Upute za Projekt iz HMO-a 2018./19.

Zamjena grupa temeljem želja studenata

Definicija problema

Na FER-u studenti koriste zamjenu grupa kako bi promijenili grupu u kojoj bi htjeli slušati određenu aktivnost na predmetu (npr. predavanja na Fizici).

Za što uspješniju zamjenu grupa potrebno je napraviti algoritam koji će temeljem prikupljenih zahtjeva studenata napraviti što veći broj zamjena studenata.

Vremensko ograničenje izvođenja programa

Vrijeme za izvođenje algoritma u kojem će se pokušati doći do najboljeg rješenja treba se moći navesti u parametrima programa. Po isteku vremena program mora završiti s radom, te spremiti sve podatke u izlaznu datoteku, neovisno o broju ulaznih podataka.

Program će se testirati nad više intervala izvođenja: 10 minuta, 30 minuta i 60 minuta.

Pokretanje programa

Program može biti pisan u bilo kojem programskom jeziku, a podatke treba moći primati putem komandne linije (eng. *command line*).

<program> označava naziv programskog rješenja koje se pokreće (npr. *program.exe*, *program.py*,..), a treba ga moći pokretati u komandnoj liniji sa sljedećim parametrima:

```
<program> –timeout 600 –award-activity "1,2,4" –award-student 1 –minmax-pen
alty 1 –students-file student.csv –requests-file requests.csv –overlaps-fil
e overlaps.csv –limits-file limits.csv
```

Dostupni parametri programa:

- ***timeout***
  - vrijeme u sekundama nakon kojeg program treba završiti s izvođenjem i izraditi izlaznu datoteku
- ***award-activity***
  - nagradni bodovi prema broju aktivnosti za pojedinog studenta za koje je napravljena zamjena. U ovom primjeru, za studenta kod kojeg su napravljene zamjene na 2 aktivnosti, ukupno rješenje dobiva broj bodova koji je naveden kao drugi parametar (2), a kod broja napravljenih zamjena koji je veći od broja parametara, ukupno rješenje dobiva iznos bodova navedenog kao zadnji parametar. Koristi se u ukupnoj ocjeni rješenja
- ***award-student***
  - nagradni bodovi za rješavanje kompletnih zamjena jednog studenta. Koristi se u ukupnoj ocjeni rješenja

- ***minmax-penalty***
    - iznos bodova za koje se umanjuje ukupna ocjena rješenja za svakog studenta u svakoj grupi u kojoj ima više studenata od željenog (*max_preferred*), ili manje studenata od željenog (*min_preferred*). Koristi se u ukupnoj ocjeni rješenja
- ***students-file***
    - naziv datoteke u kojoj se nalaze podaci o aktualnim grupama u kojima se nalaze studenti (CSV). Ovo je ujedno i datoteka u koju će se spremiti rezultat izvođenja programa
- ***requests-file***
    - naziv datoteke u kojoj se nalaze podaci o zahtjevima studenata – grupe u koje žele ući (CSV)
- ***overlaps-file***
    - naziv datoteke u kojoj se nalaze podaci o zabranjenim kombinacijama grupa (preklapanjima) (CSV)
- ***limits-file***
    - naziv datoteke u kojoj se nalaze podaci o ograničenju u broju studenata u grupama.

Stroga ograničenja (eng. *hard constraints*)

Algoritam treba napraviti najbolje moguće rješenje pri čemu treba voditi računa o ovim strogim ograničenjima – ona se moraju poštivati:

- broj studenata u pojedinoj grupi ne smije biti:
    - manji od zadanog minimuma
      (parametar *min* u datoteci *limits-file)*
    - veći od zadanog maksimuma
      (parametar *max,* u datoteci *limits-file)*
- student ne smije biti istovremeno u dvije grupe koje imaju preklapanje, osim ako su te grupe u inicijalnim podacima o grupama (*students-file*). Ako je student već imao preklapanje u inicijalnom stanju, ne smije mu se napraviti novo preklapanje, ali će se rješenje smatrati ispravnim ako je student u konačnom rješenju ima preklapanje u istim grupama kao i u inicijalnom rješenju
  (konačne grupe studenta ne smiju biti navedene u datoteci *overlaps-file)*
- student ne smije biti premješten u drugu grupu ako za tu aktivnost nije zatražio zamjenu (student, aktivnost i nova grupa moraju biti u ulaznoj datoteci *requests-file)*
- student na jednoj aktivnosti mora biti u točno jednoj grupi

Ocjena rješenja

Prihvatljivim rješenjem smatra se ono koje ispunjava sva prethodno navedena tražena ograničenja. Rješenja koja nisu prihvatljiva imaju ukupnu ocjenu 0.

BodoviA računaju se kao zbroj težišnih faktora (*swap_weight*) u datoteci (*students-file*) kod kojih je napravljena tražena zamjena

BodoviB računaju se kao zbroj svih nagradnih faktora za studenta (parametar *award-activity* prilikom poziva programa). Nagradni faktor se računa temeljem broja aktivnosti za koje je napravljena zamjena za jednog studenta. Ako je napravljena jedna zamjena tada se uzima prvi

broj, ako su napravljene dvije drugi itd. Ako je zadano manje brojeva od aktivnosti koje su zamijenjene tada se uzima zadnje zadani broj. Npr. parametar glasi "1,2,4". Studentu A je napravljena zamjena za 2 aktivnosti. Studentu B je napravljena zamjena za 5 aktivnosti. Za studenta A se dobiva 2 bodova, a za studenta B dodatnih 4 boda. Ukupno BodoviB su jednaki zbroju bodova svih studenata, odnosno u ovom primjeru 6 (2+4) bodova.

BodoviC računaju se za studente kod kojih se uspješno provedu zamjene za sve aktivnosti na kojima su tražili zamjene prema umnošku broja takvih studenata i nagradnog faktora za potpune zamjene (parametar *award-student* prilikom poziva programa).

BodoviD računaju se kao suma bodova svih grupa kod kojih je broj studenata < min_preferred tada je za tu grupu formula: (*min_preferred-students_cnt*)*(*minmax_penalty*). Na primjer ako je u grupi *min_preferred* 10 studenata, a u grupi se nakon zamjene nalazi 8 studenata, a parametar *minmax-penalty* iznosi 7, tada je za tu grupu vrijednost: (10-8)*7=2*7=14

BodoviE računaju se kao suma bodova svih grupa kod kojih je *students_cnt* > *max_preferred* tada je za tu grupu formula: (*students_cnt* - *max_preferred*)*(*minmax_penalty*)

Ukupna ocjena rješenja = BodoviA + BodoviB + BodoviC – BodoviD – BodoviE

Rješenje koje ostvari veću ocjenu smatra se boljim rješenjem.

Najveća moguća ukupna vrijednost rješenja (gornja granica rješenja) je:

- BodoviA – ukupni zbroj svih težišnih faktora
- BodoviB – ukupni zbroj svih nagradnih faktora kod svakog studenta zasebno
- BodoviC – broj studenata * *award-student* faktor
- BodoviD – 0 (nula)
- BodoviE – 0 (nula)

Zbog studentskih odabira, ne mora biti moguće ostvariti najveću ukupnu vrijednost rješenja – ako se ostvari, tada program treba završiti s radom.

Ulazne datoteke

Programsko rješenja kao ulaz treba primati skup od četiri datoteke. Sve datoteke su u CSV formatu – kao separator se koristi zarez. U prvoj liniji je ispisano zaglavlje, nakon čega slijede podaci koji sadrže samo brojeve (unsigned long integer). Datoteke su u ASCII formatu te ne koriste dijakritičke znakove. Separator linija je samo oznaka za novi red ("\n").

Svi ulazni podaci su pseudonimizirani, a oni koji se koriste kao reference bit će konzistentni i jedinstveni.

ID studenta, ID aktivnosti i ID grupe može biti jednak, no svaki ID je jedinstven unutar svoje skupine. Na primjer moguće je imati ID studenta = 1, ID aktivnosti = 1 i ID Grupe = 1, ali ako dvije aktivnosti imaju isti ID to znači da je riječ o istoj aktivnosti.

students-file

Ova datoteka sadrži inicijalne podatke o inicijalnim grupama studenata - koji student (*student_id*) je u kojoj grupi (*group_id*) na kojoj aktivnosti (*activity_id*). U datoteci su samo oni studenti koji su u zamjeni grupa označili da žele promjenu grupe na barem jednoj od aktivnosti.

Isti student (*student_id*) i ista aktivnost (*activity_id*) se u ovoj datoteci pojavljuje samo jednom – jer student može biti u samo jednoj grupi.

Inicijalni broj studenata u pojedinoj grupi se ne može odrediti temeljem ove datoteke (jer u njoj nisu svi studenti, već samo oni koji sudjeluju u zamjeni). Inicijalni broj studenata u grupi je zapisan u datoteci *limits-file* u parametru *students_cnt*.

Ova datoteka je važna kako bi se moglo nakon zamjene odrediti ima li novi raspored preklapanja. Grupe koje se preklapaju su dane u datoteci *overlaps-file*.

Parametri datoteke *students-file*

- ***student_id***
  - (unsigned long integer) ID studenta (jedinstveni za svakog studenta), ovaj podatak nije JMBAG niti je povezan s njime
- ***activity_id***
  - (unsigned long integer) ID aktivnosti za koju se radi zamjena – jedan student, na jednoj aktivnosti može biti samo u jednoj grupi. Primjer takve aktivnosti je: "Predavanja iz predmeta Matematika 1", a ona će biti u datoteci navedena kao 1341. Ovaj podatak nije šifra predmeta niti je povezan s njome.
- ***swap_weight***
  - (unsigned long integer) Težišna vrijednost ove zamjene – ako se zamjena provede onda se vrijednost ukupnog rješenja povećava za ovaj iznos, vrijednost je >= 1, a predvidivo je manja od 100
- ***group_id***
  - (unsigned long integer) ID grupe u kojoj se sada student nalazi
- ***new_group_id***
  - (unsigned long integer) – **IZLAZNI** parametar – ID grupe u kojoj se student treba nalaziti – rezultat izvođenja programa, ako ne treba mijenjati grupu ili se nije mogla provesti zamjena ovo polje se treba ostaviti prazno. Kao ulazni parametar ovo polje se treba zanemariti. Ako je upisan broj veći od nule može se uzeti kao prijedlog početnog najboljeg rješenja.

Primjer ulazne datoteke:

```
student_id,activity_id,swap_weight,group_id,new_group_id
5,542434,1,31231,0
5,4326,1,31231,0
6,542434,1,315,0
7,542434,2,315,0
```

requests-file

Ova datoteka sadrži popis zahtjeva za zamjenom grupe.

Ako student želi prijeći u jednu od više grupa, redak se može ponavljati. Redoslijed zahtjeva u datoteci je nasumičan i može se zanemariti.

Studenta se smije premjestiti isključivo u jednu grupu, a grupa i aktivnost tog studenata mora biti navedena u ovoj datoteci. Ako nije navedena grupa na aktivnosti u koju bi se student htio premjestiti – to znači da se grupa za tog studenta na aktivnosti ne smije mijenjati.

Datoteka sadrži podatke:

- *student_id*
  - (unsigned long integer) ID studenta koji odgovara ID-u studenta iz datoteke *students-file*
- *activity_id*
  - (unsigned long integer) ID aktivnosti za koju se želi raditi zamjena grupa, a koji odgovara ID-u aktivnosti studenta iz datoteke (*students-file*)
- *req_group_id*
  - (unsigned long integer) ID grupe u koju student želi prijeći

Primjer datoteke:

```
student_id,activity_id,req_group_id
6,542434,315
7,542434,314
7,542434,31723
```

overlaps-file

Ova datoteka sadrži popis svih zabranjenih kombinacija grupa. Riječ je o grupama koje imaju, na primjer, preklapanje u rasporedu i slično. Student po završetku zamjene grupa ne smije imati niti jednu kombinaciju upisanih grupa iz ove datoteke. Datoteka navodi zabranjene parove.

Datoteka sadrži obje kombinacije zabranjenih grupa:

- Grupa1,Grupa2 i
- Grupa2,Grupa1.

Grupe mogu biti unutar iste aktivnosti, ali se njihova aktivnost može i razlikovati. Na primjer, student koji je upisan u grupu A iz Matematike 1, ne smije biti upisan u grupu B iz Digitalne logike jer se, na primjer, održavaju istovremeno.

Datoteka može sadržavati i podatke o grupama koje se ne nalaze u ulaznoj datoteci *students-file* – ti podaci se mogu slobodno zanemariti.

Datoteka sadrži ove podatke:

- *group1_id*
  - (unsigned long integer) ID prve grupe koja ima preklapanje, odgovara ID-u grupe iz datoteke *students-file*, *request-file*

- *group2_id*
    - (unsigned long integer) ID druge grupe koja ima preklapanje sa ID-em prve grupe, odgovara ID-u grupe iz datoteke *students-file*, *request-file*

Kombinacija *group1_id* i *group2_id* je jedinstvena u datoteci.

Ako je datoteka prazna, to znači da nema ograničenja ove vrste i sve kombinacije odabira grupe su dozvoljene.

Primjer datoteke:

```
group1_id, group2_id
315,31232
```

limits-file

Ova datoteka sadrži popis svih ograničenja o broju studenata u grupi. Vrijednost parametra *min <= max*.

Datoteka sadrži ove podatke:

- *group_id*
    - (unsigned long integer) ID grupe koji odgovara ID-u grupe iz datoteke *students-file*, *requests-file* i *overlaps-file*
- *students_cnt*
    - (unsigned long integer) trenutni broj studenata u grupi
- *min*
    - (unsigned long integer) najmanji broj studenata koji smije biti u grupi nakon provedene zamjene (>= 0). Ovo je strogo ograničenje - ispod ovog broja ne smije biti studenata u grupi. Ovaj broj je najčešće jednak *min_preferred*, ali može biti manji
- *min_preferred*
    - (unsigned long integer) željeni najmanji broj studenata u grupi nakon provedene zamjene (>= 0). Broj studenata manji od *min_preferred* se penalizira na način opisan u ocjeni rješenja kao BodoviD. Njegova vrijednost je >= *min*
- *max*
    - (unsigned long integer) najveći broj studenata koji smije biti u grupi nakon provedene zamjene (>= 0). Ako je vrijednost ovog parametra 0 to znači da u se ovu grupu ne smiju smještati novi studenti (neovisno o tome ima li u njoj već studenata ili ne). Ovaj broj je strogo ograničenje – iznad ovog broja ne smije biti studenata. Ovaj broj je najčešće jednak *max_preferred*
- *max_preferred*
    - (unsigned long integer) željeni najveći broj studenata u grupi nakon provedene zamjene (>= 0). Broj studenata veći od max_preffered, se penalizira na način opisan u ocjeni rješenja kao BodoviE. Njegova vrijednost je <= max

Primjer datoteke:

```
group_id,students_cnt,min,min_preferred,max,max_preferred
31231,7,1,2,10,10
315,3,1,1,10,10
```

Izlazna datoteka (rezultat)

Rezultat izvođenja programa treba biti modificirana ulazna datoteka *students-file* u kojoj se u zadnjem stupcu treba nalaziti nova grupa u kojoj student treba biti. Svi ostali podaci u datoteci trebaju ostati nepromijenjeni. Redoslijed podataka u datoteci se može promijeniti i nije važan.

Za studente kojima nije moguće provesti zamjenu grupa kao izlaz treba navesti grupu u kojoj su se do sada nalazili.

Primjer datoteke:

```
student_id,activity_id,group_id,new_group_id
5,542434,31231,315
5,4326,31255,31255
6,542434,315,315
7,542434,315,31231
```

U izlaznoj datoteci se može za prvi i zadnji redak vidjeti da je došlo do zamjene grupe, dok za drugi i treći redak nije bila moguća zamjena ili zamjena nije bila potrebna.

Primjer 1.

students-file

```
student_id,activity_id,swap_weight,group_id,new_group_id
10,100,1,1001,0
10,200,1,2001,0
10,300,1,3001,0
20,100,1,1002,0
20,200,1,2002,0
```

requests-file

```
student_id,activity_id,req_group_id
10,100,1002
10,100,1003
10,300,3002
20,100,1001
```

overlaps-file

```
group1_id, group2_id
2001,1003
1003,2001
```

limits-file

```
group_id,students_cnt,min,min_preferred,max,max_preferred
1001,91,90,90,100,100
1002,100,90,90,100,100
1003,90,85,85,100,100
2001,90,90,90,100,100
2002,90,90,90,100,100
3001,95,90,90,100,100
3002,90,90,90,100,100
```

Iz datoteka je vidljivo da imamo dva studenta (sa ID-em 10 i ID-em 20) koji su zatražili zamjenu na svojim aktivnostima:

- Student 10
    - na aktivnosti 100 je u grupi 1001 (prvi redak u *students-file*).
      na aktivnosti 100 bi htio preći u grupu 1002 ili 1003. (prva dva retka u *request-file*)
      U grupi u kojoj je 1001, broj studenata je veći od min (*students_cnt* > *min-preferred)* (prvi redak u *limits-file*)
      U grupi 1002 nema mjesta, broj studenata je jednak *max*. (*students_cnt* < *max-preferred*) (drugi redak u *limits-file*)
      U grupi 1003 ima mjesta, broj studenata je manji od *max* (treći redak u *limits-file*).
      Studenta ne možemo preseliti u grupu 1003 jer grupa 2001 i grupa 1003 imaju preklapanje (*overlaps-file*), a student bi nakon zamjene bio u obje grupe. S obzirom da student nije zatražio zamjenu za grupu 2001, sigurno nije moguće prebaciti studenta u grupu 1003, već kao rješenje ostaje samo grupa 1002 ako se oslobodi mjesto ili napravi direktna zamjena.
    - na aktivnosti 200 je u grupi 2001 (drugi redak u *students-file*)
      Htio bi ostati u toj grupi (nema retka u *request-file*)
    - na aktivnosti 300 je u grupi 3001 (treći redak u *students-file*)
      Htio bi preći u grupu 3002.
      U grupi 3001 ima više studenata od min, a u grupi 3002 ima manje studenata od *max-preferred*, pa je ova zamjena moguća. Kod zamjene treba paziti da algoritam interno broji broj studenata u grupi. Odnosno u ovom primjeru bi za grupu 3001 novi broj studenata u grupi bio *students-cnt* – 1 (odnosno 94), a u grupi 3002 bi uvećao za 1, *students-cnt* + 1, odnosno bio bi 91.
- Student 20
    - Na aktivnosti 100 je u grupi 1002 (četvrti redak u *students-file*)
      Htio bi preći u grupu 1001 (treći redak u *request-file*)

Ovu zamjenu može napraviti jer u grupi 1002 ima više studenata od *min-preferred*, a u grupi 1001 ima manje studenata od *max-preferred*.
Ova zamjena je moguća, a ovom zamjenom se oslobodilo i jedno mjesto u grupi, pa ako se napravi ova zamjena je moguće prebaciti studenta 10 iz grupe 1001 u 1002. Broj studenata u grupama bi ostao nepromijenjen.

- Na aktivnosti 200 je u grupi 2002 (peti redak u *students-file*)
  Htio bi ostati u toj grupi (nema retka u *request-file*)

U ovom primjeru postoji rješenje sa maksimalno tri zamjene: Studenta 10 na aktivnosti 100 treba prebaciti u grupu 1002, te na aktivnosti 300 u 3002, a studenta 20 na aktivnosti 10 prebaciti u grupu 1001.

Izlazna datoteka u slučaju rješenja sa dvije zamjene bi bila:
students-file

```
student_id,activity_id,swap_weight,group_id,new_group_id
10,100,1,1001,1002
10,200,1,2001,2001
10,300,1,3001,3002
20,100,1,1002,1001
20,200,1,2002,2002
```

Projektni zadatak

1. Smislite i implementirajte heuristiku za rješavanje opisanog problema
2. Izvedite svoj algoritam za zadane instance problema
3. Pohranite 3 rješenja za svaku instancu: rješenje dobiveno nakon 10 minuta izvođenja, 30 minuta izvođenja i 60 minuta izvođenja
4. Uz rješenja pohranite vrijednost funkcije cilja te broj iteracija u kojima se evaluirala funkcija cilja do dobivanja pohranjenog rješenja
5. Načinite izvještaj koji opisuje Vaš algoritam. Izvještaj bi trebao uključivati:
   - Opis problema
   - Opis primijenjenog algoritma tj. heuristike (prikaz rješenja, funkcija cilja/prikladnosti, način dobivanja početnog rješenja, kriterij zaustavljanja i veličina iteracije, elementi dizajna specifični za odabranu heuristiku…)
   - Pseudokod primijenjenog algoritma
   - Opis dobivenih parametara i diskusija (npr. utjecaj određenih parametara heuristike na kvalitetu rješenja, vrijeme izvođenja algoritma)
   - Diskusiju o „fairness-u“ zamjene grupa (npr. treba li algoritam prioritizirati studente koji traže samo jednu zamjenu ili studente koji traže više zamjena; je li pošteno da studentu s traženom jednom zamjenom nije provedena zamjena, dok su studentu koji je tražio 5 zamjena, provedene 3 ili čak svih 5 zamjena; je li primjerice bolje, u slučaju ako se može provesti samo 1/5

zamjena, provesti tu zamjenu ili ne provoditi zamjene uopće). Također, u raznim slučajevima, prokomentirajte kako biste definirali nagrade/kazne (težinske faktore), s ciljem unaprjeđenja zamjene grupa na FER-u. Primjerice, osvrnite se na to kako bodovima nagraditi ili kazniti zamjenu grupa za studente koji su imali 1/1 zamjenu [*broj_provedenih_zamjena*/*broj_traženih_zamjena*], 0/1, 0/n, 1/n, 3/n, n/n)

- **OPCIONALNO:** Izvedite svoj algoritam s promijenjenim vrijednostima nagrađivanja i kažnjavanja zamjena, pohranite rješenja i kratko prokomentirajte je li došlo do poboljšanja

Docker (OPCIONALNO)

Programski kod moguće je napraviti i u Docker okolini. Najbolje rješenje zadatka koristit će se u stvarnoj primjeni tijekom FER-ove zamjene grupa i izvršavat će se unutar Docker okoline.

Docker omogućava izvršavanje programa u zasebnoj okolini (slično virtualnom računalu), a omogućava instalaciju sve potrebne dodatne programske podrške i okoline potrebne za uspješno izvođenje programa. Docker okolina može se izvoditi pod MS Windows, GNU/Linux i Mac OS operacijskim sustavima. Docker okolina, jednaka je pod svim operacijskim sustavima neovisno o tome u kojima se izvodi.

Sama Docker okolina je najčešće bazirana na GNU/Linuxu, a može biti i temeljena na MS Windows okolini. Za ovaj zadatak predlaže se korištenje Docker GNU/Linux okoline temeljene na Alpine operacijskom sustavu.

Za izradu programa koji se izvršava u Docker okolini potrebno je:

- instalirati Docker okruženje (https://www.docker.com)
- kreirati Docker datoteku (Dockerfile)

Docker datoteka može se temeljiti na već gotovoj Docker okolini ili se može kreirati instalacijom potrebnih paketa.

Pokretanje programskog rješenja

Programsko rješenje će se pokretati na sljedeći način:

```
docker exec -ti my_container -v .:/data sh -c "program -timeout 600 -studen
ts-file student.csv -requests-file requests.csv -overlaps-file overlaps.csv
-limits-file limits.csv"
```

Primjer datoteka

Primjer pune, gotove, Docker datoteke koja može pokretati programe pisane u programskom jeziku Python:

Dockerfile

```
FROM python:3
```

```
ADD program.py /
RUN pip install pystrich
CMD [ "python", "./program.py" ]
```

Objašnjenje pojedinih redova:

1. *FROM python:3*
    a. Ova Docker okolina se temelji na Python3 okolini te sadrži sve potrebne elemente potrebne za pokretanje Python 3 programa.
2. *ADD program.py /*
    a. U Docker okolinu dodaje datoteku *my_script.py* iz lokalnog direktorija (ovo je program koji će se pokrenuti unutar Docker okoline)
3. *RUN pip install pystrich*
    a. Ako je za izvođenje programa potreban neka dodatna Python biblioteka ovo je primjer kako se može dodati, u ovom primjeru dodaje se biblioteka *pystrich*, ako ne treba nijedan, ova linija se može preskočiti
    b. Ako je potrebno dodati više biblioteka ova linija se može kopirati više puta, na isti način dodaje se i bilo koji drugi program
4. *CMD **[** "python", "./program.py" **]***
    a. Pokreće program u Docker okolini

Primjer programa u programskom jeziku Python koji se izvodi – cijeli program.
program.py

```
# Sample taken from pyStrich GitHub repository
# https://github.com/mmulqueen/pyStrich


from pystrich.datamatrix import DataMatrixEncoder

encoder = DataMatrixEncoder('This is a DataMatrix.')
encoder.save('./datamatrix_test.png')
print(encoder.get_ascii())
```

Primjer pokretanja programa

Za pokretanje programa potrebno je izvesti dva koraka:

1. korak – izrada Docker okoline (slično kompiliranju programskog koda)
2. korak – pokretanje programa u Docker okolini

1. korak – prije pokretanja programa potrebno je izraditi Docker okolinu – slično kompiliranju programa. Prva izrada traje duže jer se s Interneta preuzimaju svi potrebni paketi za izvođenje. Svaka iduća izrada traje znatno kraće. Docker okolina izrađuje se tako što se u command prompt napiše:

```
docker build -t student-swap  .
```

2. korak – nakon što se Docker okolina izradi potrebno ju je pokrenuti naredbom:

```
docker run -t student-swap
```

Ako se mijenja izvorni kod programa, potrebno je ponovno izraditi Docker okolinu (1. korak), a ako se ne mijenja može se samo pokretati program (2. korak).

```
$ ls
Dockerfile  program.py

$ cat Dockerfile
FROM python:3

ADD program.py /
RUN pip install pystrich
CMD [ "python", "./program.py" ]

$ cat program.py
# Sample taken from pyStrich GitHub repository
# https://github.com/mmulqueen/pyStrich
from pystrich.datamatrix import DataMatrixEncoder

encoder = DataMatrixEncoder('This is a DataMatrix.')
encoder.save('./datamatrix_test.png')
print(encoder.get_ascii())

$ docker build -t student-swap .
Sending build context to Docker daemon   5.12kB
Step 1/4 : FROM python:3
3: Pulling from library/python
bc9ab73e5b14: Pull complete
193a6306c92a: Pull complete
e5c3f8c317dc: Pull complete
a587a86c9dcb: Pull complete
72744d0a318b: Pull complete
3493e487c18d: Pull complete
a89e0510fd87: Pull complete
2100d277cd6f: Pull complete
b61a0d6fb492: Pull complete
Digest: sha256:a837aefef8f2553789bc70436621160af4da65d95b0fb02d50
32557f887d0ca5
Status: Downloaded newer image for python:3
 ---> 2cc378c061f7
Step 2/4 : ADD program.py /
 ---> 831e8c3fe75c
Removing intermediate container 6d400c429fb2
Step 3/4 : RUN pip install pystrich
 ---> Running in fcdb4dd428e8
Collecting pystrich
  Downloading https://files.pythonhosted.org/packages/a6/e1/76feb
a239737895214b5066177f6e01055083632b49312c271b1b2936f9e/pyStrich-
0.8.tar.gz (981kB)
Collecting Pillow (from pystrich)
  Downloading https://files.pythonhosted.org/packages/62/8c/23020
4b8e968f6db00c765624f51cfd1ecb6aea57b25ba00b240ee3fb0bd/Pillow-5.
3.0-cp37-cp37m-manylinux1_x86_64.whl (2.0MB)
Building wheels for collected packages: pystrich
  Running setup.py bdist_wheel for pystrich: started
  Running setup.py bdist_wheel for pystrich: finished with status
```

```
'done'
  Stored in directory: /root/.cache/pip/wheels/e7/be/0c/8fe0bd5d5
163e625d2904f676b7fd6456f63f5907115a244a7
Successfully built pystrich
Installing collected packages: Pillow, pystrich
Successfully installed Pillow-5.3.0 pystrich-0.8
 ---> f5505b7097bf
Removing intermediate container fcdb4dd428e8
Step 4/4 : CMD python ./program.py
 ---> Running in 8c9a6071c623
 ---> b81eb132ca4b
Removing intermediate container 8c9a6071c623
Successfully built b81eb132ca4b
Successfully tagged student-swap:latest
```

$ docker run -t student-swap

```
    XX  XX  XX  XX  XX  XX  XX  XX  XX  XX
    XX  XX  XX  XXXX    XX  XX      XX  XXXX
    XXXX  XX    XX    XXXX    XXXXXX    XX
    XX    XXXX      XXXX  XXXX        XX  XX
    XX    XX    XXXX      XX  XX
    XXXXXX  XX  XX  XX  XXXX  XXXXXX    XXXX
    XXXX  XX          XXXX  XXXX    XX  XX
    XXXX      XX  XXXXXX  XXXX        XXXXXX
    XX    XX      XX  XXXXXX    XX  XX  XX
    XX  XX    XXXX  XX  XXXX    XXXXXX    XX
    XXXX      XX  XX    XXXX  XXXXXX  XX
    XX  XX  XXXXXX  XX    XX  XXXX  XX    XX
    XX          XX      XXXX  XX  XXXXXXXX
    XXXXXX  XXXXXX  XXXX  XXXXXXXXXXXX    XX
    XX  XXXXXX  XX    XX      XXXXXXXX
    XX    XX    XX  XX  XX    XXXXXXXXXX  XX
    XX  XXXXXXXX  XX    XXXXXX  XX  XX  XX
    XX  XX        XXXX  XX  XXXX  XX      XX
    XXXX  XX  XXXX    XX      XX  XXXX
    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```