# resitev

January 28, 2024

## 0.1 Day 12: Rain Risk

(Povezava na nalogo)

Tokratna naloga je trivialna, zato bomo spet izkoristili priložnost, da se naučimo kaj novega in praktičnega.

Ladjo krmilimo z ukazi `N<dist>`, `S<dist>`, `E<dist>`, `W<dist>`, ki počnejo očitno, poleg tega pa ladjo še vrtimo z `L<degrees>` in `R<degrees>` ter jo pomikamo naprej v trenutni smeri s `F<dist>`. Ko gre ladja v neko smer (N, S, E, W) se ne vrti temveč po potrebi potuje ritensko ali bočno. Ladja je v začetku obrnjena na vzhod.

V drugem delu se bo izkazalo, da L in R pomenita nekaj drugega, vendar nas bo to zaradi našega zvitega načina reševanja zelo malo prizadelo.

### 0.1.1 Kompleksno računanje

Rešitev bo res elegantna, če se spomnimo kompleksnih števil. `1` in `-1` sta pač za ena vzhodno in zahodno od izhodišča, $i$ in $-i$ pa za ena severno in južno. Python ima vdelana kompleksna števila, le da jih zapisuje po inženirsko, z `j` namesto `i`, poleg tega pa moramo pred črko nujno dati številko; pišemo torej `1j`, saj `j` pomeni spremenljivko z imenom `j`.

Koordinata ladje bo torej kompleksno število. Če hočemo 10 korakov zahodno, mu prištejemo -10. Če hočemo 10 korakov severno, prištejemo `10j`.

Tudi smer ladje bo kompleksno število, vendar vedno velikosti 1. Ker se ladja obrača le pod pravimi koti, bo smer vedno `1`, `1j`, `-1` ali `-1j` (tule smo jih našteli v krogu – V-S-Z-J). Zdaj pa pride najlepše: obračanje za 90 stopinj levo je enakovredno množenju z $i$, saj velja $1 \times i = i$, $i \times i = -1$, $-1 \times i = -i$ in $-i \times i = 1$. Obračanje za 180 stopinj je enakovredno dvema obratoma za 90, se pravi dvema množenjema z $i$, se pravi množenju z $i^2$. Ekvivalentno temu je obračanje za 270 stopinj ekvivalentno množenju z $i^3$. (To velja celo na splošno, ne le za prav kote: obrat za $d$ stopinj je ekvivalenten množenju z $i^{d/90}$).

Obrati na desno so podobni, le da tam pač množimo z $-i$ oziroma njegovimi potencami.

### 0.1.2 Priprava podatkov in konstant

V slovar `dirs` smo shranili "vektorje" dolžine 1, ki ustrezajo smerem neba. V `lr` smo shranili faktorje, s katerim je potrebno pomnožiti smer ob obratih na levo in na desno.

Podatke pa preberemo v terke (smer, razdalja).

[1]:
```
dirs = dict(E=1, N=1j, W=-1, S=-1j)
lr = dict(R=-1j, L=1j)

instr = [(v[0], int(v[1:])) for v in open("input.txt")]
```

### 0.2 Prvi del

[2]:
```
coord = 0
dir = dirs["E"]

for where, dist in instr:
    if where in dirs:
        coord += dirs[where] * dist
    elif where == "F":
        coord += dir * dist
    else:
        dir *= lr[where] ** (dist / 90)

print(int(abs(coord.real) + abs(coord.imag)))
```

```
938
```

Začetne koordinate (`coord`) so 0, smer (`dir`) pa je vzhod.

V zanki preverimo, ali - gre premike v smereh neba in v tem primeru uporabimo slovar smeri, s tem da vsako smer pomnožimo z razdaljo; - gre za premik naprej, ko premaknemo ladjo za podano razdaljo v trenutni smeri; - gre za obrat, ko smer `dist/90`-krat pomnožimo z $i$ ali $-i$.

Naloga sprašuje po Manhattanski razdalji od izhodišča, ki jo izpišemo in izračunamo v `print`-u.

### 0.3 Drugi del

Drugi del uvede “waypoint”. Predstavljajte si ga kot navidezno točko, ki je na neki določeni razdalji in v določeni smeri od ladje. Edini ukaz, ki premika ladjo je F: premakne jo v smeri proti tej točki in sicer za toliko razdalij do te točke, kolikor piše; če je točka od ladje oddaljena 8, bo F10 premaknil ladjo za 80 enot v smeri, v kateri je ta točka.

Vsi ostali ukazi premikajo to točko. Ukazi E, S, W in N jo premikajo za podano razdaljo v smereh neba. Ukaza L in R zavrtita waypoint glede na ladjo. Če je bil prej 10 enot spredaj in 1 desno, bo po obratu za 90 stopinj na desno za 10 enot desno in 1 zadaj.

Rešitev je praktično enaka rešitvi prvega dela!

[3]:
```
coord = 0
dir = 10 + 1j

for where, dist in instr:
    if where in dirs:
        dir += dirs[where] * dist
    elif where == "F":
```

```
        coord += dir * dist
    else:
        dir *= lr[where] ** (dist / 90)

print(int(abs(coord.real) + abs(coord.imag)))
```

```
54404
```

Spremembi sta le dve. - `dir = dirs["E"]` je bilo potrebno spremeniti v `dir = 10 + 1j`, ker je takšen pač začetni položaj waypointa; - `coord += dirs[where] * dist` je bilo potrebno spremeniti v `dir += dirs[where] * dist`, saj ukazi N, S, E, W ne premikajo več ladje, temveč waypoint. Vse ostalo je enako!

Kompleksna števila so zakon.