1. Previous Questions

**Q1: Do we still need to calculate the coefficients?**

A1: No. Not until we start to consider the interaction between different species.

**Q2: Is insertion of new atoms to the specific site of the previous atom necessary?**

A2: I don't think it's necessary because I've tried not to do that and the result is fine. Poscar will make adjustment during calculation. (We forgot to ask Shyam about it, next time we can make sure.)

**Q3: How to submit multiple jobs?**

A3: import glob from python lib.

Zini: I will figure it out by myself.

**Q4: About creating vacancies.**

Zini's way:

I will divide this job into the following steps:

**Step1:** find the smallest supercell that satisfy the required vacancy concentration, for example, if we want a 25% vacancy concentration, the least Li atoms we need is 4, so that we just need to remove 1 atom.

**Step2:** Get the indices of the target specie, e.g.:'Zn'

*In [17]: structure.indices\_from\_symbol("Zn")*

*Out[17]: (0, 1, 2, 3)*

**Step3:** Use spacegroup analyzer to find the equivalent positions:

In [18]: from pymatgen.analysis.structure\_analyzer import SpacegroupAnalyzer

In [19]: sga = SpacegroupAnalyzer(structure)

In [20]: sym\_struc = sga.get\_symmetrized\_structure()

In [21]: sym\_struc.equivalent\_indices

Out[22]: [[0, 1, 2, 3], [4, 5, 6, 7]]

from out[22] we can pick the indices of species we want and do the removal.

**Step4:** Construct all possible structures of target vacancy concentration, but without swapping. Relax them and find out the original structure with lowest energy.

**Step5:** We can then start swapping. ;)

**2. Analyze the outcome**

Go to the target folder

*In [2]: from pymatgen.io.vasp.outputs import Vasprun*

*In [3]: v = Vasprun("vasprun.xml")*

*In [5]: v.converged*

*Out[5]: True*

*In [6]: v.final\_energy*

*Out[6]: -28.0732105*

*In [7]: v.final\_structure*

*Structure Summary*

*Lattice*

*abc : 5.45027 5.45027 5.45027*

*angles : 90.0 90.0 90.0*

*volume : 161.90268521693466*

*A : 5.45027 0.0 0.0*

*B : 0.0 5.45027 0.0*

*C : 0.0 0.0 5.45027*

*PeriodicSite: Zn (0.0000, 0.0000, 0.0000) [0.0000, 0.0000, 0.0000]*

*PeriodicSite: Zn (0.0000, 2.7251, 2.7251) [0.0000, 0.5000, 0.5000]*

*PeriodicSite: Zn (2.7251, 0.0000, 2.7251) [0.5000, 0.0000, 0.5000]*

*......*

*In [8]: v.initial\_structure*

*Out[8]:*

*Structure Summary*

*Lattice*

*abc : 5.45027 5.45027 5.45027*

*angles : 90.0 90.0 90.0*

*volume : 161.90268521693466*

*A : 5.45027 0.0 0.0*

*B : 0.0 5.45027 0.0*

*C : 0.0 0.0 5.45027*

*PeriodicSite: Zn (0.0000, 0.0000, 0.0000) [0.0000, 0.0000, 0.0000]*

*PeriodicSite: Zn (0.0000, 2.7251, 2.7251) [0.0000, 0.5000, 0.5000]*

*PeriodicSite: Zn (2.7251, 0.0000, 2.7251) [0.5000, 0.0000, 0.5000]*

*PeriodicSite: Zn (2.7251, 2.7251, 0.0000) [0.5000, 0.5000, 0.0000]*

[in this case they are the same...]

*In [9]: pretend\_made\_structure = v.initial\_structure.copy()*

*In [11]: from pymatgen.analysis.structure\_matcher import StructureMatcher*

*In [12]: sm = StructureMatcher()*

*In [13]: sm.fit(pretend\_made\_structure,v.initial\_structure)*

*Out[13]: True*

*In [14]: sm.fit(pretend\_made\_structure,v.final\_structure)*

*Out[14]: True*

[I am not quite sure why we need these functions = = Normally the final structure we get after calculation should be different. But anyway it's not bad to learn something new.]

There are many other interesting functions to plot...

*[22]: v.initial\_structure.indices\_from\_symbol("Zn")[:int(len(v.initial\_structure.indices\_from\_symbol("Zn"))\*0.75)]*

*Out[22]: (0, 1, 2)*

*Ways to find out all possible combination...*

*In [23]: from itertools import combinations*

*In [24]: combinations((0,1,2,3),2)*

*Out[24]: <itertools.combinations at 0x2b8a1dd23a98>*

*In [25]: list(combinations((0,1,2,3),2))*

*Out[25]: [(0, 1), (0, 2), (0, 3), (1, 2), (1, 3), (2, 3)]*

*In [26]: list(combinations((0,1,2,3),3))*

*Out[26]: [(0, 1, 2), (0, 1, 3), (0, 2, 3), (1, 2, 3)]*