Prügikoristus

Empiirilisi tähelepanekuid:

- *Vastsündinute suremus* (infant mortality) — enamik objekte sureb väga noorelt.

  Reeglina 80–90% objekte sureb enne järgmise megabaidi kasutamist:

  – 60–90% CL ja 75–95% Haskelli objektidest surevad enne saamist 10 kb vanaks.

  – SML/NJ vabastab 98% objektidest iga prügikoristuse järel.

  – 95% Java objektidest on ”lühiealised”.

- Mida vanem on objekt, seda tõenäolisemalt elab ta üle järgmise prügikoristuse.

- *Viitade suunatus* (directionality of reference) — nooremad objektid reeglina viitavad vanematele.

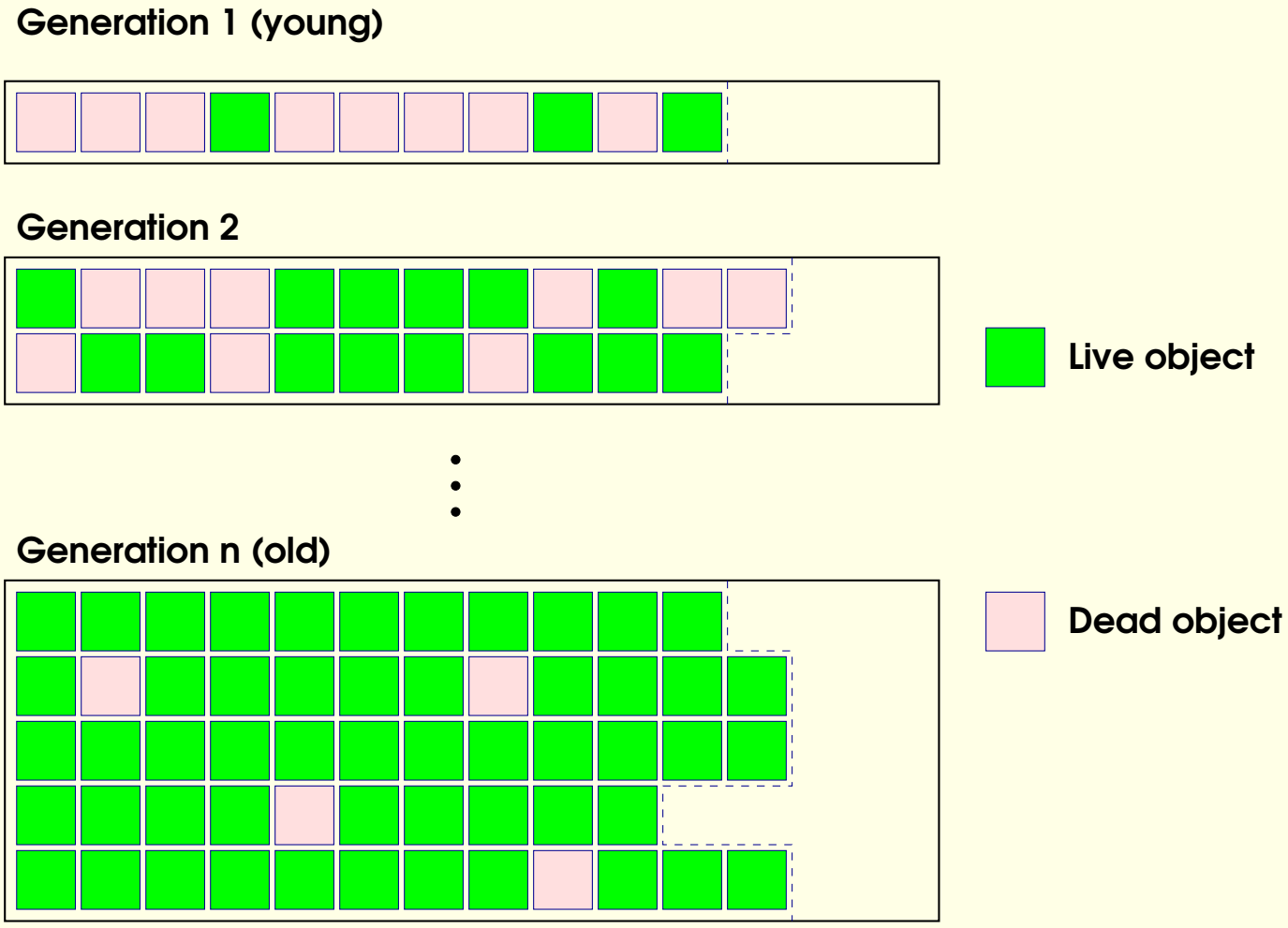
Prügikoristus

Põlvkonniti prügikoristus

- Mälu on seal paiknevate objektid vanuse järgi jaotatud *põlvkondadeks* (generations).
- Põlvkondade koguarv ja suurus on reeglina eelnevalt fikseeritud.
- Uued objektid lisatakse noorimasse, ”vastsündinute”, põlvkonda.
- Objekti vananedes edutatakse ta järgmisse põlvkonda.
- Erinevates põlvkondades toimub prügikoristus erineva sagedusega
  - põhitähelepanu pööratakse noorimale põlvkonnale.

Prügikoristus

Mälu jaotus põlvkondadeks:

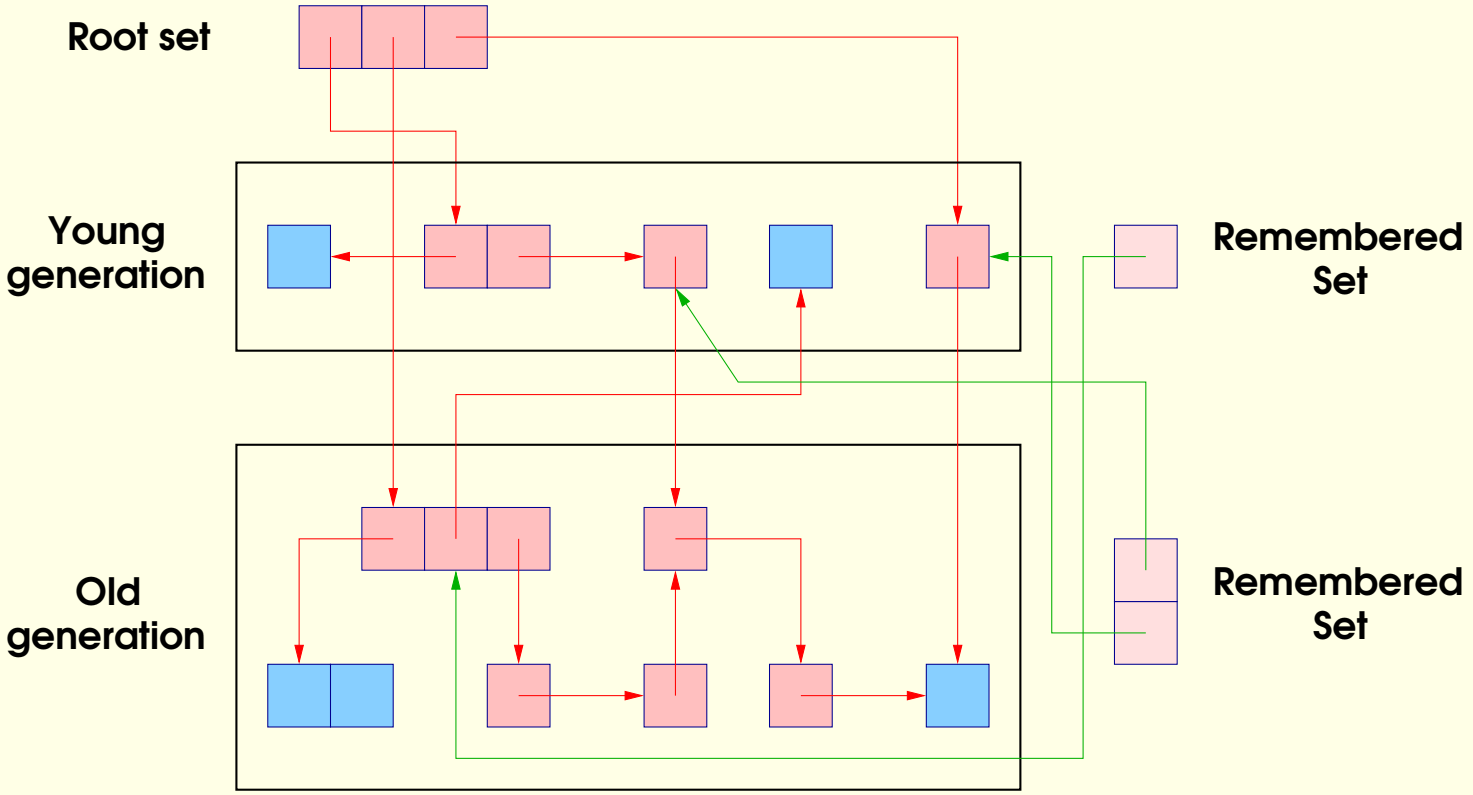


Prügikoristus

- Lisaks ”tavalistele” juurtele, on antud põlvkonna juurteks ka viidad teistest põlvkondadest temasse.
- Nende juurte asukoht pole staatiliselt kindlaksmääratav.
- Prügikoristuse aegne juurte otsimine teistest põlvkondadest on väga kulukas.
- Seetõttu seotakse iga põlvkonnaga *meelespea* (remembered set)
  - sisaldab teistest põlvkondadest tulenevaid viitu;
  - kui mingi viit on ühest põlvkonnast teise, siis lisatakse ta vastava sihtpõlvkonna meelespeasse.

Prügikoristus

Meelespead:



Prügikoristus

Probleem:

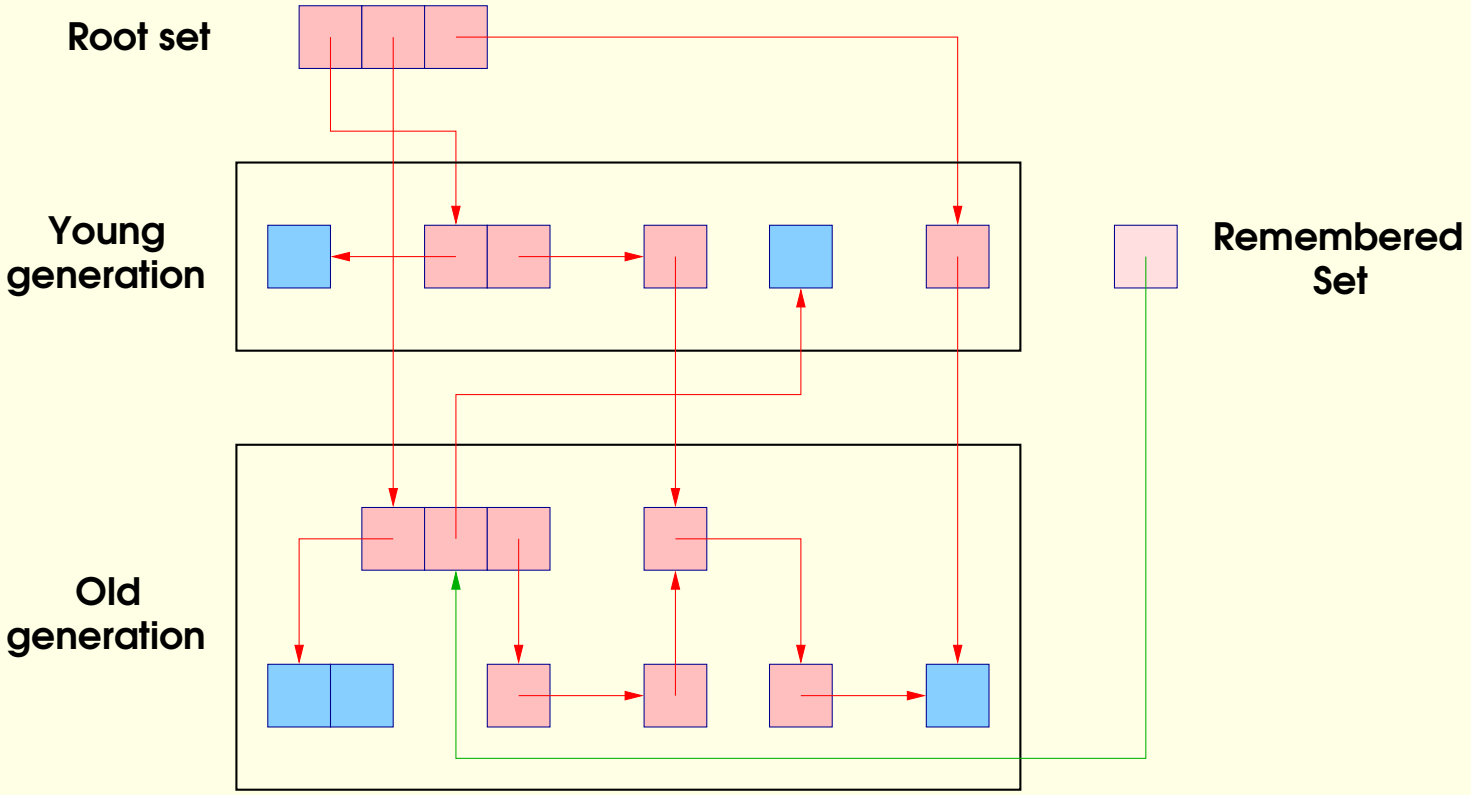
- Meelespeadele võib kuluda suhteliselt palju mälu
  - salvestada tuleb kõik põlvkondade vahelised sõltuvused.
- Meelespeasid peab haldama täitmisajal, mis võib olla väga kulukas
  - iga viitmuutuja omistamine võib potentsiaalselt olla erinevate põlvkondade vaheline.

Lahendus:

- Salvestada meelespeades ainult viidad vanematest põlvkondadest nooremates
  - kahe põlvkonna korral tarvis ainult üht meelespead (noorema põlvkonna jaoks).
- Kasutada ligikaudseid meelepeasid.

Prügikoristus

Ühesuunalised meelespead:



Prügikoristus

- Viidad *vanemast põlvkonnast nooremasse* on noorema põlvkonna juurteks:
  - selliseid viitasid esineb suhteliselt harva;
  - nad tekivad vanas objektis viida destruktiivsel muutmisel;
  - selliseid omistamisi saab kindlaks teha kasutades *kirjutustõkkeid* (write barrier).
- Viidad *nooremast põlvkonnast vanemasse* on sagedased:
  - pole probleemiks, kui vanema põlvkonna prügikoristusel alati koristatakse ka noorem põlvkond.

Prügikoristus

- Tihti kasutatakse ainult kahte põlvkonda, kus noorem põlvkond on vanemast põlvkonnast väiksema suurusega.
- Reeglina toimub ainult *pisikoristus* (minor collection), kus:
  - eemaldatakse prügi ainult nooremast põlvkonnast;
  - piisavalt vanad objektid edutatakse vanemasse põlvkonda.
- Vanema põlvkonna täitumisel teostatakse *suurpuhastus* (major collection); so. eemaldatakse prügi mõlemast põlvkonnast.
- Pisikoristus ja suurpuhastus võivad kasutada erinevaid prügikoristuse skeeme (näit. pisikoristuse korral ”copying” ning suurpuhastuse korral ”mark-compact”).

Prügikoristus

Probleemid:

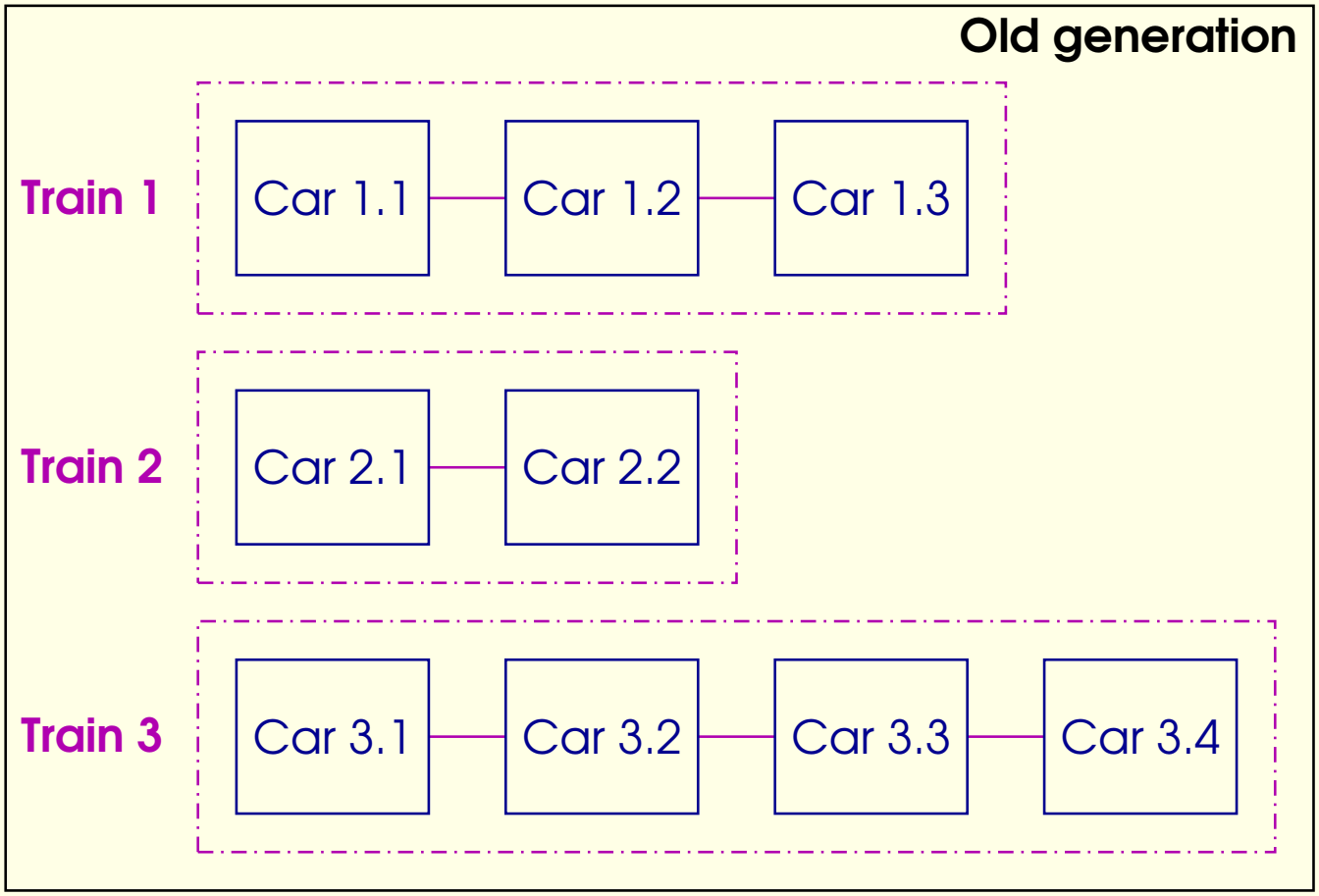
- Pisikoristused ei eemalda vana põlvkonna prügi:
  - *eakas prügi* (tenured garbage) põhjustab ka nende noorte objektide säilimise, millele ta viitab (nepotism).
- Kui vana peab objekt olema enne järgmisse põlvkonda edutamist?
  - Üks pisikoristus pole piisav, kuna vahetult eelnevalt loodud objektidel pole olnud aega suremiseks.
  - Tavaliselt loetakse piisavaks kaks pisikoristust.
- Kui suur peab olema noorim põlvkond?
  - Peab ära mahtuma põhimällu.
  - Liiga suur teeb pisikoristuse pausid liiga pikaks.
  - Liiga väike ei anna noortele objektidele aega suremiseks.

Prügikoristus

- Suurpuhastusel tekkiv paus võib olla interaktiivsete programmide jaoks liiga suur.
- Hudson'i ja Moss'i *rongi algoritm* (Train algorithm) teostab vanima põlvkonna koristamise inkrementaalselt.
- Vanim põlvkond (mature object space) jaotatakse *vaguniteks* (cars):
  - iga vaguniga on seotud oma meelespea;
  - korraga koristatakse ainult üht vagunit.
- Kuna alamstruktuurid võivad paikneda erinevates vagunites, siis grupeeritakse vagunid rongideks (trains):
  - eesmärk on seotud andmestruktuurid akumuleerida ühte rongi.

Prügikoristus

Rongi algoritm — vanima põlvkonna mälujaotus:



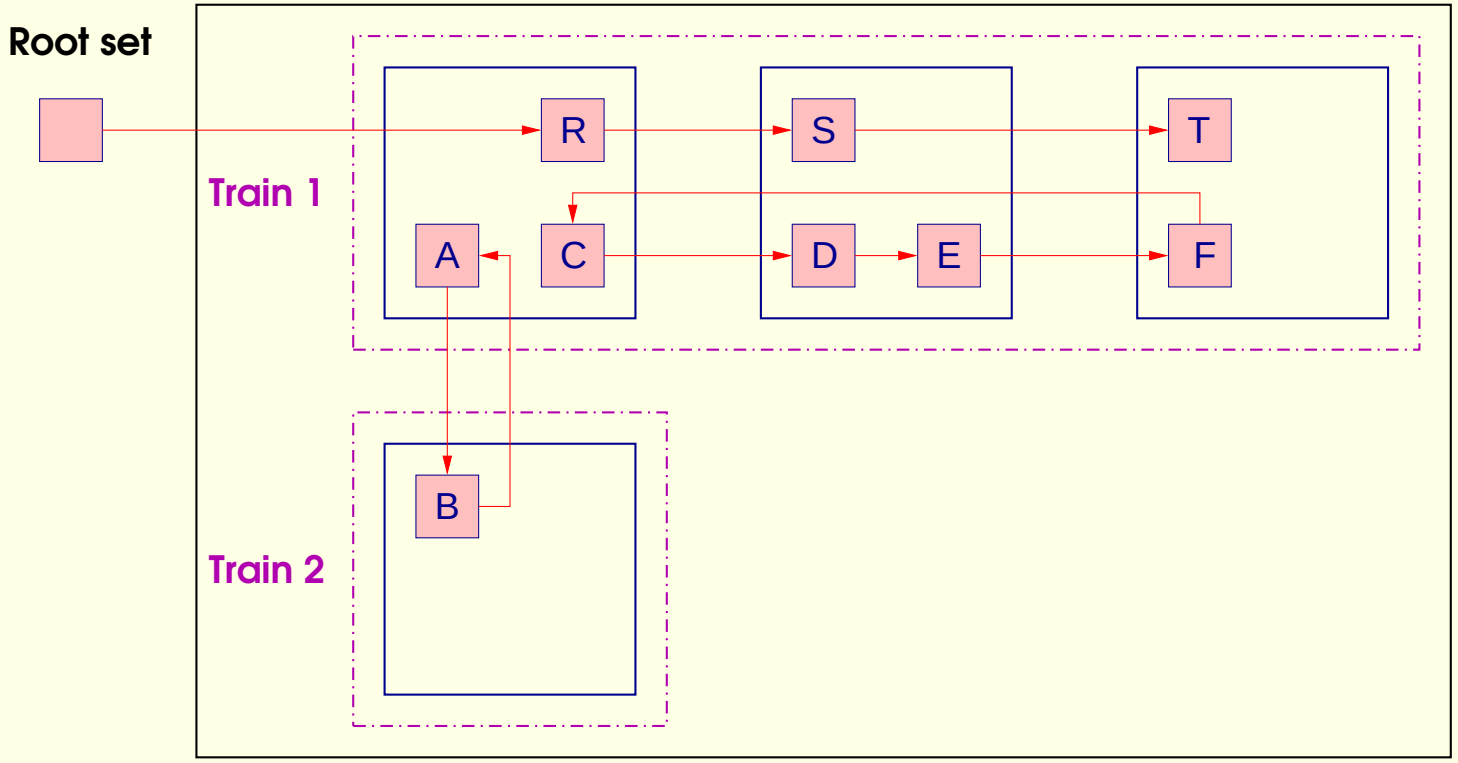
Prügikoristus

Rongi algoritm:

- Igal algoritmi inkrementaalsel väljakutsel vabastatatakse esimese rongi (FromTrain) esimene vagun (FromCar).
- Kui FromTrain-i ei ole väliseid viiteid, siis vabastatakse kogu rong.
- Vastasel korral evakueeritakse teistest rongidest viidatavad FromCar objektid (ja nende järglased) neisse rongidesse; teistest põlvkondadest viidatavad objektid (koos järglastega) evakueeritakse mõnda teise (võimalik, et täiesti uude) rongi.
- Järele jäänud FromCar välisviidad on teistest FromTrain-i vagunitest; vastavad objektid evakueeritakse FromTrain-i viimasesse vagunisse (loodes vajadusel uue vaguni), mille järel FromCar vabastatakse.

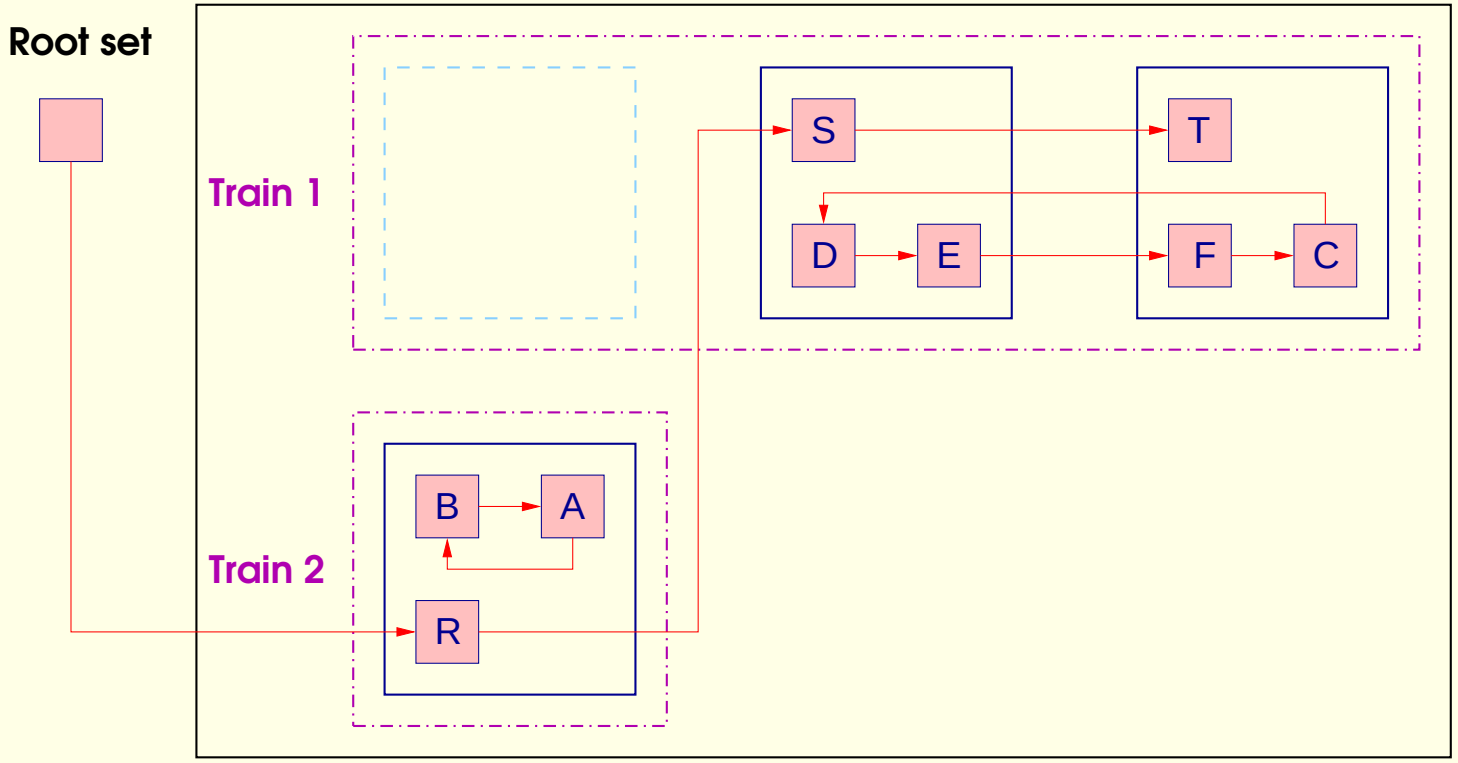
Prügikoristus

Rongi algoritm — algseis:



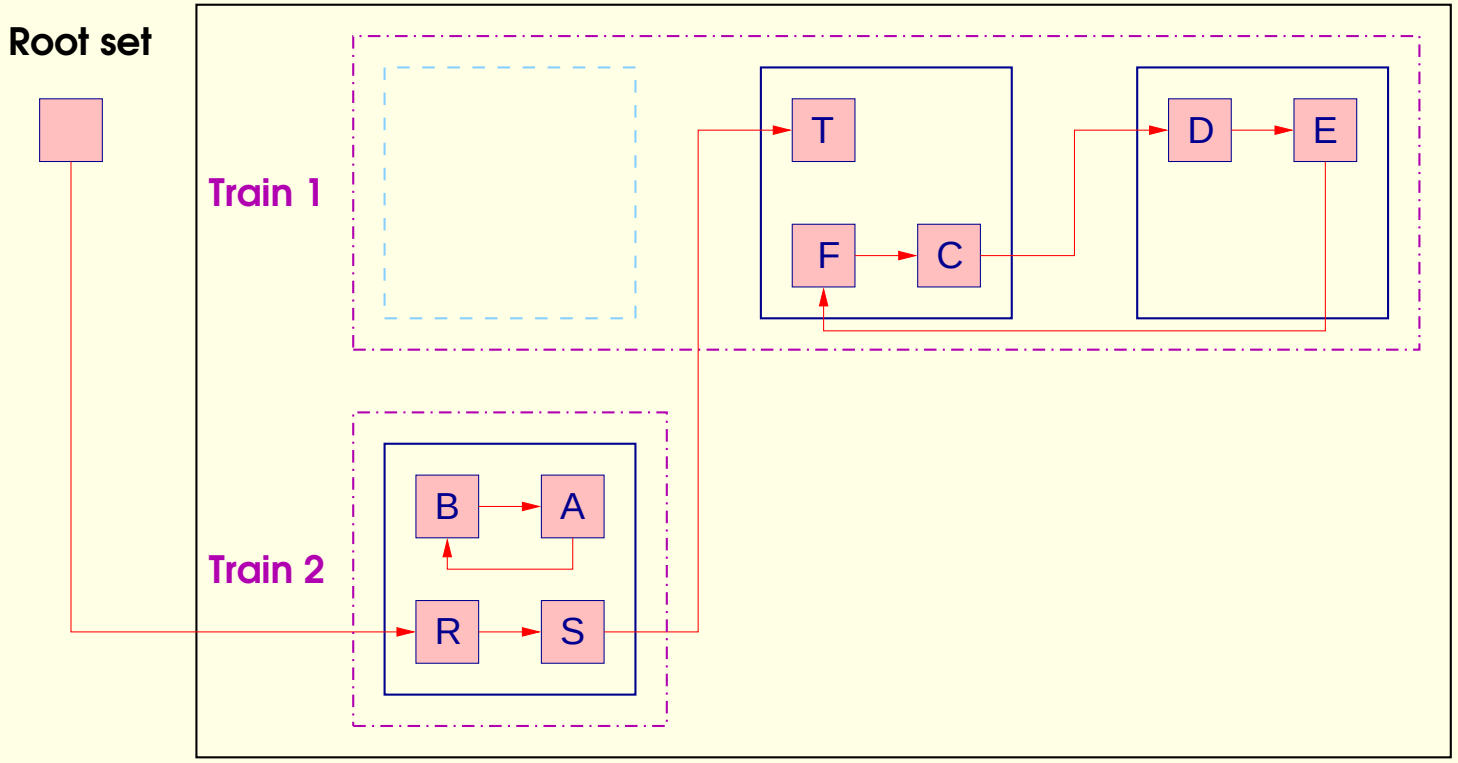
Prügikoristus

Rongi algoritm — seis pärast esimest koristust:



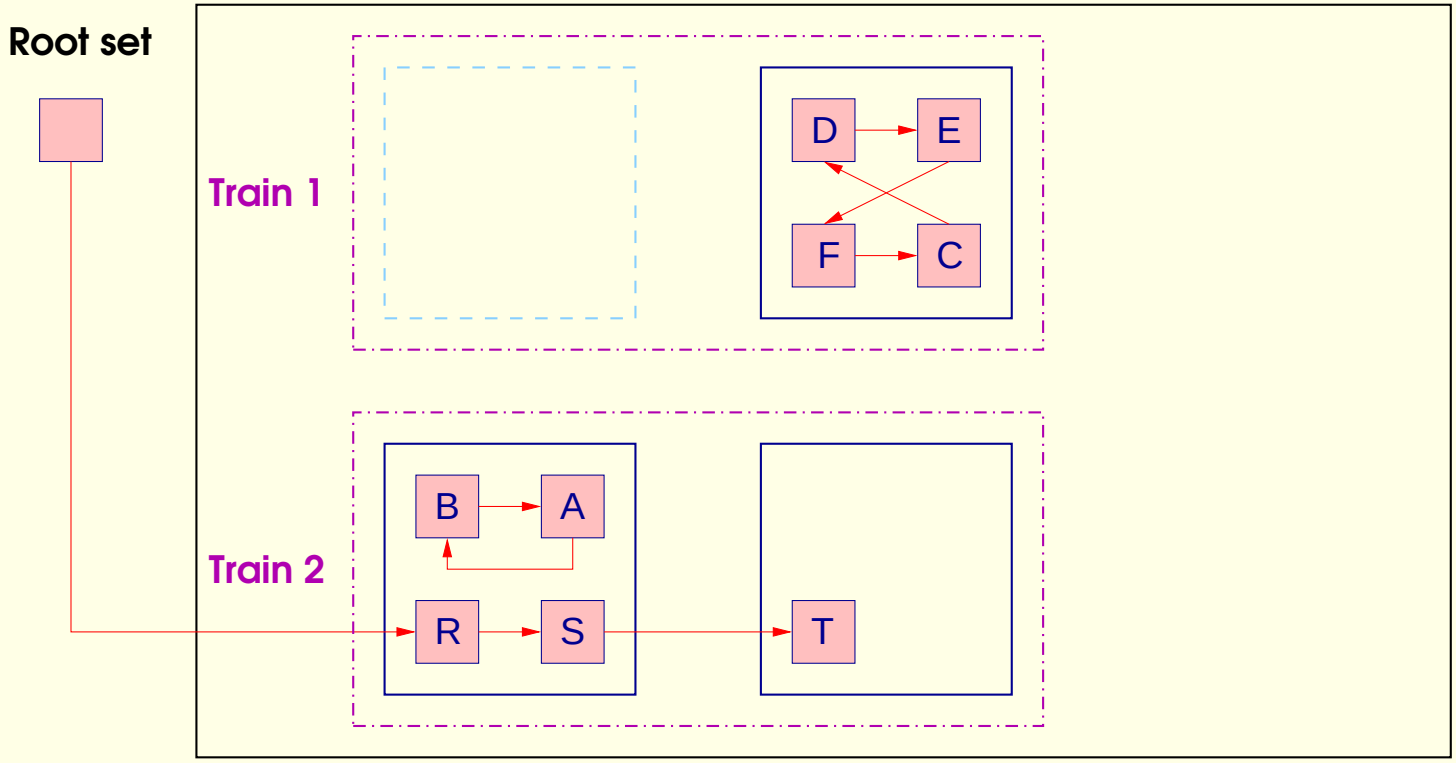
Prügikoristus

Rongi algoritm — seis pärast teist koristust:



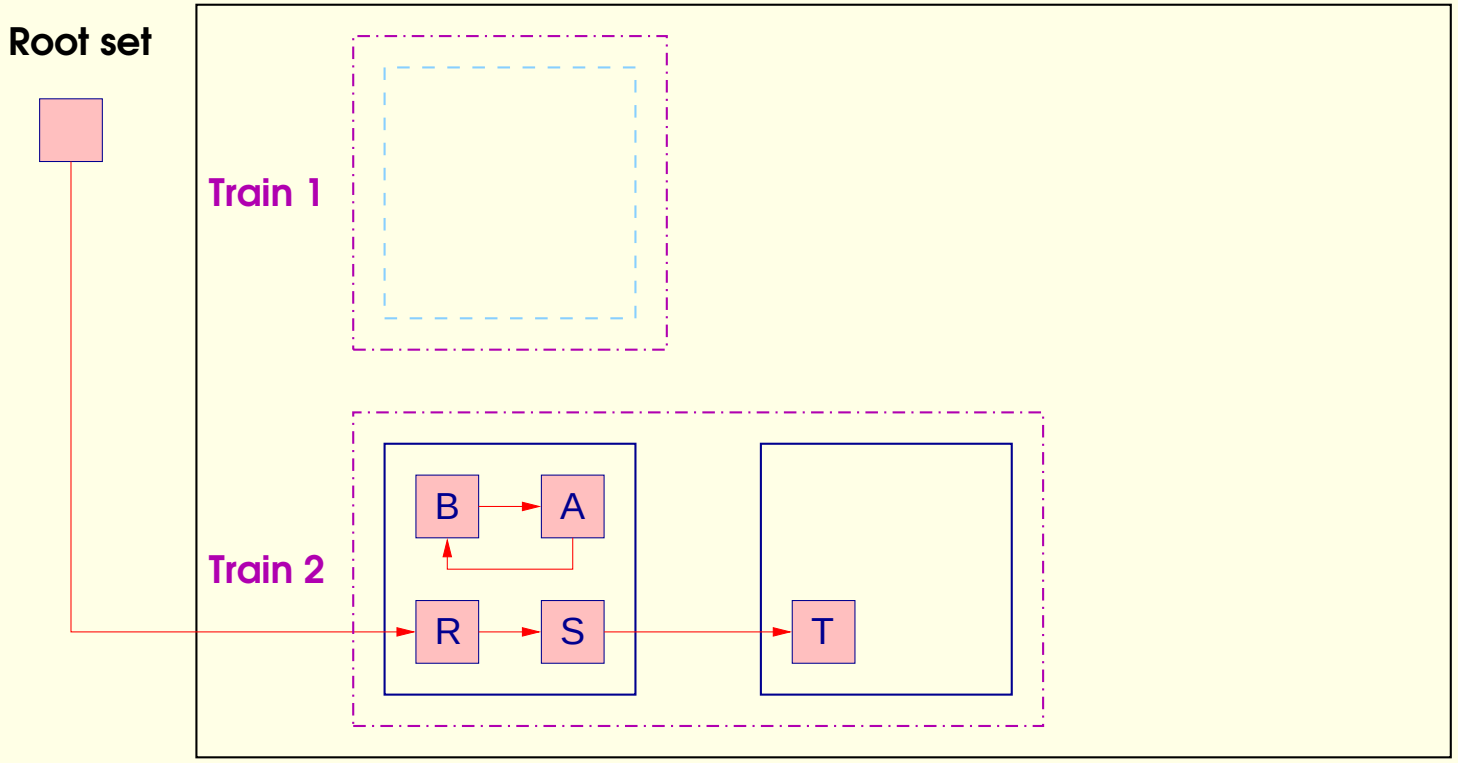
Prügikoristus

Rongi algoritm — seis pärast kolmandat koristust:



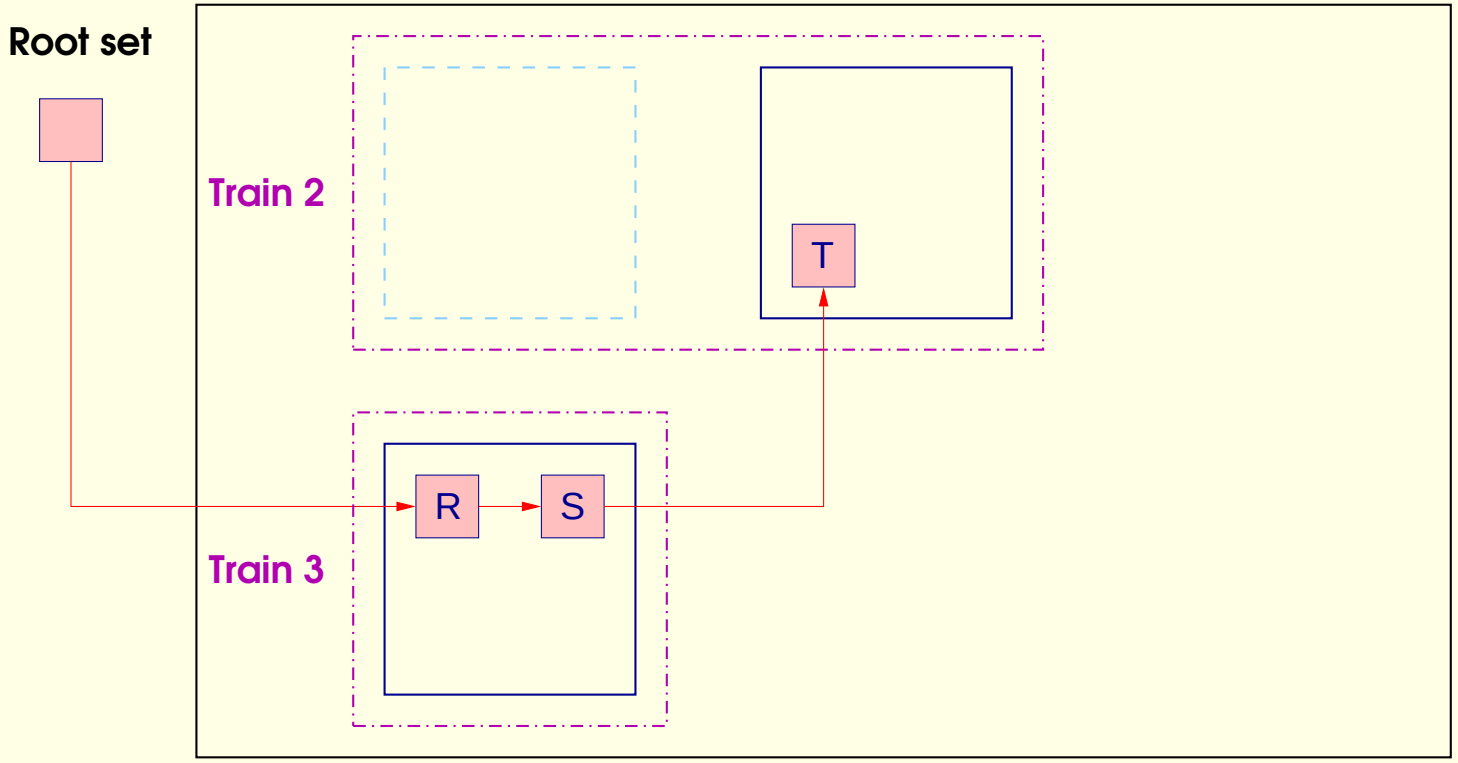
Prügikoristus

Rongi algoritm — seis pärast neljandat koristust:



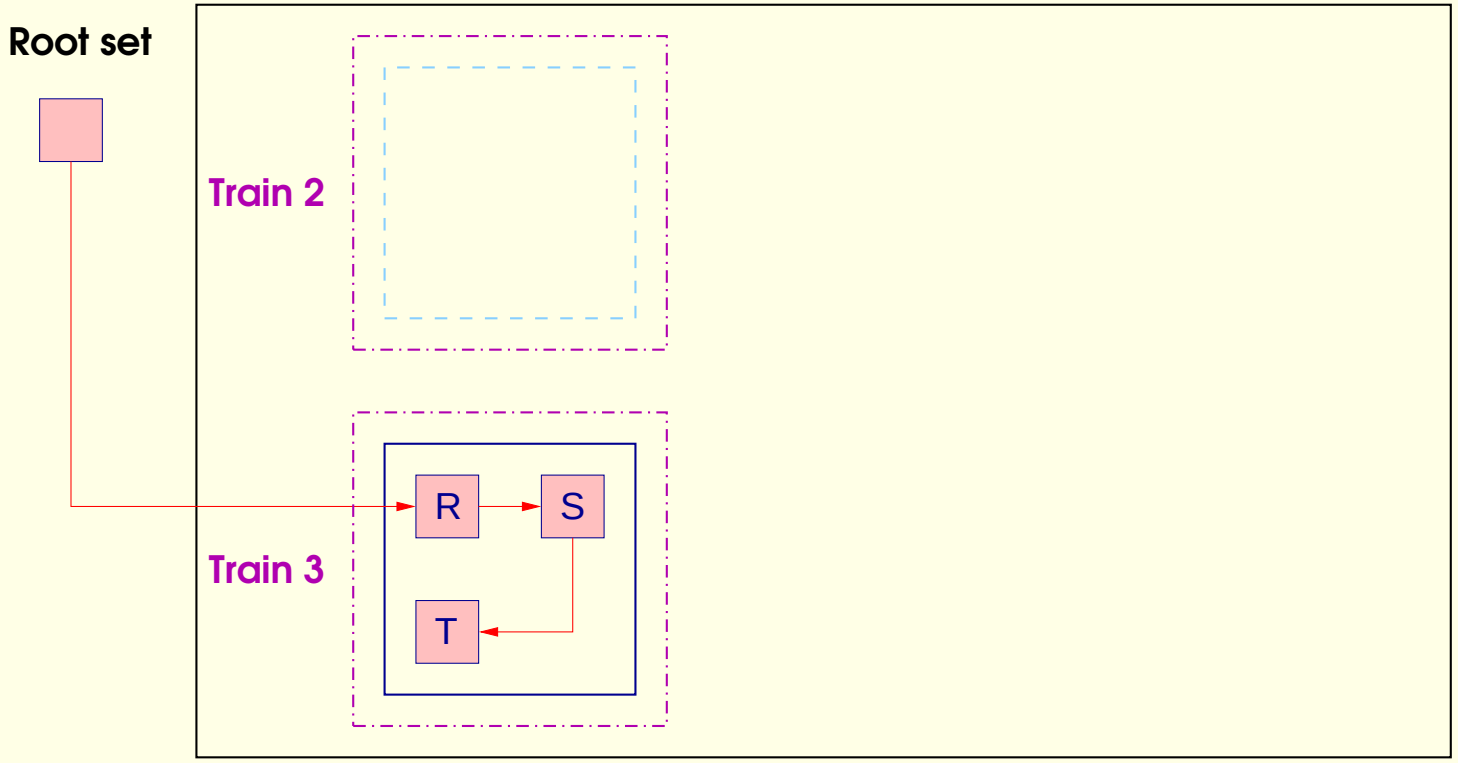
Prügikoristus

Rongi algoritm — seis pärast viiendat koristust:



Prügikoristus

Rongi algoritm — seis pärast kuuendat koristust:



Prügikoristus

Rongi algoritm — kokkuvõte:

+ alates hetkest mil terve struktuur on ühes rongis, on ta koheselt vabastatav, kui talle pole ühtegi välist viita;
+ igal koristusel kopeeritavate andmete hulk on piiratud ühe vaguni täiega;
+ kopeeritavad objektid klasterdatakse ja paigutakse kompaktselt ühte rongi;
– suhteliselt keeruline;
– meelespeadele kuluv mälu hulk on suhtelist suur.

Prügikoristus

Põlvkonniti prügikoristuse eelised:

- väga edukas paljude rakenduste korral;
- lühendab prügikoristuse pause interaktiivseteks rakendusteks talutavale tasemele;
- heade lokaalsusomadustega;
- reeglina vähendab prügikoristusele kuluvat koguaega.

Prügikoristus

Põlvkonniti prügikoristuse puudused:

- halvimal juhul on lihtsatest meetodidest kulukam;
- objektid ei tarvitse surra piisavalt kiiresti;
- rakendused võivad ”takistuda” kirjutustõketesse;
- liiga paljude viitade korral vanadest objektidest noortesse või väga sügava magasini korral võib pauside kestus pikeneda.