Keliaujančio pirklio uždavinys.

Reikia aplankyti 7 šalies miestus, pradedant kelionę iš Vilniaus ir grįžti i į tą patį miestą.
Reikia rasti trumpiausią maršrutą. Atstumų lentelė yra duota faile atstumai.pdf.
Išsprękite dviem būdais: suvedant į sveikaskaitį optimizavimo uždavinį ir patikrinant visus galimus variantus.
Miestai:
Vilnius, Klaipeda, Šiauliai, Telšiai, Marijampole, Utena, Palanga, Taurage

Patikrinsime visus variantus. Variantų skaičius yra 5040.

```
(%i1) S:listify(permutations([2,3,4,5,6,7,8]))$
```

```
(%i2) n:length(S);
(%o2) 5040
```

```
(%i3) C:matrix(
        [0,308,212,285,137,97,330,228],
        [308,0,157,88,231,326,25,109],
        [212,157,0,72,199,175,146,101],
        [285,88,72,0,216,248,78,96],
        [137,231,199,216,0,193,253,125],
        [97,326,175,248,193,0,322,255],
        [330,25,146,78,253,322,0,131],
        [228,109,101,96,125,255,131,0]
       );
```

(%o3)

0	308	212	285	137	97	330	228
308	0	157	88	231	326	25	109
212	157	0	72	199	175	146	101
285	88	72	0	216	248	78	96
137	231	199	216	0	193	253	125
97	326	175	248	193	0	322	255
330	25	146	78	253	322	0	131
228	109	101	96	125	255	131	0

```
(%i4) f(a,b,c,d,e,f,g):=C[1,a]+C[a,b]+C[b,c]+C[c,d]+C[d,e]+C[e,f]+C[f,g]+C[
(%o4) f(a,b,c,d,e,f,g):=C_{1,a}+C_{a,b}+C_{b,c}+C_{c,d}+C_{d,e}+C_{e,f}+C_{f,g}+C_{g,1}
```

```
(%i5) ff:makelist(funmake(f,S[k]),k,1,n)$ ev(ff, nouns)$
```

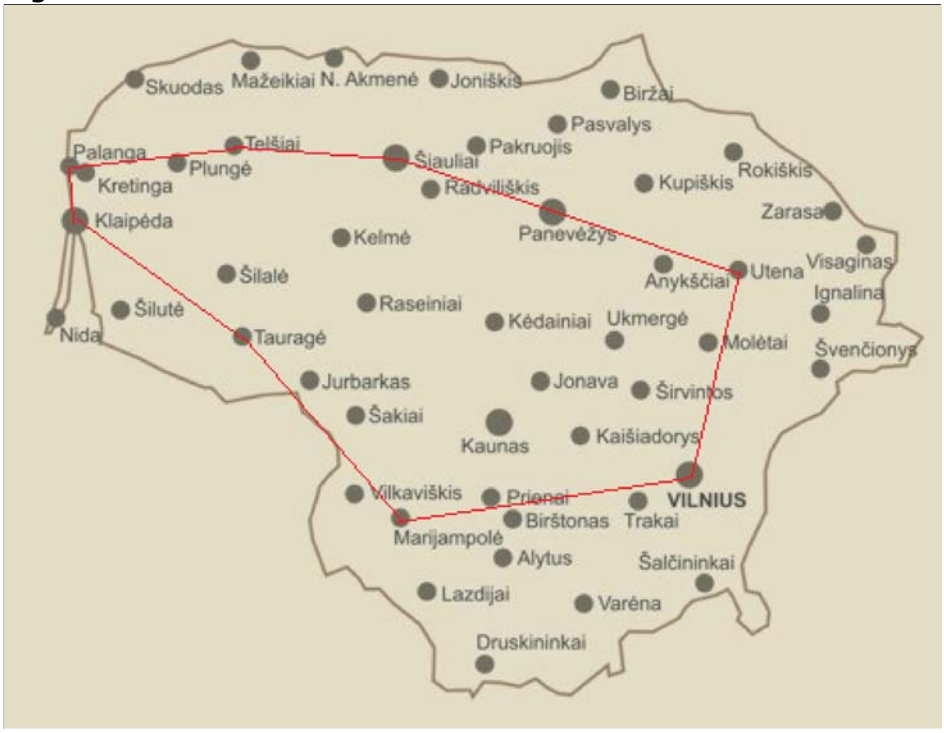
```
(%i7) m:lmin(%);
(%o7) 818
```

```
(%i8) sublist_indices (ff,lambda([x],x=m));
(%o8) [2781,3038]
```

```
(%i9) ff[2781]; ev(%);
(%o9) f(5,8,2,7,4,3,6)
(%o10) 818
```

```
(%i11) ff[3038]; ev(%);
(%o11) f(6,3,4,7,2,8,5)
(%o12) 818
```

Figure 1: U:\Desktop\Miestai.png



2 būdas.

```
(%i13) load(simplex)$
```

```
(%i14) C:matrix(
         [0,308,212,285,137,97,330,228],
         [308,0,157,88,231,326,25,109],
         [212,157,0,72,199,175,146,101],
         [285,88,72,0,216,248,78,96],
         [137,231,199,216,0,193,253,125],
         [97,326,175,248,193,0,322,255],
         [330,25,146,78,253,322,0,131],
         [228,109,101,96,125,255,131,0]
        )$
```

```
(%i15) n:length(C);
(%o15) 8
```

```
(%i16) f:sum(sum(C[i,j]*x[i,j],i,1,n),j,1,n)$
```

```
(%i17) s0:makelist(x[i,i]=0,i,1,n)$
```

```
(%i18) s1:create_list(x[i,j]<=1,i,1,n,j,1,n)$
```

```
(%i19) s2:makelist(sum(x[i,j],j,1,n)=1,i,1,n)$
```

```
(%i20) s3:makelist(sum(x[i,j],i,1,n)=1,j,1,n)$
```

```
(%i21) s4:create_list(x[i,j]+x[j,i]<=1, i,1,n,j,1,n)$
```

Uždaro ciklo 1->6->5->1 ir trupmeninių sprendinių uždraudimui įvedame papildomus apribojimus s5.
Pradžioje reikia skaičiuoti be šių apribojimų. Iš gaunamų rezultatų parenkame šiuos papildomus apribojimus:

```
(%i22) s5:[x[1,6]+x[6,5]+x[5,1]<=2,x[1,5]+x[5,6]+x[6,1]<=2,x[1,6]=1]$
```

Sudarome bendrą apribojimų sąrašą apr:

```
(%i23) apr:append(s0,s1,s2,s3,s4,s5)$
```

```
(%i24) length(apr);
(%o24) 155
```

```
(%i25) spr:minimize_lp(f, apr),nonegative_lp=true;
```

$$(\%o25)\ [818,[x_{8,7}=0,x_{8,6}=0,x_{8,5}=1,x_{8,4}=0,x_{8,3}=0,x_{8,2}=0,x_{8,1}=0,x_{7,8}=0,x_{7,6}=0,x_{7,5}=0,x_{7,4}=0,x_{7,3}=0,x_{7,2}=1,x_{7,1}=0,x_{6,8}=0,x_{6,7}=0,x_{6,5}=0,x_{6,4}=0,x_{6,3}=1,x_{6,2}=0,x_{6,1}=0,x_{5,8}=0,x_{5,7}=0,x_{5,6}=0,x_{5,4}=0,x_{5,3}=0,x_{5,2}=0,x_{5,1}=1,x_{4,8}=0,x_{4,7}=1,x_{4,6}=0,x_{4,5}=0,x_{4,3}=0,x_{4,2}=0,x_{4,1}=0,x_{3,8}=0,x_{3,7}=0,x_{3,6}=0,x_{3,5}=0,x_{3,4}=1,x_{3,2}=0,x_{3,1}=0,x_{2,8}=1,x_{2,7}=0,x_{2,6}=0,x_{2,5}=0,x_{2,4}=0,x_{2,3}=0,x_{2,1}=0,x_{1,8}=0,x_{1,7}=0,x_{1,6}=1,x_{1,5}=0,x_{1,4}=0,x_{1,3}=0,x_{1,2}=0,x_{8,8}=0,x_{7,7}=0,x_{6,6}=0,x_{5,5}=0,x_{4,4}=0,x_{3,3}=0,x_{2,2}=0,x_{1,1}=0]]$$

```
(%i26) s:sublist(spr[2],lambda([x],rhs(x)=1));
```

$$(\%o26)\ [x_{8,5}=1,x_{7,2}=1,x_{6,3}=1,x_{5,1}=1,x_{4,7}=1,x_{3,4}=1,x_{2,8}=1,x_{1,6}=1]$$

```
(%i27) tr(x):=[first(lhs(x)),second(lhs(x))]$
```

```
(%i28) L:map(tr,s);
(%o28) [[8,5],[7,2],[6,3],[5,1],[4,7],[3,4],[2,8],[1,6]]
```

Surūšiavimui apibrėžiame funkciją

```
(%i29) F(k):=block([f],if k=1 then (define(f(x),x[1]=1),sublist(L,f)[1])
       else (define(f(x),x[1]=F(k-1)[2]),sublist(L,f)[1]))$
```

Maršrutas:

```
(%i30) makelist(F(k),k,1,8);
(%o30) [[1,6],[6,3],[3,4],[4,7],[7,2],[2,8],[8,5],[5,1]]
```

Maršruto ilgis:

```
(%i31) C[1,6]+C[6,3]+C[3,4]+C[4,7]+C[7,2]+C[2,8]+C[8,5]+C[5,1];
(%o31) 818
```

```
(%i32) subst(spr[2],f);
(%o32) 818
```

Galima pasinaudoti grafų teorijos paketu:

```
(%i55) load (graphs)$
```

```
(%i34) g:create_graph(makelist(k,k,1,8), L)$
```

Maršrutas rastas kitu būdu:

```
(%i35) hc : hamilton_cycle(g);
(%o35) [1,6,3,4,7,2,8,5,1]
```

Vilnius, Klaipeda, Šiauliai, Telšiai, Marijampole, Utena, Palanga, Taurage

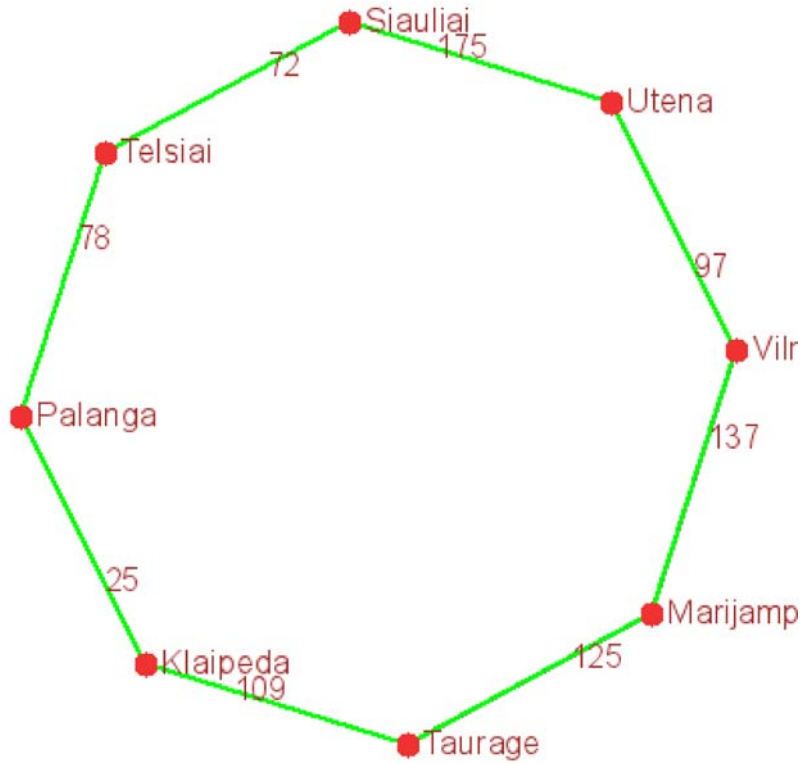
```
(%i36) set_vertex_label(1, "Vilnius", g)$
       set_vertex_label(2, "Klaipeda", g)$
       set_vertex_label(3, "Siauliai", g)$
       set_vertex_label(4, "Telsiai", g)$
       set_vertex_label(5, "Marijampole", g)$
       set_vertex_label(6, "Utena", g)$
       set_vertex_label(7, "Palanga", g)$
       set_vertex_label(8, "Taurage", g)$
```

```
(%i44) L;
(%o44) [[8,5],[7,2],[6,3],[5,1],[4,7],[3,4],[2,8],[1,6]]
```

```
(%i45) set_edge_weight([8,5], C[8,5], g)$
       set_edge_weight([7,2], C[7,2], g)$
       set_edge_weight([6,3], C[6,3], g)$
       set_edge_weight([5,1], C[5,1], g)$
       set_edge_weight([4,7], C[4,7], g)$
       set_edge_weight([3,4], C[3,4], g)$
       set_edge_weight([2,8], C[2,8], g)$
       set_edge_weight([1,6], C[1,6], g)$
```

```
(%i53) draw_graph(
       g,
       show_edges=vertices_to_path(hc),
       show_edge_width=2,
       show_edge_color=green,
       vertex_type=filled_circle,
       vertex_size=2,
       show_label=true,
       label_alignment = 'left,
       show_weight=true,
       text_color=brown
           )$
```

(%t53)



Vilnius, Klaipeda, Šiauliai, Telšiai, Marijampole, Utena, Palanga, Taurage

```
(%i54) hamilton_cycle(g);
(%o54) [1,6,3,4,7,2,8,5,1]
```