从头开始用 VASP 做结构优化

张旻烨　王越超　许熙　栾东

登入 Linux

编辑文件

VASP 输入文件解读

执行 VASP 计算

解读结构优化过程

登入 Linux

编辑文件

Linux 下常用文本编辑器: Vim

- 语法高亮
- 简单的自动补全
- 列编辑和多文件编辑
- 通过键盘输入命令实现, 同时支持鼠标操作

学习 Vim

- 善用搜索引擎 (百度, Google, …)

  想知道 vim 如何查找替换 ⇒ 搜 “vim 查找替换”
- Linux 下各种命令与程序的用法: CSDN 博客, 脚本之家

启动, 保存与退出 I

命令行输入 `vim`, 回车, 会出现 vim 的一个界面

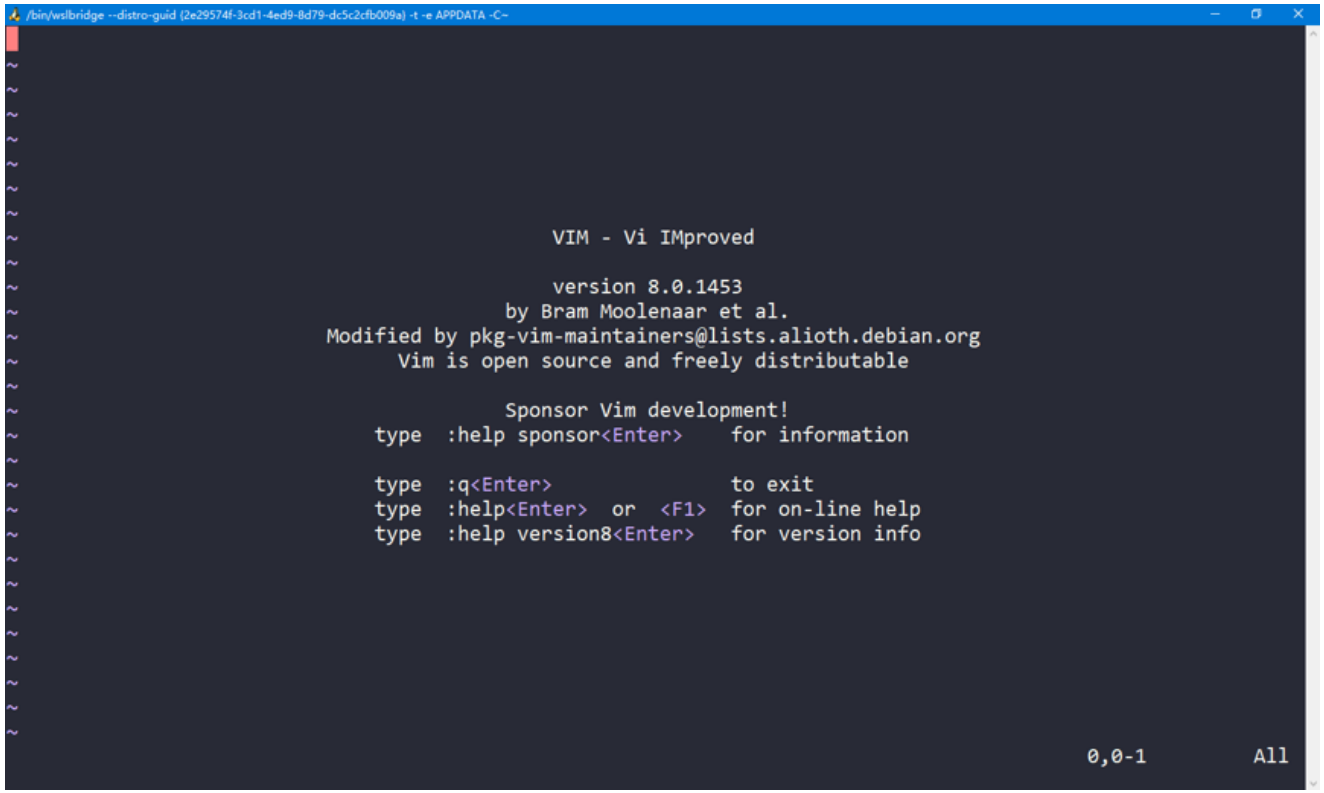
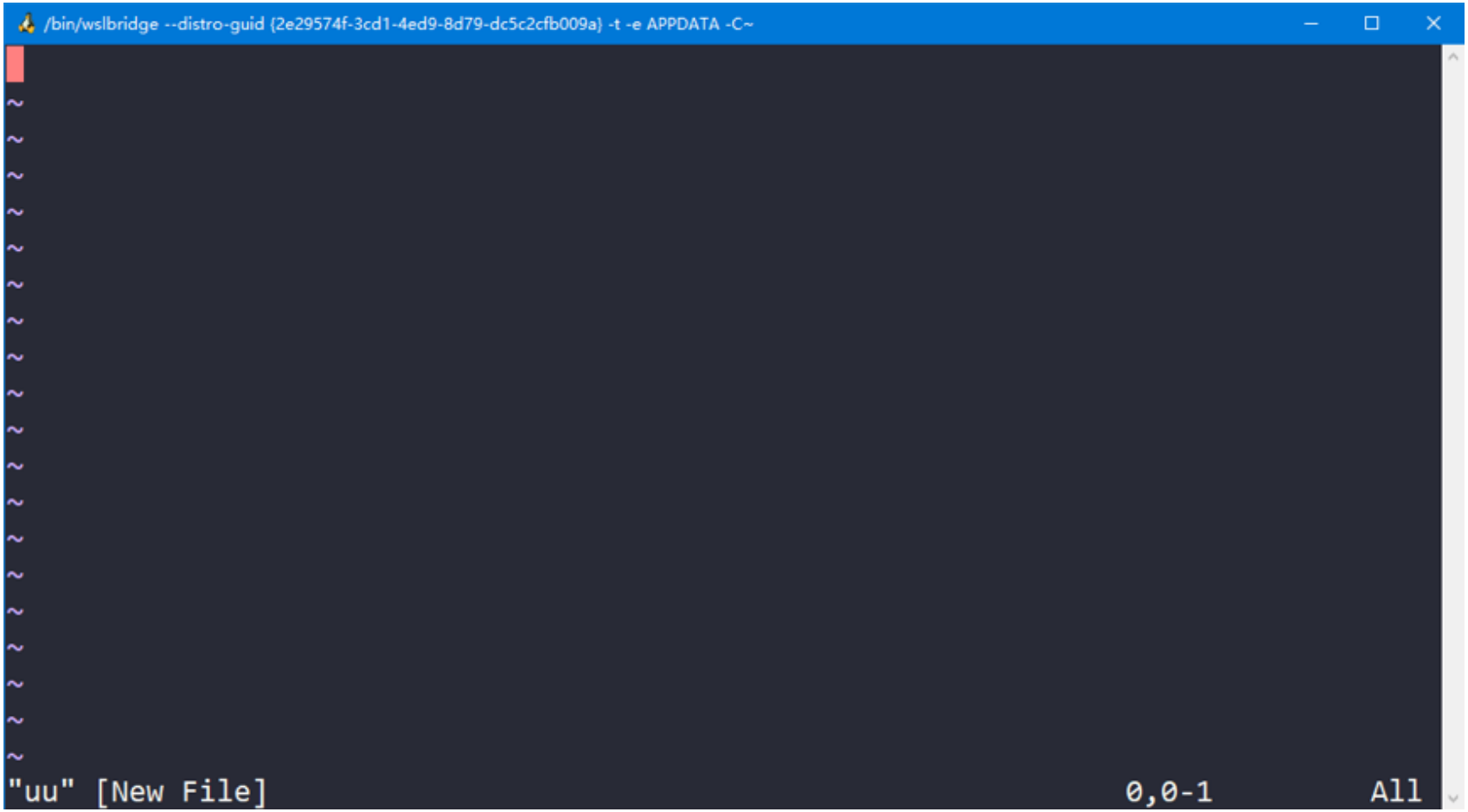


Figure 1: `vim` 界面

如果后面加上文件名就会打开相应的文件, 若该文件不存在则创建一个新文件

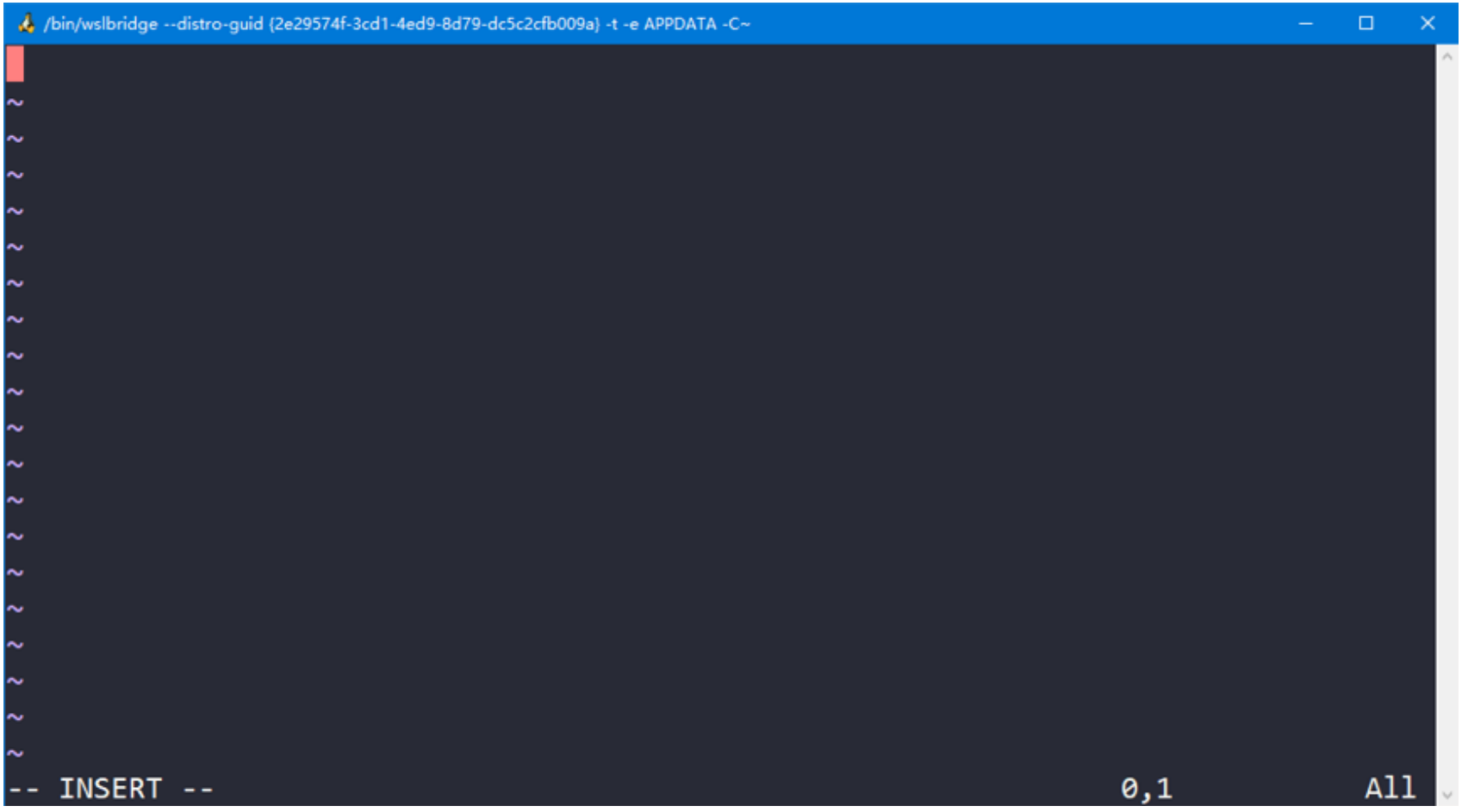
启动, 保存与退出 II

此时你在键盘上的任何输入一般都是无效的



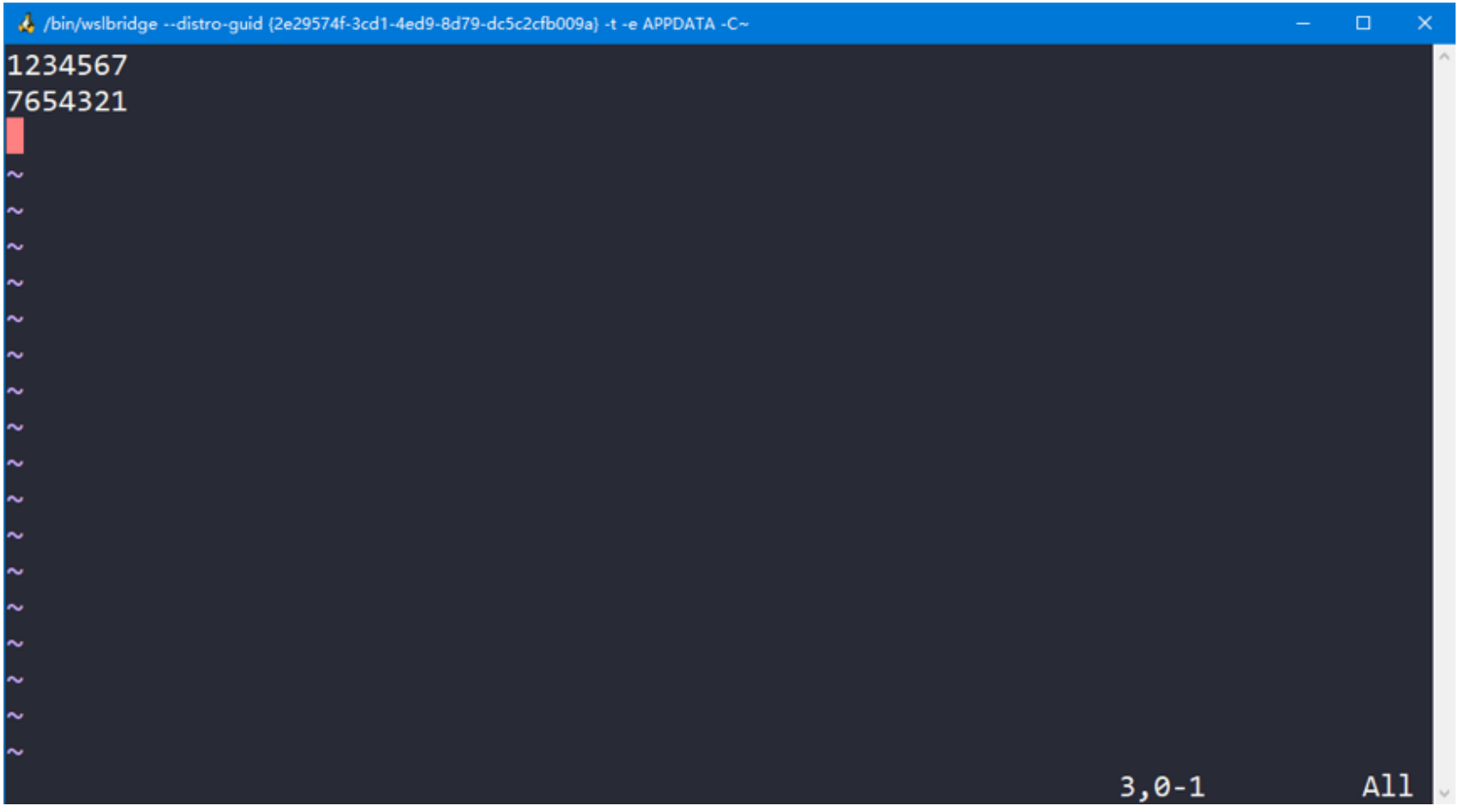
启动, 保存与退出 III

需要按一下 `i` 键变成可输入状态, 然后向里面输入内容



启动, 保存与退出 IV

需要按一下 `i` 键变成可输入状态, 然后向里面输入内容



启动, 保存与退出 V

- vim 的很多操作是通过 “命令” 进行的
- 在--INSERT--状态下, 任何键盘输入都成为了输入的内容
- 要想执行命令先要按一下 Esc 键, 从--INSERT--状态下退出

启动, 保存与退出 VI

输入命令

- `:wq`(write and quit), 回车 ⇒ 保存并退出
- `:q!` (不保存直接退出), 回车 ⇒ 直接退出, 不保存

注意: 两个命令都以一个冒号开头

复制, 粘贴和删除文本

命令	作用
`yy`	复制当前一整行的内容
`p`	把复制的内容粘贴到光标所在行的下一行
`dd`	删除光标所在的一整行内
`u`	撤销上次输入
`.`	恢复上次输入

- Vim 不支持鼠标光标选中后 Delete
- 注意上面这些都是按了 Esc 之后执行, 不需要加冒号, 在`--INSERT--`状态无法执行

文本定位, 查找和替换 I

跳转

- Shift + G 跳转到文件末尾
- 按两次 [可以跳转到文件开头
- :n 跳转到某一行使用命令, n 是一个行号
- 使用命令:set nu 显示行号

文本定位, 查找和替换 II

查找文本

- 以 `FORCE` 为例, 在非插入状态下输入/FORCE, 光标即可跳转到 FORCE 所在的位置
- 按一下 `n` 键 (next) 跳转到下一个 `FORCE` 文本所在的位置
- 要想从文件末尾查找, 则先按下 `Shift + G` 跳转到文件末尾, 然后按 `Shift + N`, 即可跳转到从末尾数的第一个 FORCE 所在位置

文本定位, 查找和替换 III

替换: 与 sed 命令类似

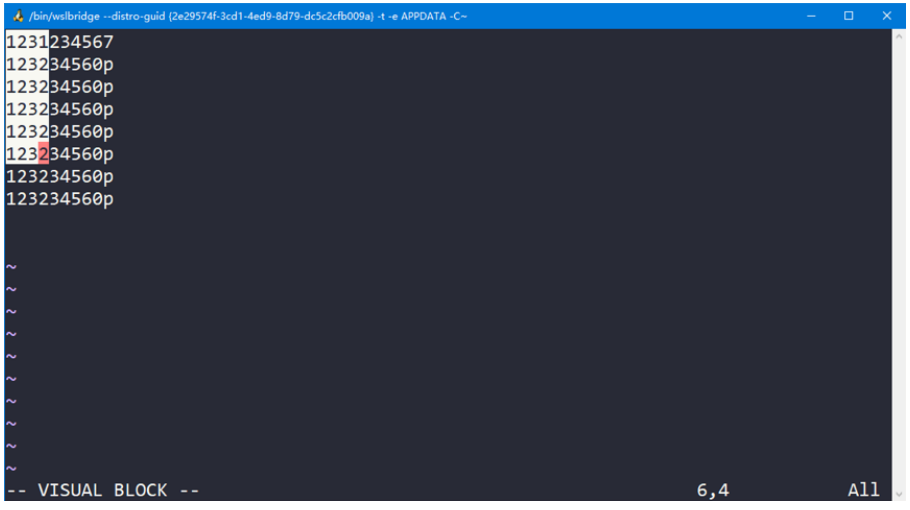
- 在非插入状态下输入命令`:%s/FORCE/force/g`, 把所有 `FORCE` 替换成 `force`
- 如果没有前面的百分号, 则每次只替换一个

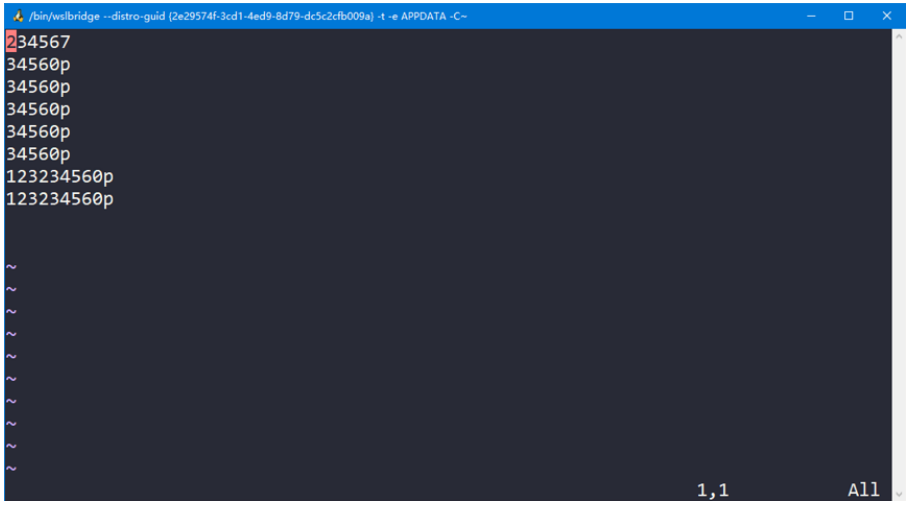
列编辑 I

非插入状态下按下 `Ctrl + V` 即进入列选择状态

按住方向键上下左右即可按列选择文本

按一下 `d` 键则删除按列选中的文本

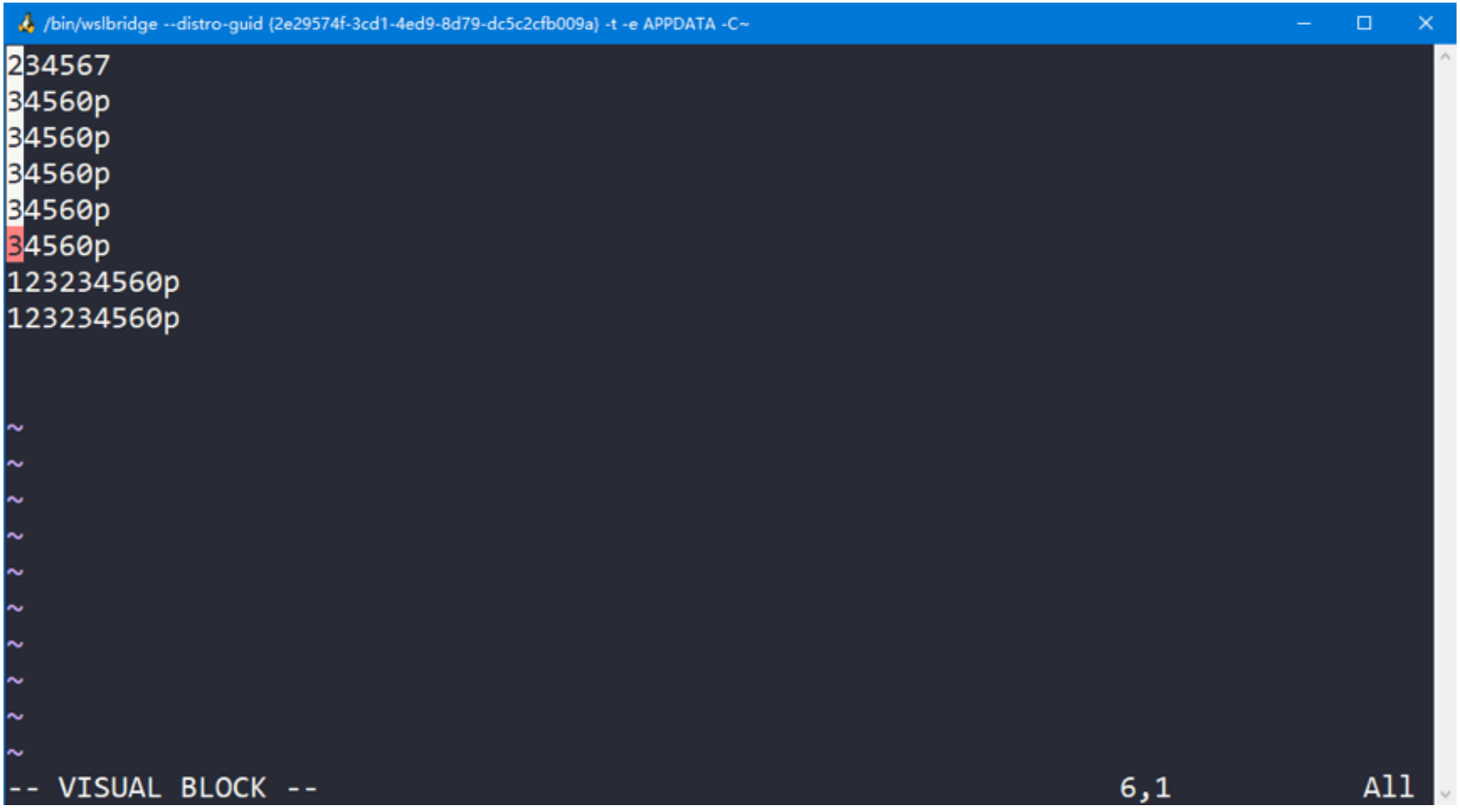




列编辑 II

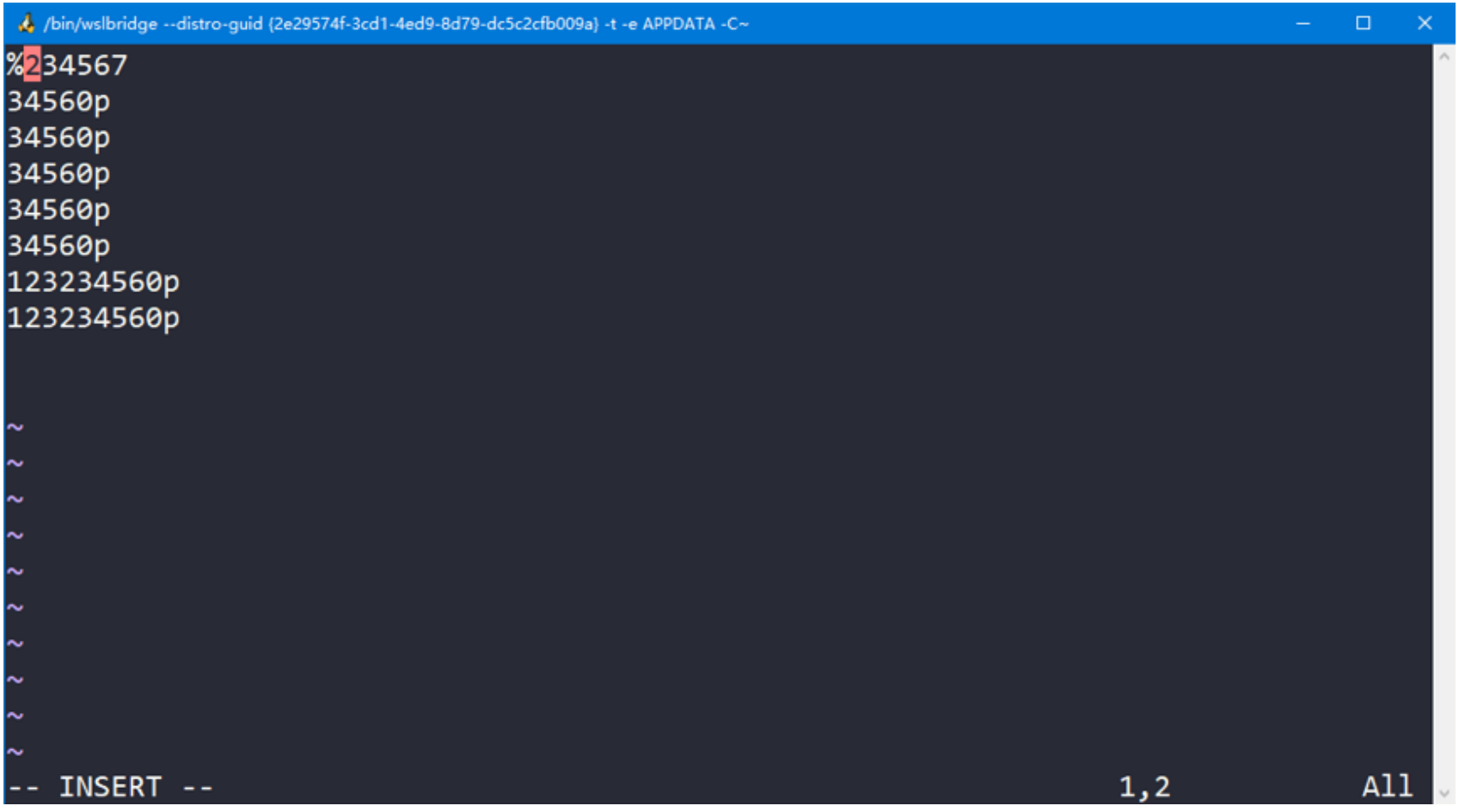
另一个常用的操作是给很多行加注释

按 Esc 键进入非插入状态, 按 `Ctrl + V` 进入列选择状态, 然后选择要插入的列范围



列编辑 III

按下 Shift + i 键, 输入文本



再连按两次 Esc, 前面选择过的行前都出现了相同的内容

VASP 输入文件解读

POSCAR: 晶体结构文件

获取晶体结构 I

Inorganic Crystal Structure Database: 实验结构

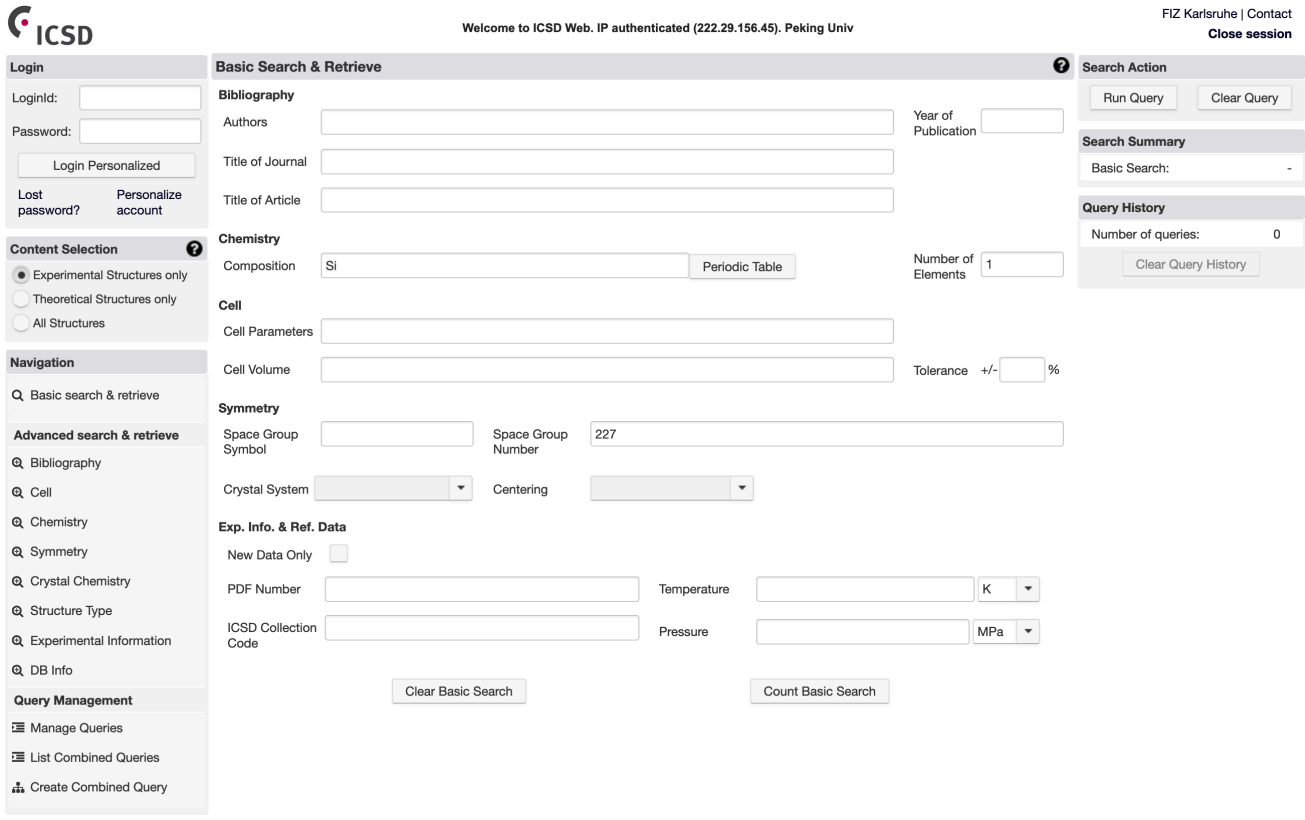


Figure 2: ICSD 搜索页面

获取晶体结构 II

AFLOW: Duke 材料基因组学数据库

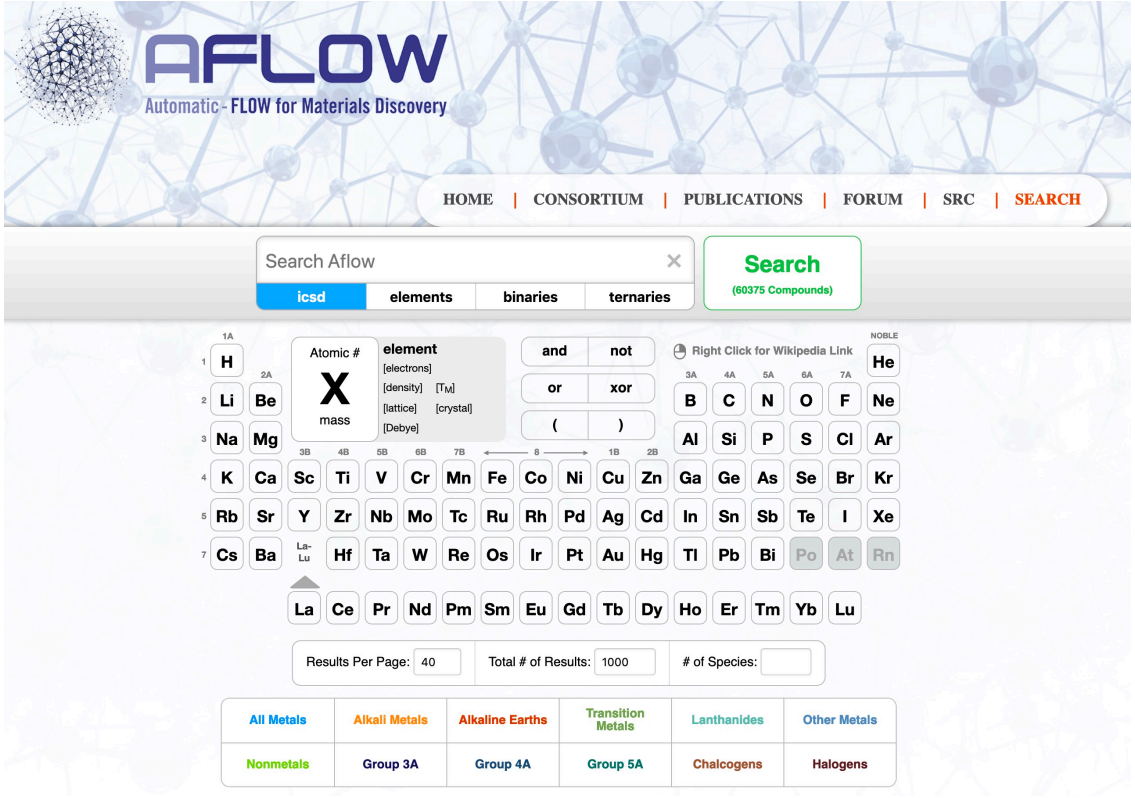


Figure 3: AFLOW 搜索页面

获取晶体结构 III

Materials Project: 基于 `pymatgen` 的材料基因组学数据库

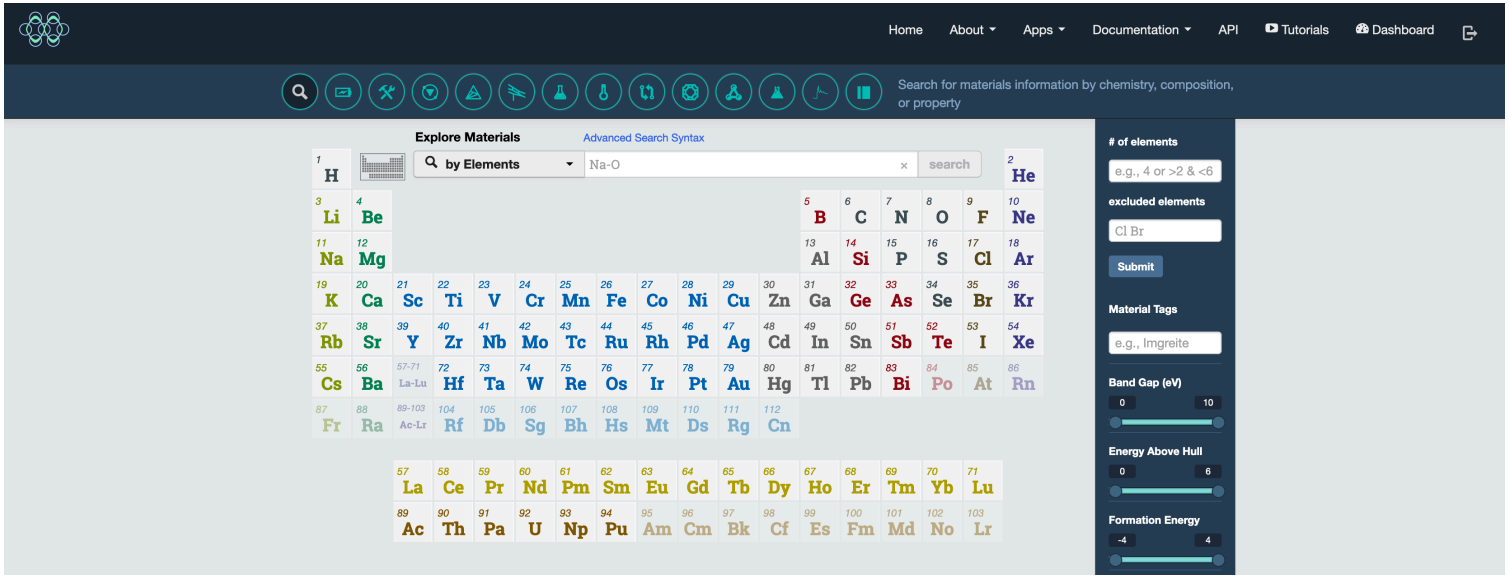


Figure 4: MaterialsProject 搜索页面

执行 VASP 计算

创建运算目录

- 教学一号

```
$ ls
POSCAR POTCAR INCAR KPOINTS sc_run_vasp.sh
$ mkdir session-1/
$ cp POSCAR POTCAR INCAR KPOINTS session-1/
$ cp sc_run_vasp.sh session-1/
$ cd session-1/
```

- TMC PC

```
$ ls
Documents Downloads tests ...
$ mkdir -p tests/YOUR_NAME
$ cp POSCAR POTCAR INCAR KPOINTS tests/YOUR_NAME
$ cd tests/YOUR_NAME
```

运行 VASP I

确认 vasp 可执行程序的位置: which

- 教学一号

```
$ which vasp_std
$ module load vasp/5.4.4-intel18.0
$ which vasp_std
/nfs-share/software/vasp/intel18.0/bin/vasp_std
```

- TMC PC

```
$ which vasp_std
/home/dft003/software/vasp.5.4.4/bin/vasp_std
```

运行 VASP II

为什么第一次 `which` 的结果不同?

在教学一号上

```
$ echo $PATH
/nfs-share/software/vasp/intel18.0/bin/:
/nfs-share/software/module/bin:/usr/local/bin:/usr/bin:
...
```

运行 VASP III

```
$ cat sc_run_vasp.sh
#!/usr/bin/env bash                    # 解释器
#SBATCH -A 150xxxxxxx                  # 学号
#SBATCH --nodes=1                      # 使用一个节点
#SBATCH -c 2                           # 每个任务用 2 个核心
#SBATCH --partition=compute            # 指定计算分区
#SBATCH -J test                        # 任务名
#SBATCH -o stdout

module load intel/2018.0               # 载入 Intel 编译器环境变量
module load vasp/5.4.4-intel18.0 # 载入 VASP 环境变量 (PATH)

mpirun -np 2 vasp_std                  # 运行
```

运行 VASP IV

- 教学一号

```
$ sbatch sc_run_vasp.sh
Submitted batch job xxxx
$ watch -n 1 cat stdout
```

- TMC PC

```
$ mpirun -np 2 vasp_std > out &
$ watch -n 1 cat out
```

解读结构优化过程