Lösningsbeskrivning

För att köra

Följande är en enkel beskrivning av hur man kör programmet

```
# Generisk beskrivning:
#  * num_procs: Antalet process som process-poolen får använda under körning
#  * file_glob_x: En filsökväg som får innehålla wildcards (*) för att matcha filer
# python parallel_search.py <num_procs> <file_glob_0> ... <file_glob_N>

# Exempel:
python parallel_search.py 2 data/utf*.txt
```

Implementationens egenskaper:

- Beroende på vilken typ av disk programmet läser från har programmet potential att vara I/O-bundet
  - Implementationen kommer att skala sämre med låga läshastigheter på disken från vilken filer hämtas
  - Använder man tillräckligt många trådar som läser samtidigt från samma disk kommer problemet att vara I/O-bundet
  - När programmet når I/O begränsningar kommer möjligheten att skala begränsas
- Implementationen är begränsad till att skala upp till och med antalet filer
- Beräkningsbördan för processerna är jämn om storleken filer emellan är jämn

Implementationsbeskrivning

- En process-pool används för att parallellisera över tillgängliga filer
  - Ett förvalt antal processer allokeras för arbetet
- En process exekverar hela flödet för en fil:
  - Initialisera `lokalt set`
  - Öppna filbuffer
  - För varje rad i filen:
    - Om första bokstaven är 'r' -> lägg till i `lokalt set`
  - Returnera `lokalt set`
- Huvudprocessen samlar in resultat från beräkningsprocesserna
- Resultaten slås samman sekventiellt

Möjliga förbättringar

- Implementera parallellisering över enskilda filer för att möjliggöra ytterligare skalning
  - Kan vara en aning problematiskt i standardpython pga. GIL

  - Vill man försatt göra en så liten operation som i lekexemplet kan det vara värt att fundera på att välja ett mer lågnivåspråk för att få maximal vinst av trådningen.
    * För detta bör man sannolikt välja ett API som stödjer en smidig modell av delat minne för att undvika överflödiga kopieringar tex. C, C++ med OpenMP
- Om man indexerar filernas innehåll borde det vara lättare att effektivt parallellisera inläsningen av chunks