Čiščenje pomnilnika

Kaj bom predstavil?

- Kaj je čiščenje pomnilnika
- Kratka zgodovina
- Najpogosteje uporabljeni algoritmi
- Nekaj konkretnih primerov iz programskih jezikov

Opis problema

- Upravljanje pomnilnika s
  - Skladom
  - Kopico

Zgodovina

- Samo statična alokacija
- Kasneje nalaganje na sklad
- Leta 1959 prvi garbage collector v LISPu
- Zaradi počasnosti ne postane popularen do devetdesetih let z prihodom Jave
- Danes v skoraj vseh jezikih

Zakaj garbage collection?

- Večja varnost
- Dober garbage collector je bolj učinkovit kot ročno delo s pomnilnikom

Ročno upravljanje s pomnilnikom

Ročno uporavljanje s pomnilnikom

- Danes redko, srečamo ga v C, C++ in nekaterih jezikih za sistemsko programiranje.
- Uporabno za pisanje gonilniko, operacijskih sistemov in za vgrajene sisteme

Ročno uporavljanje s pomnilnikom

- V uporabi dva načina
  - `malloc` in `free`
  - `smart pointerji`

Ročno uporavljanje s pomnilnikom

- Prednosti
  - predvidljiv čas izvajanja
  - predvidljiva poraba pomnilnika
  - v nekaterih primerih hitrejše (ne nujno!)

Ročno uporavljanje s pomnilnikom

- Slabosti
  - Memory leak
  - Dangling pointer
  - Double free

Reference counting

Reference counting

- Ideja
  - Poleg vsakega objekta hranimo število referenc nanj
  - Če je število referenc enako 0, ga recikliramo

Reference counting

- Prednosti
  - Preprosta implementacija
  - Predvidljivo delovanje

Reference counting

- Slabosti
  - Ciklične reference
  - Dodatno delo
  - Števci fiksne dolžine
  - Dolgotrajno čiščenje velikih objektov

Tracing garbage collection

Tracing garbage collection

- Najbolj sodobne metode
- Veliko različnih variant
- Za nekatere edini ‘pravi’ garbage collection

Tracing garbage collection

- Osnovna ideja
  - Sledimo referencam iz root set-a
  - Objekte, ki niso dosegljivi počistimo

Tracing garbage collection

- Kdaj algoritem teče?
  - Stop-the-world
  - Incremental
  - Concurrent
  - Parallel

Tracing garbage collection

- Način iskanja referenc
  - Precise
  - Conservative

Tracing garbage collection

- Non-moving
- Moving

Tracing garbage collection

- Operacije
  - Mark
    - $O(|Liveset|)$
  - Sweep
    - $O(|Heap|)$
  - Compact
    - $O(|Liveset|)$
  - Copy
    - $O(|Liveset|)$

Tracing garbage collection

- Konkretni pogosto uporabljeni algoritmi
  - Mark-sweep
  - Mark-sweep-compact
  - Mark-compact
  - Mark-don't sweep

Konkretni primeri

Java

- Odvisno od JVM-ja
- OpenJDK in Oracle:
  - Generational garbage collection
    - Na mladih objektih stop-the-world copying
    - Na starih concurrent mark-sweep

Python

- Reference counting z detekcijo ciklov
  - Šteje razliko med številom alokacij in dealokacij
  - Ko to preseže neko vrednost, se požene algoritem za detekcijo ciklov

C++11

- `unique pointer` - počisti se, ko gre iz scopa, dovoljuje samo eno referenco na en objekt.
- `shared pointer` - reference counting. Ciklom se izogne z `weak pointerji`.

Haskell

- Parallel generational garbage collection
  - Nespremenljivi (immutable) objekti olajšajo delo z generacijami
  - Collector teče na svoji niti

Mercury

- Deklarativen jezik, ki spominja na mešanico Prologa in Haskella
- Compile-time garbage collection
  - Zanimviost, samo v Mercuryju
  - Problem upravljanja spomina se reši že med prevajanjem