```
(*
Instructor：Hiroyuki Akama
 Misc:Sparse Array  and others*)
```

In[1]:=

```
s = Table[Table[Random[Integer, {0, 1}], {20}], {20}]
```

Out[1]=

```
{{0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0},
 {0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1},
 {1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1},
 {0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1},
 {1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0},
 {0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1},
 {0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0},
 {1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0},
 {0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1},
 {0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0},
 {0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1},
 {1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0},
 {1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1},
 {1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1},
 {1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1},
 {0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1},
 {1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1},
 {1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0},
 {1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0}}
```

In[2]:=

```
sas = SparseArray[s]
? SparseArray
```

Out[2]=

```
SparseArray[<203>, {20, 20}]
```

SparseArray[{pos_1 –> val_1, pos_2 –> val_2, …}] yields a sparse array in which values val_i appear at positions pos_i.
SparseArray[{pos_1, pos_2, …} –> {val_1, val_2, …}] yields the same sparse array.
SparseArray[*list*] yields a sparse array version of *list*.
SparseArray[*data*, {d_1, d_2, …}] yields a sparse array representing a $d_1 \times d_2 \times \ldots$ array.
SparseArray[*data*, *dims*, *val*] yields a
sparse array in which unspecified elements are taken to have value *val*. ≫

In[4]:=

```
Normal[sas]
? Normal
```

Out[4]=

```
{{0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0},
 {0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1},
 {1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1},
 {0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1},
 {1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0},
 {0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1},
 {0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0},
 {1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0},
 {0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1},
 {0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0},
 {0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1},
 {1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0},
 {1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1},
 {1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1},
 {1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1},
 {0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1},
 {1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1},
 {1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0},
 {1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0}}
```

Normal[*expr*] converts *expr* to a normal expression, from a variety of special forms. ≫

In[6]:=

```
inversesas = Inverse[sas];
SetPrecision[inversesas, 5]
? SetPrecision
```

Out[7]=

```
{{0.34172, 0.018762, -0.38483, 0.16607, -1.6192, -0.0067864,
  0.56527, 0.096607, 1.1297, 0.052295, 0.37046, 0.025150, 0.15250,
  0.85828, -0.27585, 0.061876, -0.23074, 0.37804, -0.34172, -0.73733},
 {-0.25150, -0.062874, 0.044910, 0.017964, 0.074850, 0.20359, 0.041916,
  0.10180, -0.39222, -0.068862, -0.11377, 0.24551, -0.074850,
  0.25150, -0.22455, 0.14371, -0.077844, -0.34132, 0.25150, 0.11976},
 {0.23034, 0.34092, 0.15649, 0.10259, -0.25030, -0.059481, 0.48383,
  -0.32974, 0.54890, -0.071058, 0.30579, -0.30898, -0.016367, 0.16966,
  -0.18244, -0.045908, -0.022355, -0.21597, -0.23034, -0.22715},
 {0.22555, 0.13972, -0.099800, -0.039920, 0.38922, -0.0079840, 0.017964,
  -0.0039920, 0.093812, -0.29142, 0.14172, -0.32335, -0.055888,
  -0.22555, 0.49900, 0.013972, -0.27146, 0.091816, -0.22555, -0.043912},
 {-0.34451, -0.30279, -0.098004, 0.00079840, 0.99222, 0.12016, -0.42036,
  -0.23992, -0.56188, -0.014172, -0.38283, -0.033533, -0.25888,
  -0.25549, 0.090020, 0.13972, 0.085429, -0.28184, 0.34451, 0.76088},
 {-0.56088, -0.22355, 0.15968, 0.063872, 1.3772, -0.38723, -0.62874, -0.19361,
  -0.95010, -0.13373, -0.62675, 0.31737, 0.28942, -0.43912, 0.20160, 0.17764,
  -0.16567, -0.54691, 0.56088, 0.87026}, {-0.55170, -0.50459, 0.81756, 0.087026,
  1.6515, 0.097405, -0.81916, -0.15130, -1.7445, -0.044711, -0.72894, 0.34491,
  0.28184, -0.84830, 0.31218, 0.22954, 0.31178, -0.72016, 0.55170, 0.93573},
 {1.3238, 0.26427, -0.84591, 0.38164, -1.7210, 0.43633, 1.0683, -0.68184,
  1.4232, 0.22595, 0.0051896, -0.028743, 0.25429, 0.87625, -0.97046, -0.21357,
  -0.16487, 0.28224, -0.32375, -1.3002}, {0.67425, 0.31856, -0.42754, 0.30898,
  -0.51257, 0.50180, 0.32096, -0.84910, 0.053892, 0.015569, -0.15689, 0.022754,
  0.31257, 0.12575, -0.66228, 0.071856, 0.061078, -0.070659, 0.32575, -0.54012},
 {-0.25269, -0.11317, 0.48084, 0.23234, 0.73473, -0.033533, -0.32455,
  0.18323, -0.50599, -0.12395, -0.40479, 0.24192, -0.33473, -0.34731,
  0.19581, -0.34132, -0.14012, -0.014371, 0.25269, 0.41557},
 {0.23234, 0.25808, -0.070060, -0.58802, -0.51677, 0.0023952, 0.094611,
  0.20120, 0.57186, 0.18743, 0.45749, -0.30299, -0.083234, 0.16766,
  -0.049701, 0.095808, 0.081437, 0.57246, -0.23234, -0.38683},
 {-1.2443, -0.26108, 0.92934, -0.26826, 1.6156, -0.37365, -0.75928, 0.61317,
  -1.2096, -0.23832, -0.36766, 0.26707, -0.015569, -1.1557, 0.75329, 0.053892,
  0.29581, -0.30299, 0.24431, 1.3449}, {-0.89102, -0.42275, 0.33054, -0.50778,
  0.95090, -0.42156, -0.65150, 0.58922, -0.64671, 0.013174, 0.48263, 0.32695,
  -0.35090, -0.50898, 0.74731, 0.13772, -0.33293, 0.24790, -0.10898, 1.0814},
 {-0.055888, -0.18064, -0.15669, 0.33733, -0.038922, 0.26747, -0.10180,
  0.13373, -0.14271, 0.26248, -0.24750, -0.16766, 0.37226, 0.055888,
  -0.21657, 0.031936, 0.093812, -0.075848, 0.055888, -0.52894},
 {-0.42116, -0.021956, 0.30140, -0.27944, 0.72455, -0.055888, 0.12575,
  -0.027944, -0.34331, -0.039920, -0.0079840, -0.26347, -0.39122,
  -0.57884, 0.49301, 0.097804, 0.099800, -0.35729, 0.42116, 0.69261},
 {-0.27226, -0.10140, 0.10100, 0.40040, 0.34611, -0.23992, -0.31018,
  0.18004, -0.53094, 0.14291, -0.49142, 0.18323, -0.079441, -0.12774,
  -0.10499, 0.069860, -0.15729, -0.34092, 0.27226, 0.58044},
 {-0.28782, -0.25529, 0.26806, 0.18723, 0.42455, -0.32255, -0.074251, 0.23872,
  -0.50998, -0.073253, -0.27465, 0.13653, 0.042116, 0.087824, -0.14032, -0.23553,
  0.033134, -0.090619, 0.28782, 0.42595}, {0.65749, 0.11437, 0.17545, 0.31018,
  0.22575, 0.18204, 0.19042, -0.70898, -0.038922, -0.25569, -0.23114, -0.027545,
  0.17425, -0.25749, -0.27725, 0.28144, 0.18922, -0.49341, 0.34251, -0.39880},
 {-0.32735, 0.084830, -0.34631, -0.73852, -0.79940, -0.14770, -0.16766, 0.92615,
  0.73553, 0.10878, 1.1218, 0.017964, -0.53393, 0.32735, 0.73154, -0.24152,
  -0.021956, 0.69860, -0.67265, 0.18762}, {1.5465, 0.61996, -0.92854, 0.50858,
  -2.4587, 0.54172, 1.2311, -0.82914, 1.5848, 0.47265, 0.13453, -0.36048,
  0.59202, 1.2535, -1.1573, 0.0019960, 0.41836, 0.47026, -0.54651, -2.3206}}
```

SetPrecision[*expr*, *p*] yields a version of *expr* in which all numbers have been set to have precision *p*. ≫

In[9]:=
```
inversesas.s
inversesas.sas
```

Out[9]=
```
{{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1}}
```

Out[10]=
```
{{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1}}
```

In[11]:=
```
bigmatdata = Table[Table[RandomInteger[], {i, 1, 1000}], {i, 1, 1000}];
```

In[12]:=
```
sparsebigmatdata = SparseArray[bigmatdata]
```

Out[12]=
```
SparseArray[<499 908>, {1000, 1000}]
```

In[13]:=
```
Export["sparsebigmatdata.mtx", sparsebigmatdata, "MTX"]
```

Out[13]=
```
sparsebigmatdata.mtx
```

In[14]:=
```
tmp = Import["sparsebigmatdata.mtx"]
```

Out[14]=
```
SparseArray[<499 908>, {1000, 1000}]
```

In[15]:=
```
sparsebigmatdata == tmp
```

Out[15]=
```
True
```

In[16]:=
```
499487 / (1000 * 1000) // N
```

Out[16]=
```
0.499487
```

In[17]:=
```
pos = Tuples[{Range[10], Range[10]}]
Length[%]
? Tuples
```

Out[17]=
```
{{1, 1}, {1, 2}, {1, 3}, {1, 4}, {1, 5}, {1, 6}, {1, 7}, {1, 8}, {1, 9}, {1, 10}, {2, 1}, {2, 2},
 {2, 3}, {2, 4}, {2, 5}, {2, 6}, {2, 7}, {2, 8}, {2, 9}, {2, 10}, {3, 1}, {3, 2}, {3, 3}, {3, 4},
 {3, 5}, {3, 6}, {3, 7}, {3, 8}, {3, 9}, {3, 10}, {4, 1}, {4, 2}, {4, 3}, {4, 4}, {4, 5},
 {4, 6}, {4, 7}, {4, 8}, {4, 9}, {4, 10}, {5, 1}, {5, 2}, {5, 3}, {5, 4}, {5, 5}, {5, 6},
 {5, 7}, {5, 8}, {5, 9}, {5, 10}, {6, 1}, {6, 2}, {6, 3}, {6, 4}, {6, 5}, {6, 6}, {6, 7},
 {6, 8}, {6, 9}, {6, 10}, {7, 1}, {7, 2}, {7, 3}, {7, 4}, {7, 5}, {7, 6}, {7, 7}, {7, 8},
 {7, 9}, {7, 10}, {8, 1}, {8, 2}, {8, 3}, {8, 4}, {8, 5}, {8, 6}, {8, 7}, {8, 8}, {8, 9},
 {8, 10}, {9, 1}, {9, 2}, {9, 3}, {9, 4}, {9, 5}, {9, 6}, {9, 7}, {9, 8}, {9, 9}, {9, 10},
 {10, 1}, {10, 2}, {10, 3}, {10, 4}, {10, 5}, {10, 6}, {10, 7}, {10, 8}, {10, 9}, {10, 10}}
```

Out[18]=
```
100
```

Tuples[*list*, *n*] generates a list of all possible *n*-tuples of elements from *list*.
Tuples[{$list_1$, $list_2$, …}] generates a list of all possible tuples whose i^{th} element is from $list_i$. ≫

In[20]:=
```
val = Table[Random[BinomialDistribution[1, 0.1]], {i, 1, 100}]
? BinomialDistribution
```

Out[20]=
```
{0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0}
```

BinomialDistribution[*n*, *p*] represents a binomial distribution with *n* trials and success probability *p*. ≫

In[22]:=
```
arrayrule = MapThread[#1 → #2 &, {pos, val}]
```

Out[22]=
```
{{1, 1} → 0, {1, 2} → 0, {1, 3} → 0, {1, 4} → 0, {1, 5} → 0, {1, 6} → 0, {1, 7} → 0, {1, 8} → 0,
 {1, 9} → 1, {1, 10} → 0, {2, 1} → 0, {2, 2} → 1, {2, 3} → 0, {2, 4} → 0, {2, 5} → 0, {2, 6} → 1,
 {2, 7} → 0, {2, 8} → 0, {2, 9} → 0, {2, 10} → 0, {3, 1} → 0, {3, 2} → 1, {3, 3} → 0, {3, 4} → 0,
 {3, 5} → 1, {3, 6} → 0, {3, 7} → 0, {3, 8} → 0, {3, 9} → 0, {3, 10} → 0, {4, 1} → 0, {4, 2} → 0,
 {4, 3} → 0, {4, 4} → 0, {4, 5} → 0, {4, 6} → 0, {4, 7} → 0, {4, 8} → 0, {4, 9} → 0, {4, 10} → 0,
 {5, 1} → 0, {5, 2} → 0, {5, 3} → 0, {5, 4} → 0, {5, 5} → 0, {5, 6} → 0, {5, 7} → 0, {5, 8} → 0,
 {5, 9} → 0, {5, 10} → 0, {6, 1} → 0, {6, 2} → 0, {6, 3} → 0, {6, 4} → 0, {6, 5} → 0, {6, 6} → 0,
 {6, 7} → 0, {6, 8} → 0, {6, 9} → 0, {6, 10} → 0, {7, 1} → 0, {7, 2} → 0, {7, 3} → 1, {7, 4} → 0,
 {7, 5} → 0, {7, 6} → 0, {7, 7} → 0, {7, 8} → 0, {7, 9} → 0, {7, 10} → 0, {8, 1} → 0,
 {8, 2} → 0, {8, 3} → 0, {8, 4} → 0, {8, 5} → 0, {8, 6} → 0, {8, 7} → 0, {8, 8} → 1, {8, 9} → 0,
 {8, 10} → 0, {9, 1} → 0, {9, 2} → 0, {9, 3} → 0, {9, 4} → 0, {9, 5} → 0, {9, 6} → 0,
 {9, 7} → 0, {9, 8} → 0, {9, 9} → 0, {9, 10} → 0, {10, 1} → 0, {10, 2} → 0, {10, 3} → 0,
 {10, 4} → 0, {10, 5} → 0, {10, 6} → 0, {10, 7} → 1, {10, 8} → 0, {10, 9} → 0, {10, 10} → 0}
```

In[23]:=
```
sparseranddata1 = SparseArray[arrayrule, {10, 10}]
```

Out[23]=
```
SparseArray[<8>, {10, 10}]
```

In[24]:=
```
Normal[sparseranddata1] // MatrixForm
```

Out[24]//MatrixForm=

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

In[25]:=
```
Export["sparseranddata1.mtx", sparseranddata1, "MTX"]
```

Out[25]= sparseranddata1.mtx

In[26]:=
```
tmp1 = Import["sparseranddata1.mtx"]
```

Out[26]= SparseArray[<8>, {10, 10}]

In[27]:=
```
megasparsedata1 = SparseArray[MapThread[#1 → #2 &, {Tuples[{Range[100], Range[100]}],
    Table[Random[BinomialDistribution[1, 0.001]], {i, 1, 10 000}]}], {100, 100}]
```

Out[27]= SparseArray[<9>, {100, 100}]

In[28]:=
```
ByteCount[megasparsedata1]
? ByteCount
```

Out[28]= 1000

ByteCount[*expr*] gives the number of bytes used internally by *Mathematica* to store *expr*. ≫

In[30]:=
```
ByteCount[Normal[megasparsedata1]]
```

Out[30]= 40 088

In[31]:=
```
megasparsedata2 = SparseArray[MapThread[#1 → #2 &, {Tuples[{Range[100], Range[100]}],
    Table[Random[BinomialDistribution[1, 0.001]], {i, 1, 10 000}]}], {100, 100}]
? BinomialDistribution
```

Out[31]= SparseArray[<8>, {100, 100}]

BinomialDistribution[*n*, *p*] represents a binomial distribution with *n* trials and success probability *p*. ≫

In[33]:=
```
megasparsedata1 + megasparsedata2
```

Out[33]= SparseArray[<17>, {100, 100}]

In[34]:=
```
megasparsedata1 * megasparsedata2
```

Out[34]= SparseArray[<0>, {100, 100}]

In[35]:=
```
Join[megasparsedata1, megasparsedata2]
```

Out[35]= SparseArray[<17>, {200, 100}]

In[36]:=
```
ReplacePart[megasparsedata1, {{1, 1} → 1}]
? ReplacePart
```

Out[36]=
```
SparseArray[<10>, {100, 100}]
```

ReplacePart[*expr*, *i* –> *new*] yields an expression in which the ith part of *expr* is replaced by *new*.
ReplacePart[*expr*, {i_1 –> new_1, i_2 –> new_2, …}] replaces parts at positions i_n by new_n.
ReplacePart[*expr*, {*i*, *j*, …} –> *new*] replaces the part at position {*i*, *j*, …}.
ReplacePart[*expr*, {{i_1, j_1, …} –> new_1, …}] replaces parts at positions {i_n, j_n, …} by new_n.
ReplacePart[*expr*, {{i_1, j_1, …}, …} –> *new*] replaces all parts at positions {i_n, j_n, …} by *new*. ≫

In[38]:=
```
Length[megasparsedata2]
```

Out[38]=
```
100
```

In[39]:=
```
Transpose[megasparsedata2]
```

Out[39]=
```
SparseArray[<8>, {100, 100}]
```

In[40]:=
```
Position[megasparsedata2, 1]
Position[Normal[megasparsedata2], 1]
? Position
```

Out[40]=
```
{}
```

Out[41]=
```
{{36, 15}, {36, 39}, {48, 62}, {61, 32}, {67, 34}, {72, 62}, {79, 44}, {85, 69}}
```

Position[*expr*, *pattern*] gives a list of the positions at which objects matching *pattern* appear in *expr*.
Position[*expr*, *pattern*, *levelspec*] finds only objects that appear on levels specified by *levelspec*.
Position[*expr*, *pattern*, *levelspec*, *n*] gives the positions of the first *n* objects found. ≫

In[43]:=
```
ar = ArrayRules[megasparsedata2]
? ArrayRules
```

Out[43]=
```
{{36, 15} → 1, {36, 39} → 1, {48, 62} → 1, {61, 32} → 1,
 {67, 34} → 1, {72, 62} → 1, {79, 44} → 1, {85, 69} → 1, {_, _} → 0}
```

ArrayRules[SparseArray[…]] gives the rules {pos_1 –> val_1, pos_2 –> val_2, …} specifying elements in a sparse array.
ArrayRules[*list*] gives rules for SparseArray[*list*]. ≫

In[45]:=
```
Most[ar]
#[[1]] & /@ Most[ar]
? Most
```

Out[45]=
```
{{36, 15} → 1, {36, 39} → 1, {48, 62} → 1,
 {61, 32} → 1, {67, 34} → 1, {72, 62} → 1, {79, 44} → 1, {85, 69} → 1}
```

Out[46]=
```
{{36, 15}, {36, 39}, {48, 62}, {61, 32}, {67, 34}, {72, 62}, {79, 44}, {85, 69}}
```

Most[*expr*] gives *expr* with the last element removed. ≫

In[48]:=
```
megasparsedata3 = SparseArray[MapThread[#1 → #2 &, {Tuples[{Range[100], Range[100]}],
    Table[Round[Random[NormalDistribution[0, 0.2]]], {i, 1, 10 000}]}], {100, 100}]
```

Out[48]=
```
SparseArray[<107>, {100, 100}]
```

In[49]:=
```
ar3 = ArrayRules[megasparsedata3]
```

Out[49]=
```
{{1, 99} → -1, {2, 42} → 1, {3, 42} → -1, {4, 6} → 1, {4, 55} → -1, {6, 65} → 1,
 {8, 19} → -1, {9, 13} → -1, {11, 30} → 1, {11, 50} → -1, {14, 35} → -1, {16, 37} → -1,
 {16, 52} → 1, {18, 8} → 1, {19, 1} → -1, {21, 66} → -1, {22, 47} → 1, {23, 5} → -1,
 {23, 98} → 1, {24, 32} → -1, {24, 35} → -1, {24, 63} → -1, {25, 56} → 1, {26, 6} → -1,
 {26, 25} → -1, {27, 73} → 1, {28, 4} → -1, {28, 20} → 1, {28, 33} → 1, {30, 34} → -1,
 {30, 56} → -1, {30, 85} → 1, {33, 10} → 1, {33, 46} → -1, {33, 49} → -1, {35, 83} → -1,
 {35, 84} → -1, {36, 26} → -1, {36, 57} → -1, {36, 63} → 1, {37, 68} → 1, {38, 6} → -1,
 {38, 40} → 1, {38, 72} → -1, {38, 80} → 1, {43, 48} → -1, {44, 86} → 1, {46, 77} → 1,
 {48, 28} → 1, {48, 70} → -1, {48, 85} → 1, {50, 14} → -1, {51, 26} → 1, {54, 46} → -1,
 {55, 91} → -1, {56, 18} → 1, {57, 81} → -1, {58, 56} → -1, {59, 79} → -1, {61, 70} → -1,
 {63, 34} → 1, {65, 31} → 1, {65, 38} → -1, {65, 56} → -1, {66, 37} → -1, {66, 96} → -1,
 {67, 16} → -1, {67, 89} → -1, {68, 52} → 1, {70, 7} → 1, {71, 82} → 1, {71, 84} → 1,
 {74, 32} → 1, {74, 48} → 1, {75, 44} → 1, {77, 6} → 1, {78, 80} → -1, {79, 4} → 1,
 {79, 76} → 1, {80, 23} → 1, {81, 7} → 1, {81, 47} → -1, {86, 50} → -1, {86, 85} → -1,
 {87, 13} → -1, {87, 96} → 1, {88, 34} → 1, {88, 49} → -1, {88, 54} → -1, {89, 1} → 1,
 {90, 35} → -1, {90, 83} → -1, {92, 88} → 1, {93, 83} → -1, {93, 85} → -1, {94, 98} → 1,
 {95, 94} → -1, {96, 61} → -1, {97, 39} → -1, {98, 22} → -1, {98, 52} → 1, {98, 61} → 1,
 {98, 80} → -1, {98, 91} → 1, {99, 9} → -1, {99, 41} → 1, {99, 47} → -1, {_, _} → 0}
```

In[50]:=
```
Position[#[[2]] & /@ ar3, 1]
% // Flatten
Part[#[[1]] & /@ ar3, %]
```

Out[50]=
```
{{2}, {4}, {6}, {9}, {13}, {14}, {17}, {19}, {23}, {26}, {28}, {29}, {32}, {33}, {40}, {41}, {43},
 {45}, {47}, {48}, {49}, {51}, {53}, {56}, {61}, {62}, {69}, {70}, {71}, {72}, {73}, {74},
 {75}, {76}, {78}, {79}, {80}, {81}, {86}, {87}, {90}, {93}, {96}, {101}, {102}, {104}, {106}}
```

Out[51]=
```
{2, 4, 6, 9, 13, 14, 17, 19, 23, 26, 28, 29, 32, 33, 40, 41, 43, 45, 47, 48, 49, 51, 53, 56, 61,
 62, 69, 70, 71, 72, 73, 74, 75, 76, 78, 79, 80, 81, 86, 87, 90, 93, 96, 101, 102, 104, 106}
```

Out[52]=
```
{{2, 42}, {4, 6}, {6, 65}, {11, 30}, {16, 52}, {18, 8}, {22, 47}, {23, 98}, {25, 56}, {27, 73},
 {28, 20}, {28, 33}, {30, 85}, {33, 10}, {36, 63}, {37, 68}, {38, 40}, {38, 80}, {44, 86},
 {46, 77}, {48, 28}, {48, 85}, {51, 26}, {56, 18}, {63, 34}, {65, 31}, {68, 52}, {70, 7},
 {71, 82}, {71, 84}, {74, 32}, {74, 48}, {75, 44}, {77, 6}, {79, 4}, {79, 76}, {80, 23}, {81, 7},
 {87, 96}, {88, 34}, {89, 1}, {92, 88}, {94, 98}, {98, 52}, {98, 61}, {98, 91}, {99, 41}}
```

In[53]:=
```
SparsePart[sparsearray_, value_] :=
  Module[{arrayrules, positions}, arrayrules = ArrayRules[sparsearray]; positions =
    Position[#[[2]] & /@ arrayrules, value] // Flatten; Part[#[[1]] & /@ arrayrules, positions]];
```

General::spell1 :
New symbol name "arrayrules" is similar to existing symbol "arrayrule" and may be misspelled. ≫

In[54]:=
```
SparsePart[megasparsedata3, -1]
```

Out[54]=
```
{{1, 99}, {3, 42}, {4, 55}, {8, 19}, {9, 13}, {11, 50}, {14, 35}, {16, 37}, {19, 1},
 {21, 66}, {23, 5}, {24, 32}, {24, 35}, {24, 63}, {26, 6}, {26, 25}, {28, 4}, {30, 34},
 {30, 56}, {33, 46}, {33, 49}, {35, 83}, {35, 84}, {36, 26}, {36, 57}, {38, 6}, {38, 72},
 {43, 48}, {48, 70}, {50, 14}, {54, 46}, {55, 91}, {57, 81}, {58, 56}, {59, 79},
 {61, 70}, {65, 38}, {65, 56}, {66, 37}, {66, 96}, {67, 16}, {67, 89}, {78, 80},
 {81, 47}, {86, 50}, {86, 85}, {87, 13}, {88, 49}, {88, 54}, {90, 35}, {90, 83},
 {93, 83}, {93, 85}, {95, 94}, {96, 61}, {97, 39}, {98, 22}, {98, 80}, {99, 9}, {99, 47}}
```

In[55]:=
```
<< Combinatorica`

MySparseRandomGraph[n_, p_] := Module[{d = BinomialDistribution[Binomial[n, 2], p]},
   pairs = Flatten[Map[{NthPair[#]} &, RandomKSubset[Range[Binomial[n, 2]], Random[d]]], 1];
   SparseArray[ToAdjacencyMatrix[FromUnorderedPairs[pairs]]]];
```

In[57]:=
```
Plus[1, 2, 3, 4]
Apply[Plus, {1, 2, 3, 4}]
Plus @@ {1, 2, 3, 4}
? Apply
```

Out[57]= 10

Out[58]= 10

Out[59]= 10

Apply[*f*, *expr*] or *f* @@ *expr* replaces the head of *expr* by *f*.
Apply[*f*, *expr*, *levelspec*] replaces heads in parts of *expr* specified by *levelspec*. ≫

In[61]:=
```
? List
List[a + b + c + d]
Apply[List, a + b + c + d]
List[Plus[a, b, c, d]]
Apply[List, Plus[a, b, c, d]]
List @@ {a + b + c + d}
List @@@ {a + b + c + d}
```

{e_1, e_2, …} is a list of elements. ≫

Out[62]= {a + b + c + d}

Out[63]= {a, b, c, d}

Out[64]= {a + b + c + d}

Out[65]= {a, b, c, d}

Out[66]= {a + b + c + d}

Out[67]= {{a, b, c, d}}

In[68]:=
```
Plus[{1, {2}, {{3}}}]
Apply[Plus, {1, {2}, {{3}}}]
```

Out[68]= {1, {2}, {{3}}}

Out[69]= {{6}}

In[70]:=
```
? Map
? Head
```

Map[*f*, *expr*] or *f* /@ *expr* applies *f* to each element on the first level in *expr*.
Map[*f*, *expr*, *levelspec*] applies *f* to parts of *expr* specified by *levelspec*. ≫

Head[*expr*] gives the head of *expr*. ≫

In[72]:=
```
Map[Head, {1, 0.2, "Tokyo", {2}, Function}]
```

Out[72]= {Integer, Real, String, List, Symbol}

In[73]:=
```
Head[#] & /@ {1, 0.2, "Tokyo", {2}, Function}
Function[x, Map[Head, x]][{1, 0.2, "Tokyo", {2}, Function}]
```

Out[73]= {Integer, Real, String, List, Symbol}

Out[74]= {Integer, Real, String, List, Symbol}

In[75]:=
```
? Thread
```

Thread[*f*[*args*]] "threads" *f* over any lists that appear in *args*.
Thread[*f*[*args*], *h*] threads *f* over any objects with head *h* that appear in *args*.
Thread[*f*[*args*], *h*, *n*] threads *f* over objects with head *h* that appear in the first *n* *args*. ≫

In[76]:=
```
Thread[List[{1, 2, 3}, {4, 5, 6}]]
Thread[{a, b, c} < {d, e, f}]
{1, 2, 3} < {4, 5, 6}
Thread[%]
```

Out[76]= {{1, 4}, {2, 5}, {3, 6}}

Out[77]= {a < d, b < e, c < f}

Out[78]= {1, 2, 3} < {4, 5, 6}

Out[79]= {True, True, True}

In[80]:=
```
? MapThread
```

MapThread[*f*, {{a_1, a_2, …}, {b_1, b_2, …}, …}] gives {*f*[a_1, b_1, …], *f*[a_2, b_2, …], …}.
MapThread[*f*, {$expr_1$, $expr_2$, …}, *n*] applies *f* to the parts of the $expr_i$ at level *n*. ≫

In[81]:=
```
MapThread[List, {{1, 2, 3}, {4, 5, 6}}]
MapThread[Function[x, List][x, y], {{1, 2, 3}, {4, 5, 6}}]
MapThread[List[#1, #2] &, {{1, 2, 3}, {4, 5, 6}}]
```

Out[81]= {{1, 4}, {2, 5}, {3, 6}}

Out[82]= {{1, 4}, {2, 5}, {3, 6}}

Out[83]= {{1, 4}, {2, 5}, {3, 6}}

In[84]:=
```
MapThread[Divide, {{1, 2, 3}, {4, 5, 6}}]
```

Out[84]= $\left\{\frac{1}{4}, \frac{2}{5}, \frac{1}{2}\right\}$

In[85]:=
```
MapThread[Insert[#1, #2, -1] &, {{{1, 2}, {3, 4}, {5, 6}}, {7, 8, 9}}]
MapThread[Insert, {{{1, 2}, {3, 4}, {5, 6}}, {7, 8, 9}}]
```

Out[85]= {{1, 2, 7}, {3, 4, 8}, {5, 6, 9}}

Insert::argrx : Insert called with 2 arguments; 3 arguments are expected. ≫

Insert::argrx : Insert called with 2 arguments; 3 arguments are expected. ≫

Insert::argrx : Insert called with 2 arguments; 3 arguments are expected. ≫

General::stop : Further output of Insert::argrx will be suppressed during this calculation. ≫

Out[86]= {Insert[{1, 2}, 7], Insert[{3, 4}, 8], Insert[{5, 6}, 9]}

In[87]:=
```
? Nest
```

Nest[*f*, *expr*, *n*] gives an expression with *f* applied *n* times to *expr*. ≫

In[88]:=
```
Nest[f, a, 4]
Nest[Sqrt, a, 4]
```

Out[88]= f[f[f[f[a]]]]

Out[89]= $a^{1/16}$

In[90]:=
```
? NestList
```

NestList[*f*, *expr*, *n*] gives a list of the results of applying *f* to *expr* 0 through *n* times. ≫

In[91]:=
```
NestList[f, a, 4]
NestList[Sqrt, a, 4]
```

Out[91]= {a, f[a], f[f[a]], f[f[f[a]]], f[f[f[f[a]]]]}

Out[92]= $\left\{a, \sqrt{a}, a^{1/4}, a^{1/8}, a^{1/16}\right\}$

In[93]:=
```
? Fold
? FoldList
```

Fold[*f*, *x*, *list*] gives the last element of FoldList[*f*, *x*, *list*]. ≫

FoldList[*f*, *x*, {*a*, *b*, …}] gives {*x*, *f*[*x*, *a*], *f*[*f*[*x*, *a*], *b*], …}. ≫

In[95]:=
```
Fold[#1^#2 &, x, {a, b, c, d}]
FoldList[#1^#2 &, x, {a, b, c, d}]
```

Out[95]= $\left(\left(\left(x^a\right)^b\right)^c\right)^d$

Out[96]= $\left\{x, x^a, \left(x^a\right)^b, \left(\left(x^a\right)^b\right)^c, \left(\left(\left(x^a\right)^b\right)^c\right)^d\right\}$

In[98]:=
```
NestList[(1 / 2)^# &, 1, 20]
% // N
```

Out[98]=
$$\left\{1, \frac{1}{2}, \frac{1}{\sqrt{2}}, 2^{-\frac{1}{\sqrt{2}}}, 2^{-2^{-\frac{1}{\sqrt{2}}}}, 2^{-2^{-2^{-\frac{1}{\sqrt{2}}}}}, 2^{-2^{-2^{-2^{-\frac{1}{\sqrt{2}}}}}}, 2^{-2^{-2^{-2^{-2^{-\frac{1}{\sqrt{2}}}}}}}, 2^{-2^{-2^{-2^{-2^{-2^{-\frac{1}{\sqrt{2}}}}}}}}, 2^{-2^{-2^{-2^{-2^{-2^{-2^{-\frac{1}{\sqrt{2}}}}}}}}}, 2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-\frac{1}{\sqrt{2}}}}}}}}}}, 2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-\frac{1}{\sqrt{2}}}}}}}}}}}, 2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-\frac{1}{\sqrt{2}}}}}}}}}}}}, 2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-\frac{1}{\sqrt{2}}}}}}}}}}}}}, 2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-\frac{1}{\sqrt{2}}}}}}}}}}}}}}, 2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-\frac{1}{\sqrt{2}}}}}}}}}}}}}}}, 2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-\frac{1}{\sqrt{2}}}}}}}}}}}}}}}}, 2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-\frac{1}{\sqrt{2}}}}}}}}}}}}}}}}}, 2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-\frac{1}{\sqrt{2}}}}}}}}}}}}}}}}}}, 2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-\frac{1}{\sqrt{2}}}}}}}}}}}}}}}}}}}, 2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-\frac{1}{\sqrt{2}}}}}}}}}}}}}}}}}}}}, 2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-\frac{1}{\sqrt{2}}}}}}}}}}}}}}}}}}}}}, 2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-\frac{1}{\sqrt{2}}}}}}}}}}}}}}}}}}}}}}, 2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-\frac{1}{\sqrt{2}}}}}}}}}}}}}}}}}}}}}}}, 2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-\frac{1}{\sqrt{2}}}}}}}}}}}}}}}}}}}}}}}}, 2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-\frac{1}{\sqrt{2}}}}}}}}}}}}}}}}}}}}}}}}}\right\}$$

Out[99]=
```
{1., 0.5, 0.707107, 0.612547, 0.654041, 0.635498, 0.643719,
 0.640061, 0.641686, 0.640964, 0.641285, 0.641142, 0.641205, 0.641177,
 0.64119, 0.641184, 0.641187, 0.641185, 0.641186, 0.641186, 0.641186}
```

In[100]:=
```
? FixedPoint
FixedPoint[(1 / 2)^# &, (1 / 2), 30]
% // N
```

FixedPoint[*f*, *expr*] starts with *expr*, then applies *f* repeatedly until the result no longer changes. ≫

Out[101]=
$$2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-2^{-

In[106]:=
```
NestList[(1 / 2)^# &, 2, 10] // N
NestWhile[N[(1 / 2)^# &], 2, Abs[#2 - #1] > 0.001 &, 2]
```

Out[106]= {2., 0.25, 0.840896, 0.558297, 0.679104, 0.624553, 0.648621, 0.63789, 0.642652, 0.640534, 0.641475}

Out[107]= 0.641475

In[108]:=
```
? NestWhileList
NestWhileList[N[(1 / 2)^# &], 2, Abs[#2 - #1] > 0.001 &, 2]
```

NestWhileList[*f*, *expr*, *test*] generates a list of the results of applying *f* repeatedly, starting with *expr*, and continuing until applying *test* to the result no longer yields True.
NestWhileList[*f*, *expr*, *test*, *m*] supplies the most recent *m* results as arguments for *test* at each step.
NestWhileList[*f*, *expr*, *test*, All] supplies all results so far as arguments for *test* at each step.
NestWhileList[*f*, *expr*, *test*, *m*, *max*] applies *f* at most *max* times. ≫

Out[109]= {2, 0.25, 0.840896, 0.558297, 0.679104, 0.624553, 0.648621, 0.63789, 0.642652, 0.640534, 0.641475}

In[110]:=
```
tmp1 = NestList[N[(1 / 2)^# &], 2, 30]
```

Out[110]= {2, 0.25, 0.840896, 0.558297, 0.679104, 0.624553, 0.648621, 0.63789, 0.642652, 0.640534, 0.641475, 0.641057, 0.641243, 0.64116, 0.641197, 0.641181, 0.641188, 0.641185, 0.641186, 0.641186, 0.641186, 0.641186, 0.641186, 0.641186, 0.641186, 0.641186, 0.641186, 0.641186, 0.641186, 0.641186}

In[111]:=
```
pairings = Select[Tuples[{Range[30], Range[30]}], #[[2]] - #[[1]] == 1 &]
```

Out[111]= {{1, 2}, {2, 3}, {3, 4}, {4, 5}, {5, 6}, {6, 7}, {7, 8}, {8, 9}, {9, 10}, {10, 11}, {11, 12}, {12, 13}, {13, 14}, {14, 15}, {15, 16}, {16, 17}, {17, 18}, {18, 19}, {19, 20}, {20, 21}, {21, 22}, {22, 23}, {23, 24}, {24, 25}, {25, 26}, {26, 27}, {27, 28}, {28, 29}, {29, 30}}

In[112]:=
```
diff = Abs[Part[tmp1, #[[1]]] - Part[tmp1, #[[2]]] & /@ pairings]
```

Out[112]= {1.75, 0.590896, 0.2826, 0.120807, 0.0545503, 0.0240674, 0.0107307, 0.00476228, 0.00211787, 0.000940992, 0.000418263, 0.000185881, 0.0000826143, 0.0000367164, 0.0000163182, 7.25236×10^{-6}, 3.22321×10^{-6}, 1.43251×10^{-6}, 6.3666×10^{-7}, 2.82955×10^{-7}, 1.25755×10^{-7}, 5.58902×10^{-8}, 2.48396×10^{-8}, 1.10396×10^{-8}, 4.90641×10^{-9}, 2.18058×10^{-9}, 9.6913×10^{-10}, 4.30717×10^{-10}, 1.91426×10^{-10}}

In[113]:=
```
len = Select[diff, # < 0.001 &] // Length
```

Out[113]= 20

In[114]:=
```
Take[tmp1, Length[tmp1] - len]
```

Out[114]= {2, 0.25, 0.840896, 0.558297, 0.679104, 0.624553, 0.648621, 0.63789, 0.642652, 0.640534, 0.641475}