```
div-graph-iso  THM

⊢  DivGraph_2 ≡ DivGraph_1
| by (Unfolds [‘divides-graph1‘;‘divides-graph2‘] 0)
|    THEN
|    (Using [‘H’,Graph(i,j:ℕ+. ∃c:ℕ+. j = i·c)]
|     ((BackThru
|       Thm* ∀G,H,K:Graph. G ≡ H ⇒ H ≡ K ⇒ G ≡ K
|       {graph_DASH_isomorphic_transitivity})
|      THENA
|      (Auto THEN Easy)))
|
|\
| ⊢  Graph(i:ℕ+ --> i·n | n:ℕ+) ≡ Graph(i,j:ℕ+. ∃c:ℕ+. j = i·c)
| | by (BackThru Thm: fun_DASH_graph_DASH_rel_DASH_graph)
| |    THEN
| |    (Try
| |     (FwdThru
| |      Thm* ∀a,b:ℤ, n:ℤ-0. n·a = n·b ⇒ a = b    {mul_cancel_in_eq}
| |      [-1]))
| |    THEN
| |    (Try Easy)
| |
| .....decidable? .....
| 1. i : ℕ+
| 2. j : ℕ+
| ⊢  Dec(∃c:ℕ+. j = i·c)
| | by (Assert Dec(i | j)) THEN (Unfold ‘divides‘ -1)
| |    THEN
| |    (RepeatFor 2 (ParallelOp -1))
| |    THEN
| |    (Try Easy)
| |
| |\
| | 1. i : ℕ+
| | 2. j : ℕ+
| | 3. ∃c:ℤ. j = i·c
| | ⊢  ∃c:ℕ+. j = i·c  by (ParallelOp -1) THEN (Try Easy)
| | |
| | .....set predicate .....
| | 1. i : ℕ+
| | 2. j : ℕ+
| | 3. c : ℤ
| | 4. j = i·c
| | ⊢  0<c
| |   by (Inst
| |       Thm* ∀a,b:ℤ. a·b>0 ⟺ a>0 & b>0 ∨ a<0 & b<0
| |       {pos_mul_arg_bounds}
| |       [i;c])
| |      THEN
| |      (Analyze -1)
| |      THEN
| |      (Analyze -2)
| |
| .\
```

```
| . 1. i : ℕ+
| . 2. j : ℕ+
| . 3. ¬(∃c:ℤ. j = i·c)
| . ⊢  ¬(∃c:ℕ+. j = i·c)
| .   by (ParallelOp -1) THEN (Try Easy)
| .      THEN
| .      (RepeatFor 2 (ParallelOp -1))
|
.\
. ⊢  Graph(i,j:ℕ+. ∃c:ℕ+. j = i·c) ≡ Graph(i,j:ℕ+. i | j)
.   by (BackThru
.       Thm* ∀R1,R2:(T→T→Prop).
.             (∀x,y:T. R1(x,y) ⟺ R2(x,y))
.             ⇒
.             Graph(x,y:T. R1(x,y)) ≡ Graph(x,y:T. R2(x,y))
.       {rel_DASH_graph_functionality})
.      THEN
.      (Try (ParallelOp -1))
.      THEN
.      (Try Easy)
END
```