D:\tmp\gs_tutorial_code\ralf_gk_python_soln_gms_tutorial\sort_merge.py

```python
#
# cctbx sort-merge solution for Siena exercise given by George Sheldrick
#
# sort_merge_initial.py was written in exactly 30 minutes while
# sitting in the audience as others explained their solutions.
#
# sort_merge.py is a slight enhancement (use diff for details).
# It doesn't solve the exercise exactly, but demonstrates how to
# work with the high-level cctbx facilities to solve most of the
# exercise. Note that sort_merge.py produces significantly more
# information than was requested, e.g. the space group name,
# data completeness, etc.
#

from cctbx.array_family import flex
from cctbx import crystal
from cctbx import uctbx
from cctbx import sgtbx
from cctbx import miller
import sys

def run(args):
  assert len(args) == 1
  lines = open(args[0]).read().splitlines()
  title = lines[0]
  unit_cell = uctbx.unit_cell(lines[1])
  n_symops = int(lines[2].split()[0])
  space_group = sgtbx.space_group()
  for line in lines[3:3+n_symops]:
    coeffs = [float(field) for field in line.split()]
    space_group.expand_smx(sgtbx.rt_mx(coeffs[:9], coeffs[9:]))
  crystal_symmetry = crystal.symmetry(
    unit_cell=unit_cell,
    space_group=space_group)
  miller_indices = flex.miller_index()
  data = flex.double()
  sigmas = flex.double()
  for i_line in xrange(3+n_symops,len(lines)):
    fields = lines[i_line].split()
    assert len(fields) == 5
    miller_indices.append([int(value) for value in fields[:3]])
    data.append(float(fields[3]))
    sigmas.append(float(fields[4]))
  miller_set=miller.set(
    crystal_symmetry=crystal_symmetry,
    indices=miller_indices,
    anomalous_flag=False)
  miller_array = miller_set.array(
    data=data,
    sigmas=sigmas).set_observation_type_xray_intensity()
  print "Before merging:"
  miller_array.show_summary()
  print
  merged = miller_array.merge_equivalents()
  merged.show_summary()
  print
  merged_array = merged.array()
  print "After merging:"
  merged_array.show_comprehensive_summary()
  print

if (__name__ == "__main__"):
  run(sys.argv[1:])
```