```
Cities = Import[
      "https://github.com/hflabs/city/raw/ae661bffe572880472249097c9b29c42b09650ea/city.csv", {"CSV", "Dataset"}, HeaderLines → 1][
    All, {"address", "city", "geo_lat", "geo_lon", "population"}][
   SortBy[-#population &]][
  1 ;; 30] //
 Map[Append[#,
    "city" → If[
      #["city"] == "",
      StringDrop[#["address"], 2],
      #["city"]
     ]
    ] &
 ]
```

address	geo_lat	geo_lon	population	city
г Москва	55.754	37.6204	11 514 330	Москва
г Санкт–Петербург	59.9391	30.3159	4 848 742	Санкт–Петербург
г Новосибирск	55.0282	82.9211	1 498 921	Новосибирск
г Екатеринбург	56.8385	60.6055	1 377 738	Екатеринбург
г Нижний Новгород	56.3241	44.0054	1 250 615	Нижний Новгород
г Казань	55.7944	49.1115	1 216 965	Казань
г Самара	53.195	50.107	1 164 900	Самара
г Омск	54.9849	73.3675	1 154 000	Омск
г Челябинск	55.1603	61.4008	1 130 273	Челябинск
г Ростов–на–Дону	47.2225	39.7188	1 091 544	Ростов–на–Дону
г Уфа	54.7349	55.9578	1 062 300	Уфа
г Волгоград	48.707	44.517	1 021 244	Волгоград
г Пермь	58.0103	56.2342	1 000 679	Пермь
г Красноярск	56.0094	92.8525	973 826	Красноярск
г Воронеж	51.6593	39.1969	889 680	Воронеж
г Саратов	51.5336	46.0343	836 900	Саратов
г Краснодар	45.0402	38.976	744 933	Краснодар
Самарская обл, г Тольятти	53.5205	49.3894	719 484	Тольятти
г Барнаул	53.348	83.7798	635 585	Барнаул
г Ижевск	56.8527	53.2115	628 117	Ижевск

rows 1–20 of 30

```
CityPositions = Normal[GeoPosition[{#["geo_lat"], #["geo_lon"]}] & /@ Cities];
CityNames = Normal[#["city"] & /@ Cities];
CityCount = Length[Cities];

GeoDistance[CityPositions[[1]], CityPositions[[2]]]
```

636.023 km

```
CircuitPairs[p_] := Module[{step},
  step[{d_, old_}, new_] := {Append[d, {old, new}], new};
  Fold[step, {{}, p[[-1]]}, p][[1]]
 ]

CircuitPairs[{1, 2, 3}]
```

{{3, 1}, {1, 2}, {2, 3}}

```
CircuitDistance[p_] := Module[{step},
  step[{d_, old_}, new_] := {d + GeoDistance[CityPositions[[old]], CityPositions[[new]]], new};
  Fold[step, {0, p[[-1]]}, p][[1]]
 ]

UnitConvert[CircuitDistance[{1, 2, 3, 4, 5, 6}], "Kilometers"]
UnitConvert[CircuitDistance[Range[CityCount]], "Kilometers"]
```

7217.34 km

65 368.5 km

```
UpdateCircuit[p_] := Module[{a, b},
  a = RandomInteger[{1, Length[p]}];
  b = RandomInteger[{1, Length[p]}];
  If[a == b,
   UpdateCircuit[p],
   ReplacePart[p, {a → p[[b]], b → p[[a]]}]
  ]
 ]

UpdateCircuit[Range[3]]
```

{2, 1, 3}

```
Options[Annealing] = {"InitialTemperature" → 60 000., "FinishTemperature" → 80, "CoolingFactor" → 0.95};
Annealing[OptionsPattern[]] := Module[{C, Te, prop, step},
  C = OptionValue["CoolingFactor"];
  Te = OptionValue["FinishTemperature"];
  prop[x_, T_] := Exp[-
      QuantityMagnitude[
       UnitConvert[
        CircuitDistance[x],
        "Kilometers"
       ]
      ]/T];
  step[{T_, x_}] := Module[{x1, α, u}, {
     T*C,
     (
      (*Print[T, x];*)
      x1 = UpdateCircuit[x];
      α = prop[x1, T] / prop[x, T];
      u = RandomReal[];
      If[u ≤ α, x1, x]
     )
    }];

  NestWhileList[step, {
    OptionValue["InitialTemperature"],
    RandomSample[Range[CityCount]]
   }, #[[1]] > Te &]
 ]

(*Annealing[InitialTemperature → 200, CoolingFactor → 0.99]*)

AnnealingTime = Timing[Results = <|
      "slow" → Annealing[CoolingFactor → 0.90],
      "middle" → Annealing[CoolingFactor → 0.95],
      "fast" → Annealing[CoolingFactor → 0.99]
      |>][[1]];

StringForm["Finished the simulated annealing in ``s", AnnealingTime]
```

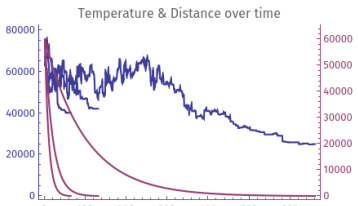
Finished the simulated annealing in 9.456375`s

```
Distances[h_] := CircuitDistance[#[[2]]] & /@ h;

ResultDistances = Distances[#] & /@ Results;

ResultTemperatures = (#[[1]] & /@ #) & /@ Results;

UnitConvert[#[[-1]], "Kilometers"] & /@ ResultDistances
```

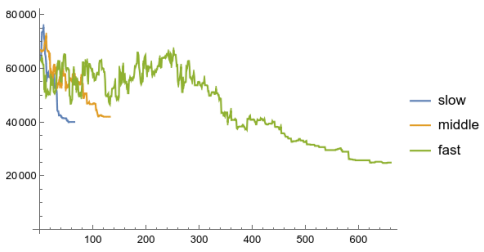
```
⟨|slow → 40 213.4 km, middle → 42 154.4 km, fast → 25 085.6 km|⟩
```

```
(* Taken from https://mathematica.stackexchange.com/questions/627/1-plot-2-scale-axis *)
TwoAxisListPlot[{f_, g_}] := Module[{fgraph, ggraph, frange, grange, fticks, gticks}, {fgraph, ggraph} = MapIndexed[ListPlot[#, Axes → True, PlotStyle → ColorData[1][#2[[1]]] &, {f, g}];
  {frange, grange} = Last[PlotRange /. AbsoluteOptions[#, PlotRange]] & /@ {fgraph, ggraph};
  fticks = Last[Ticks /. AbsoluteOptions[fgraph, Ticks]] /. _RGBColor | _GrayLevel | _Hue :↦ ColorData[1][1];
  gticks = (MapAt[Function[r, Rescale[r, grange, frange]], #, {1}] & /@ Last[Ticks /. AbsoluteOptions[ggraph, Ticks]]) /. _RGBColor | _GrayLevel | _Hue → ColorData[1][2];
  Show[fgraph, ggraph /. Graphics[graph_, s___] :↦ Graphics[GeometricTransformation[graph, RescalingTransform[{{0, 1}, grange}, {{0, 1}, frange}]], s], Axes → False, Frame → True, FrameStyle → {ColorData[1] /@ {1, 2}, {Automatic, Transparent}},
   FrameTicks → {{fticks, gticks}, {Automatic, Automatic}}]]

TwoAxisListLinePlot[{f_, g_}] := Module[{fgraph, ggraph, frange, grange, fticks, gticks}, {fgraph, ggraph} = MapIndexed[ListLinePlot[#, Axes → True, PlotStyle → ColorData[1][#2[[1]]] &, {f, g}];
  {frange, grange} = Last[PlotRange /. AbsoluteOptions[#, PlotRange]] & /@ {fgraph, ggraph};
  fticks = Last[Ticks /. AbsoluteOptions[fgraph, Ticks]] /. _RGBColor | _GrayLevel | _Hue :↦ ColorData[1][1];
  gticks = (MapAt[Function[r, Rescale[r, grange, frange]], #, {1}] & /@ Last[Ticks /. AbsoluteOptions[ggraph, Ticks]]) /. _RGBColor | _GrayLevel | _Hue → ColorData[1][2];
  Show[fgraph, ggraph /. Graphics[graph_, s___] :↦ Graphics[GeometricTransformation[graph, RescalingTransform[{{0, 1}, grange}, {{0, 1}, frange}]], s], Axes → False, Frame → True, FrameStyle → {ColorData[1] /@ {1, 2}, {Automatic, Transparent}},
   FrameTicks → {{fticks, gticks}, {Automatic, Automatic}}]]
```

```
Show[
 TwoAxisListLinePlot[{ResultDistances, ResultTemperatures}],
 PlotLabel → "Temperature & Distance over time",
 LabelStyle → {FontFamily → "Fira Sans"}
]
```




```
ListPlot[ResultDistances, Joined → True]
```




```
PlotState[{T_, p_}] := (
  Overlay[{
    GeoGraphPlot[
     DirectedEdge @@ (CityPositions[[#]] & /@ #) & /@ CircuitPairs[p],
     (*VertexLabels→(Map[CityPositions[[#]]→ CityNames[[#]]&,Range[CityCount]]),*)
     GraphLayout → "Geodesic",
     GeoBackground → "VectorClassic"
    ]
   }]
 )

PlotState[Results["slow"][[1]]]
PlotState[Results["fast"][[-1]]]
```

```
Frames = (PlotState /@ Results["fast"]);
```

```
··· General: Using a limited version of FFmpeg. Install FFmpeg to get more complete codec support.
```

```
··· Export: PlotState cannot be converted to an expression suitable for video export.
```

```
Export["animation.avi", Frames, BitRate → 1 000 000, RasterSize → 1920];
```