5, 6, 7, 8, 1, 3, 4],
MDT[2, 5, 6, 7, 8, 1, 4, 3], MDT[2, 5, 7, 8, 6, 1, 4, 3],
MDT[2, 5, 8, 7, 6, 1, 4, 3], MDT[3, 4, 5, 6, 7, 8, 1, 2],
MDT[3, 4, 6, 1, 7, 8, 2, 5], MDT[3, 4, 6, 7, 2, 8, 1, 5],
MDT[3, 4, 7, 6, 2, 8, 1, 5], MDT[3, 5, 6, 7, 8, 2, 1, 4],
MDT[3, 5, 6, 8, 7, 2, 1, 4], MDT[3, 6, 5, 8, 7, 2, 1, 4],
MDT[2, 4, -6, 1, -7, -3, -8, -5],
MDT[2, 4, -6, 1, 7, -3, 8, 5], MDT[2, 4, -6, 1,
 -7, -8, -3, -5], MDT[2, 4, 5, 7, 1, 8, 3, 9, 6],
MDT[2, 4, 5, 7, 1, 8, 9, 3, 6], MDT[2, 4, 5, 7, 1, 9, 8, 3, 6],
MDT[2, 4, 6, 1, 8, 3, 9, 5, 7], MDT[2, 4, 6, 1, 8, 7, 3, 9, 5],
MDT[2, 4, 6, 7, 1, 8, 9, 5, 3], MDT[2, 4, 7, 1, 8, 9, 3, 6, 5],
MDT[2, 4, 7, 1, 9, 8, 3, 6, 5], MDT[2, 4, 7, 5, 1, 8, 9, 3, 6],
MDT[2, 4, 7, 5, 1, 9, 8, 3, 6], MDT[2, 4, 7, 6, 1, 8, 9, 5, 3],
MDT[2, 5, 6, 7, 1, 9, 8, 3, 4], MDT[2, 5, 6, 7, 1, 9, 8, 4, 3],
MDT[2, 5, 6, 7, 8, 1, 3, 9, 4], MDT[2, 5, 6, 7, 8, 1, 9, 4, 3],
MDT[2, 5, 6, 8, 1, 4, 9, 3, 7], MDT[2, 5, 6, 8, 7, 1, 9, 4, 3],
MDT[2, 5, 7, 6, 8, 1, 3, 9, 4], MDT[2, 5, 7, 8, 1, 9, 4, 3, 6],
MDT[2, 5, 7, 8, 6, 1, 9, 3, 4], MDT[2, 5, 7, 8, 6, 1, 9, 4, 3],
MDT[2, 5, 8, 7, 1, 9, 4, 3, 6], MDT[2, 6, 7, 8, 9, 1, 5, 3, 4],
MDT[2, 6, 7, 8, 9, 1, 5, 4, 3], MDT[2, 6, 8, 9, 7, 1, 4, 5, 3],
MDT[2, 6, 8, 9, 7, 1, 5, 4, 3], MDT[2, 6, 9, 8, 7, 1, 5, 4, 3],
MDT[3, 4, 5, 8, 7, 9, 2, 1, 6], MDT[3, 5, 7, 6, 8, 1, 9, 2, 4],
MDT[3, 5, 7, 9, 2, 8, 1, 4, 6], MDT[3, 5, 7, 9, 2, 8, 4, 1, 6],
MDT[3, 5, 7, 9, 8, 1, 4, 2, 6], MDT[3, 6, 7, 8, 9, 1, 2, 5, 4],
MDT[3, 6, 7, 8, 9, 2, 1, 5, 4], MDT[3, 6, 7, 9, 8, 1, 2, 5, 4],
MDT[3, 6, 7, 9, 8, 2, 1, 5, 4], MDT[3, 8, 7, 6, 2, 1, 9, 5, 4],
MDT[4, 6, 7, 8, 9, 1, 2, 3, 5], MDT[4, 6, 7, 8, 9, 1, 3, 2, 5],
MDT[4, 6, 8, 7, 9, 2, 1, 3, 5], MDT[5, 6, 7, 8, 9, 1, 2, 3, 4],
MDT[2, 4, 5, -7, 1, -8, -3, -9, -6],
MDT[2, 4, 5, -7, 1, 8, -3, 9, 6],
```

```
MDT[2, 4, 5, 7, 1, -8, 3, -9, -6],
MDT[2, 4, 5, -7, 1, -8, -9, -3, -6],
MDT[2, 5, -7, -6, -8, 1, -3, -9, -4],
MDT[2, 5, -7, -6, 8, 1, -3, 9, 4],
MDT[3, 4, 5, 8, 7, -9, 2, 1, -6],
MDT[3, -5, -7, 6, -8, -1, 9, -2, -4],
MDT[2, 4, 5, 7, 1, 8, 9, 3, 10, 6],
MDT[2, 4, 5, 7, 1, 9, 3, 10, 6, 8],
MDT[2, 4, 5, 7, 1, 9, 8, 3, 10, 6],
MDT[2, 4, 5, 8, 1, 9, 10, 3, 7, 6],
MDT[2, 4, 5, 8, 1, 10, 9, 3, 7, 6],
MDT[2, 4, 6, 1, 7, 9, 3, 10, 5, 8],
MDT[2, 4, 6, 1, 8, 3, 9, 5, 10, 7],
MDT[2, 4, 6, 1, 8, 3, 9, 10, 5, 7],
MDT[2, 4, 6, 1, 8, 3, 10, 9, 5, 7],
MDT[2, 4, 6, 1, 9, 7, 3, 10, 5, 8],
MDT[2, 4, 6, 7, 1, 8, 10, 9, 5, 3],
MDT[2, 4, 6, 9, 1, 8, 10, 3, 5, 7],
MDT[2, 4, 7, 1, 8, 9, 3, 10, 5, 6],
MDT[2, 4, 7, 1, 8, 9, 3, 10, 6, 5],
MDT[2, 4, 7, 1, 8, 9, 10, 3, 5, 6],
MDT[2, 4, 7, 1, 8, 9, 10, 3, 6, 5],
MDT[2, 4, 7, 1, 9, 8, 3, 6, 10, 5],
MDT[2, 4, 7, 1, 9, 10, 8, 3, 6, 5],
MDT[2, 4, 7, 1, 10, 9, 8, 3, 6, 5],
MDT[2, 4, 7, 5, 1, 9, 3, 10, 6, 8],
MDT[2, 4, 7, 6, 1, 8, 10, 9, 5, 3],
MDT[2, 4, 8, 5, 1, 9, 10, 3, 7, 6],
MDT[2, 4, 8, 5, 1, 10, 9, 3, 7, 6],
MDT[2, 4, 9, 6, 1, 8, 10, 3, 5, 7],
MDT[2, 5, 6, 7, 8, 1, 10, 9, 4, 3],
MDT[2, 5, 6, 7, 9, 1, 3, 10, 4, 8],
MDT[2, 5, 6, 7, 9, 1, 8, 3, 10, 4],
MDT[2, 5, 6, 8, 1, 4, 9, 10, 3, 7],
```

```
MDT[2, 5, 6, 8, 1, 4, 10, 9, 3, 7],
MDT[2, 5, 6, 8, 1, 10, 3, 9, 4, 7],
MDT[2, 5, 6, 8, 1, 10, 4, 9, 3, 7],
MDT[2, 5, 6, 8, 7, 1, 10, 9, 4, 3],
MDT[2, 5, 6, 8, 9, 1, 10, 3, 4, 7],
MDT[2, 5, 6, 8, 9, 1, 10, 4, 3, 7],
MDT[2, 5, 6, 8, 10, 1, 4, 9, 3, 7],
MDT[2, 5, 7, 6, 1, 8, 9, 10, 4, 3],
MDT[2, 5, 7, 6, 9, 1, 3, 10, 4, 8],
MDT[2, 5, 7, 6, 9, 1, 8, 3, 10, 4],
MDT[2, 5, 7, 8, 1, 4, 9, 10, 6, 3],
MDT[2, 5, 7, 8, 1, 9, 4, 3, 10, 6],
MDT[2, 5, 7, 8, 1, 9, 10, 3, 4, 6],
MDT[2, 5, 7, 8, 1, 9, 10, 4, 3, 6],
MDT[2, 5, 7, 8, 1, 10, 9, 3, 4, 6],
MDT[2, 5, 7, 8, 1, 10, 9, 4, 3, 6],
MDT[2, 5, 7, 9, 1, 8, 3, 10, 4, 6],
MDT[2, 5, 7, 9, 1, 8, 4, 10, 6, 3],
MDT[2, 5, 7, 9, 1, 8, 10, 4, 6, 3],
MDT[2, 5, 8, 6, 1, 4, 9, 10, 3, 7],
MDT[2, 5, 8, 6, 1, 4, 10, 9, 3, 7],
MDT[2, 5, 8, 7, 1, 4, 9, 10, 6, 3],
MDT[2, 5, 8, 7, 1, 9, 4, 3, 10, 6],
MDT[2, 5, 8, 7, 1, 10, 4, 9, 3, 6],
MDT[2, 5, 8, 9, 6, 1, 10, 4, 3, 7],
MDT[2, 5, 9, 8, 6, 1, 10, 4, 3, 7],
MDT[2, 6, 7, 8, 9, 1, 5, 10, 4, 3],
MDT[2, 6, 7, 8, 9, 1, 10, 3, 4, 5],
MDT[2, 6, 7, 8, 9, 1, 10, 3, 5, 4],
MDT[2, 6, 7, 8, 9, 1, 10, 5, 4, 3],
MDT[2, 6, 7, 8, 9, 10, 1, 3, 4, 5],
MDT[2, 6, 7, 8, 9, 10, 1, 3, 5, 4],
MDT[2, 6, 7, 8, 9, 10, 1, 5, 4, 3],
MDT[2, 6, 7, 8, 10, 9, 1, 4, 3, 5],
```

```
MDT[2, 6, 7, 9, 8, 1, 5, 10, 4, 3],
MDT[2, 6, 7, 9, 8, 1, 10, 5, 4, 3],
MDT[2, 6, 7, 9, 8, 10, 1, 5, 4, 3],
MDT[2, 6, 8, 7, 9, 1, 4, 10, 5, 3],
MDT[2, 6, 8, 7, 9, 1, 10, 3, 5, 4],
MDT[2, 6, 8, 9, 7, 1, 5, 10, 3, 4],
MDT[2, 6, 8, 9, 7, 1, 5, 10, 4, 3],
MDT[2, 6, 8, 9, 10, 7, 1, 5, 3, 4],
MDT[2, 6, 8, 9, 10, 7, 1, 5, 4, 3],
MDT[2, 6, 8, 10, 9, 1, 4, 3, 5, 7],
MDT[2, 6, 9, 10, 7, 8, 1, 5, 4, 3],
MDT[2, 6, 9, 10, 8, 7, 1, 5, 4, 3],
MDT[2, 6, 10, 9, 8, 7, 1, 5, 4, 3],
MDT[3, 4, 5, 7, 8, 9, 10, 1, 2, 6],
MDT[3, 4, 5, 7, 8, 10, 9, 1, 2, 6],
MDT[3, 4, 6, 1, 8, 2, 9, 10, 5, 7],
MDT[3, 4, 7, 1, 8, 9, 2, 10, 5, 6],
MDT[3, 4, 7, 1, 8, 9, 2, 10, 6, 5],
MDT[3, 4, 7, 1, 8, 9, 10, 2, 5, 6],
MDT[3, 4, 7, 1, 8, 9, 10, 2, 6, 5],
MDT[3, 4, 7, 8, 2, 9, 10, 1, 5, 6],
MDT[3, 4, 7, 8, 2, 9, 10, 1, 6, 5],
MDT[3, 4, 7, 9, 8, 2, 10, 5, 1, 6],
MDT[3, 4, 8, 7, 2, 9, 10, 1, 5, 6],
MDT[3, 4, 8, 7, 2, 9, 10, 1, 6, 5],
MDT[3, 4, 9, 7, 8, 2, 10, 1, 5, 6],
MDT[3, 5, 6, 7, 9, 8, 10, 1, 2, 4],
MDT[3, 5, 6, 10, 9, 8, 4, 1, 2, 7],
MDT[3, 5, 7, 1, 8, 9, 10, 4, 2, 6],
MDT[3, 5, 7, 1, 8, 10, 9, 4, 2, 6],
MDT[3, 5, 7, 8, 1, 9, 2, 10, 4, 6],
MDT[3, 5, 7, 8, 2, 9, 1, 10, 6, 4],
MDT[3, 5, 7, 8, 9, 2, 10, 1, 4, 6],
MDT[3, 5, 7, 9, 1, 8, 10, 2, 4, 6],
```

```
MDT[3, 5, 7, 9, 8, 2, 10, 1, 4, 6],
MDT[3, 5, 8, 7, 1, 9, 4, 10, 2, 6],
MDT[3, 5, 8, 7, 9, 2, 10, 1, 6, 4],
MDT[3, 5, 8, 10, 7, 1, 9, 2, 4, 6],
MDT[3, 5, 8, 10, 7, 2, 9, 1, 6, 4],
MDT[3, 5, 9, 6, 2, 8, 10, 4, 1, 7],
MDT[3, 5, 9, 7, 1, 8, 10, 4, 2, 6],
MDT[3, 5, 9, 7, 8, 2, 10, 4, 1, 6],
MDT[3, 5, 9, 8, 7, 2, 10, 4, 1, 6],
MDT[3, 5, 10, 7, 8, 9, 2, 4, 1, 6],
MDT[3, 6, 7, 8, 9, 1, 2, 10, 4, 5],
MDT[3, 6, 7, 8, 9, 1, 2, 10, 5, 4],
MDT[3, 6, 7, 8, 9, 2, 1, 10, 5, 4],
MDT[3, 6, 7, 8, 9, 10, 2, 1, 4, 5],
MDT[3, 6, 7, 8, 9, 10, 2, 1, 5, 4],
MDT[3, 6, 7, 9, 10, 8, 2, 1, 5, 4],
MDT[3, 6, 7, 10, 9, 8, 2, 1, 5, 4],
MDT[3, 7, 6, 8, 9, 10, 2, 1, 4, 5],
MDT[3, 7, 6, 8, 9, 10, 2, 1, 5, 4],
MDT[3, 7, 6, 9, 10, 8, 2, 1, 5, 4],
MDT[3, 7, 6, 10, 9, 8, 2, 1, 5, 4],
MDT[3, 8, 6, 7, 9, 2, 10, 1, 4, 5],
MDT[3, 8, 6, 7, 9, 2, 10, 1, 5, 4],
MDT[3, 8, 9, 7, 1, 2, 10, 4, 5, 6],
MDT[4, 5, 6, 7, 8, 9, 10, 1, 2, 3],
MDT[4, 5, 7, 8, 1, 9, 10, 3, 2, 6],
MDT[4, 5, 8, 7, 1, 9, 10, 3, 2, 6],
MDT[2, 4, 5, -7, 1, -8, -9, -3, -10, -6],
MDT[2, 4, 5, -7, 1, 8, 9, -3, 10, 6],
MDT[2, 4, 5, -7, 1, -9, -3, -10, -6, -8],
MDT[2, 4, -6, 1, -7, -9, -3, -10, -5, -8],
MDT[2, 4, -6, 1, 7, 9, -3, 10, 5, 8],
MDT[2, 4, 6, 1, -7, -9, 3, -10, -5, -8],
MDT[2, 4, -6, 1, -8, -3, -9, -5, -10, -7],
```

```
MDT[2, 4, -6, 1, 8, -3, 9, 5, 10, 7],
MDT[2, 4, 6, 1, -8, 3, -9, -5, -10, -7],
MDT[2, 4, -6, 1, -8, -3, -9, -10, -5, -7],
MDT[2, 4, -6, 1, 8, -3, 9, 10, 5, 7],
MDT[2, 4, 6, 1, -8, 3, -9, -10, -5, -7],
MDT[2, 4, -6, 1, -8, -3, -10, -9, -5, -7],
MDT[2, 4, -6, -9, 1, -8, -10, -3, -5, -7],
MDT[2, 4, -7, 1, -8, -9, -3, -10, -5, -6],
MDT[2, 4, -7, 1, 8, 9, -3, 10, 5, 6],
MDT[2, 4, 7, 1, -8, -9, 3, -10, -5, -6],
MDT[2, 4, -7, 1, -8, -9, -3, -10, -6, -5],
MDT[2, 4, -7, 1, 8, 9, -3, 10, 6, 5],
MDT[2, 4, 7, 1, -8, -9, 3, -10, -6, -5],
MDT[2, 4, -7, 1, -8, -9, -10, -3, -5, -6],
MDT[2, 4, -7, 1, -8, -9, -10, -3, -6, -5],
MDT[2, 4, -9, -6, 1, -8, -10, -3, -5, -7],
MDT[2, 5, -7, 6, 1, 8, 9, -10, 4, -3],
MDT[2, 5, -7, -8, 1, -9, -4, -3, -10, -6],
MDT[2, 5, -7, -8, 1, 9, -4, -3, 10, 6],
MDT[2, 5, -7, -8, 1, -9, -10, -3, -4, -6],
MDT[2, 5, -7, -8, 1, -9, -10, -4, -3, -6],
MDT[2, 5, -7, -8, 1, 9, 10, -4, -3, 6],
MDT[2, 5, 7, 8, 1, -9, -10, 4, 3, -6],
MDT[2, 6, -8, -7, -9, 1, -4, -10, -5, -3],
MDT[2, 6, -8, 7, -9, 1, 4, -10, -5, -3],
MDT[2, 6, 8, -7, 9, 1, -4, 10, 5, 3],
MDT[3, 4, 5, 7, 8, -9, -10, 1, 2, -6],
MDT[3, 4, 5, 7, 8, -10, -9, 1, 2, -6],
MDT[3, 4, 6, 1, -8, 2, -9, -10, -5, -7],
MDT[3, 4, 7, 9, 8, 2, -10, 5, 1, -6],
MDT[3, -5, -6, 7, -9, -8, 10, -1, -2, -4],
MDT[3, 5, 7, 8, 9, 2, -10, 1, 4, -6],
MDT[3, 5, 7, 9, 8, 2, -10, 1, 4, -6],
MDT[3, -5, -8, 7, -1, -9, 4, 10, -2, -6],
```

```
MDT[3, -5, -9, 7, -1, -8, 10, 4, -2, 6]}}}
```