Script python para ejecutar un archivo de ordenes, con dependencias

Para tener en cuenta las dependencias entre tareas he de incluir la informacion de prioridad que debe tener cada tarea. Esto lo puedo hacer sencillamente con un numero (en este modelo que es simple). Ejemplo, la primera tarea la marco con un **1**, la tarea que depende de esta la marco con un **2**, la siguiente con un **3**, etc. Cuando terminen las dependencias y vaya a empezar con una tarea independiente, comenzare a etiquetar con un **1** nuevamente. Y asi sucesivamente,

```
[osotolongo@brick03 slurmit]$ cat listado.txt
1:echo "chr1.list"; fgrep -vw "#"
/nas/GRACE/StageII/rareVariants/plink_files/Topmed/chr1.dose.pvar | fgrep -
vw "CHROM" | awk '{print "chr"$1,$2,$2+1,$3}' | sed 's/chr23/chrX/g' | sed
's/chr24/chrY/g' | sed 's/chr26/chrM/g' | sed 's/ /    /g'  >
/nas/GRACE/StageII/rareVariants/plink_files/Topmed/chr1.dose.pvar.list
2:/nas/Adapted/Protocol/QC/liftOver
/nas/GRACE/StageII/rareVariants/plink_files/Topmed/chr1.dose.pvar.list
/nas/Adapted/Protocol/QC/hg38ToHg19.over.chain
/nas/GRACE/StageII/rareVariants/plink_files/Topmed/hglft_genome_chr1.bed
/nas/GRACE/StageII/rareVariants/plink_files/Topmed/unmapped_chr1.bed
1:echo "chr2.list"; fgrep -vw "#"
/nas/GRACE/StageII/rareVariants/plink_files/Topmed/chr2.dose.pvar | fgrep -
vw "CHROM" | awk '{print "chr"$1,$2,$2+1,$3}' | sed 's/chr23/chrX/g' | sed
's/chr24/chrY/g' | sed 's/chr26/chrM/g' | sed 's/ /    /g'  >
/nas/GRACE/StageII/rareVariants/plink_files/Topmed/chr2.dose.pvar.list
2:/nas/Adapted/Protocol/QC/liftOver
/nas/GRACE/StageII/rareVariants/plink_files/Topmed/chr2.dose.pvar.list
/nas/Adapted/Protocol/QC/hg38ToHg19.over.chain
/nas/GRACE/StageII/rareVariants/plink_files/Topmed/hglft_genome_chr2.bed
/nas/GRACE/StageII/rareVariants/plink_files/Topmed/unmapped_chr2.bed
1:echo "chr3.list"; fgrep -vw "#"
/nas/GRACE/StageII/rareVariants/plink_files/Topmed/chr3.dose.pvar | fgrep -
vw "CHROM" | awk '{print "chr"$1,$2,$2+1,$3}' | sed 's/chr23/chrX/g' | sed
's/chr24/chrY/g' | sed 's/chr26/chrM/g' | sed 's/ /    /g'  >
/nas/GRACE/StageII/rareVariants/plink_files/Topmed/chr3.dose.pvar.list
2:/nas/Adapted/Protocol/QC/liftOver
/nas/GRACE/StageII/rareVariants/plink_files/Topmed/chr3.dose.pvar.list
/nas/Adapted/Protocol/QC/hg38ToHg19.over.chain
/nas/GRACE/StageII/rareVariants/plink_files/Topmed/hglft_genome_chr3.bed
/nas/GRACE/StageII/rareVariants/plink_files/Topmed/unmapped_chr3.bed
```

Esto tiene un nivel algo mas alto de complejidad que el script anterior pero tampoco es nada del otro mundo.

Para empezar, se define el entorno en que se va a trabajar, se inicializan las variables, etc

```
#!/usr/bin/env python
import sys
```

```
import os
import re
import subprocess

time = '3:0:0'
cpus = 4
partition = 'fast'
memxcpu = '4G'
ifile = str(sys.argv[1])
wdir = 'slurm'
if not os.path.isdir(wdir): os.mkdir(wdir)
count = 0
precedence = 1
```

Ahora abrimos el fichero con las ordenes y recorremos las lineas una a una,

```
with open(ifile, 'r') as orf:
  for line in orf:
    count+=1
```

De cada linea capturo el numero y la orden posterior. Para asegurarme, convierto el numero en un entero

```
    order = re.search(r"(\d+):(.*)", line)
    norder = int(order.group(1))
```

defino, el script que voy a ejecutar y su contenido. Ojo aqui a la funcion *order.group(2)* que devuelve el segundo elemento de la linea que estamos leyendo, es decir la orden a ejecutar.

```
    ofile = wdir+'/sorder_{:04d}'.format(count)+'.sh'
    cont = '#!/bin/bash\n'
    cont += '#SBATCH -J '+ifile+'\n'
    cont += '#SBATCH -c '+str(cpus)+'\n'
    cont += '#SBATCH --mem-per-cpu='+memxcpu+'\n'
    cont += '#SBATCH --time='+time+'\n'
    cont += '#SBATCH --mail-type=FAIL,TIME_LIMIT,STAGE_OUT\n'
    cont += '#SBATCH --mail-user='+os.environ.get('USER')+'\n'
    cont += '#SBATCH -o '+wdir+'/'+ifile+'_order-%j\n'
    cont += '#SBATCH -p '+partition+'\n'
    cont += order.group(2)+'\n'
    exf = open(ofile, 'w')
    exf.write(cont)
    exf.close()
```

Ahora viene la logica de la ejecucion. Si *norder* (el primer elemento de la linea) es mayor que *precedence* (inicialmente 1) quiere decir que este script depende de otro cuyo jobid ya tenemos y podemos ejecutarlo usando un *afterok*. En caso contrario (*norder* no es mayor que *precedence*) quiere decir que estamos recorriendo un ciclo nuevo de dependencias y hacemos una ejecucion simple. en cada caso se captura de nuevo el *jobid* y se actualiza *precedence* al valor del nuemro de orden que acabamos de ejecutar.

```
    if norder > precedence:
      exthis = ['sbatch --parsable --dependency=afterok:'+str(jobid)+'
'+ofile]
    else:
      exthis = ['sbatch --parsable '+ofile]
    jobid = int(subprocess.check_output(exthis, shell=True))
    precedence = norder
```

Despues de eso basta ejecutar un script de aviso (solo envia email) usando *singleton* pues hemos usado siempre el mismo nombre de tarea (switch *-J* de sbatch).

El codigo completo aqui

slurmize_wdeps.py

```
#!/usr/bin/env python
import sys
import os
import re
import subprocess

time = '3:0:0'
cpus = 4
partition = 'fast'
memxcpu = '4G'
ifile = str(sys.argv[1])
#Creo el directorio slurm en caso necesario
wdir = 'slurm'
if not os.path.isdir(wdir): os.mkdir(wdir)
#esto es un contador para diferenciar los scripts que se ejecutan
count = 0
precedence = 1
#abro el archivo y leo linea a linea
with open(ifile, 'r') as orf:
  for line in orf:
    count+=1
    order = re.search(r"(\d+):(.*)", line) # la orden se divide en un
numero que marca la precedencia y la linea a ejecutar
    norder = int(order.group(1))
    ofile = wdir+'/sorder_{:04d}'.format(count)+'.sh'
    cont = '#!/bin/bash\n'
    cont += '#SBATCH -J '+ifile+'\n'
    cont += '#SBATCH -c '+str(cpus)+'\n'
    cont += '#SBATCH --mem-per-cpu='+memxcpu+'\n'
    cont += '#SBATCH --time='+time+'\n'
    cont += '#SBATCH --mail-type=FAIL,TIME_LIMIT,STAGE_OUT\n'
    cont += '#SBATCH --mail-user='+os.environ.get('USER')+'\n'
    cont += '#SBATCH -o '+wdir+'/'+ifile+'_order-%j\n'
    cont += '#SBATCH -p '+partition+'\n'
    cont += order.group(2)+'\n'
```

```
    exf = open(ofile, 'w')
    exf.write(cont)
    exf.close()
    if norder > precedence:
      exthis = ['sbatch --parsable --dependency=afterok:'+str(jobid)+'
'+ofile]
    else:
      exthis = ['sbatch --parsable '+ofile]
    jobid = int(subprocess.check_output(exthis, shell=True))
    precedence = norder
ofile = wdir+'/'+ifile+'_end.sh'
cont = '#!/bin/bash\n'
cont += '#SBATCH -J '+ifile+'\n'
cont += '#SBATCH --mail-type=END\n'
cont += '#SBATCH --mail-user='+os.environ.get('USER')+'\n'
cont += '#SBATCH -o '+wdir+'/'+ifile+'_end-%j\n'
cont += ':\n'
exf = open(ofile, 'w')
exf.write(cont)
exf.close()
os.system('sbatch --dependency=singleton '+ofile)
```