Prügikoristus

Prügikoristus

- *Prügikoristus* (garbage collection) vabastab automaatselt kuhjaobjektid, mida programm ei saa enam kasutada.
- Koosneb kahest osast:
  - elusate objektide ja prügi eristamisest (garbage detection);
  - prügi vabastamisest (garbage reclamation).
- *Elusus* (liveness) on globaalne semantiline omadus, mida üldjuhul pole võimalik täpselt kindlaks määrata.
- Prügikoristus kasutab ligikaudset kriteeriumit: objekt on elus, kui ta on ”juurtest” (root set) lähtudes kättesaadav.

Prügikoristus

- ”Mark-scan” prügikoristus toimub kahes faasis:
  - esimese faasis, lähtudes juurtest, märgendatakse kõik kättesaadavad objektid;
  - teises faasis teostatakse kuhja täisläbivaatus ning märgendamata objektid vabastatakse.

```
void gc (void) {
  foreach x ∈ Roots do
    mark (x);
  end;
  collect ();
}
```

Prügikoristus

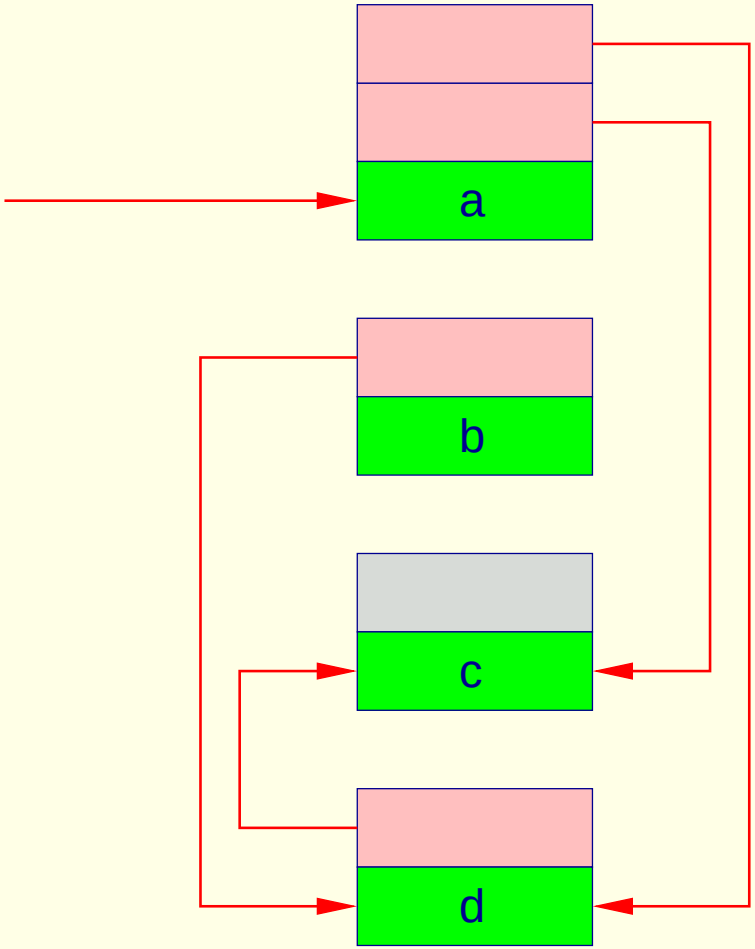
- Protseduur `mark()` märgendab etteantud tipu ning seejärel rekursiivselt kõik kättesaadavad tipud.

```
void mark (ref x) {
  if (x→mark == 0) {
    x→mark = 1;
    foreach y ∈ sons(x) do
      mark (y);
    end;
  }
}
```

- Rekursioon lõpeb, kui tipp on juba märgendatud või kui antud tipp sisaldab ainult baasväärtusi.

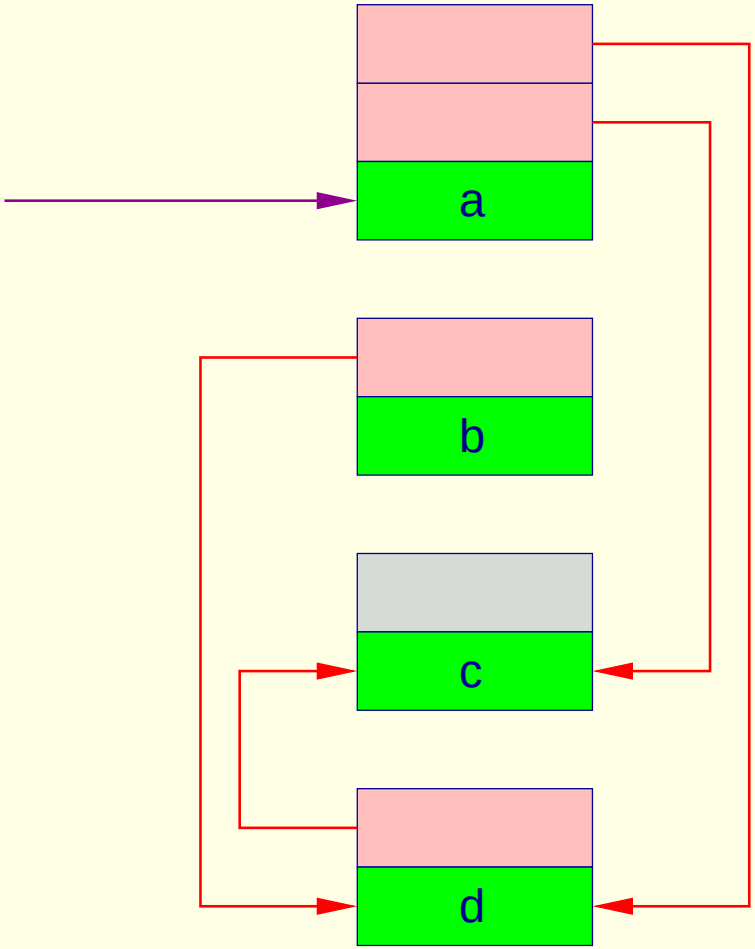
Prügikoristus

Rekursiivne märkimine:



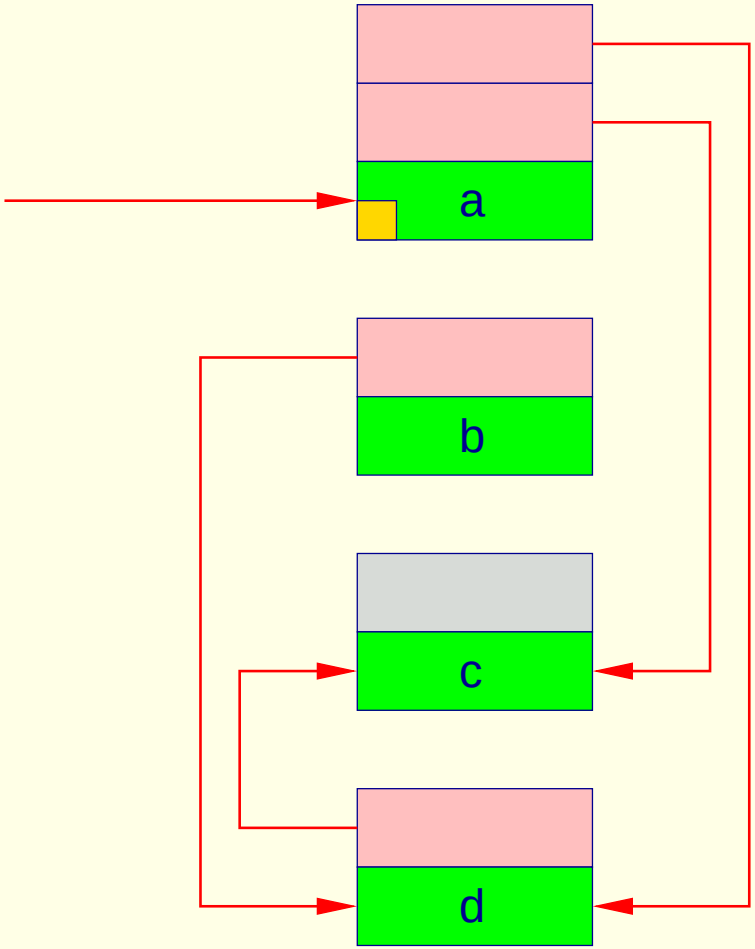
Prügikoristus

Rekursiivne märkimine:



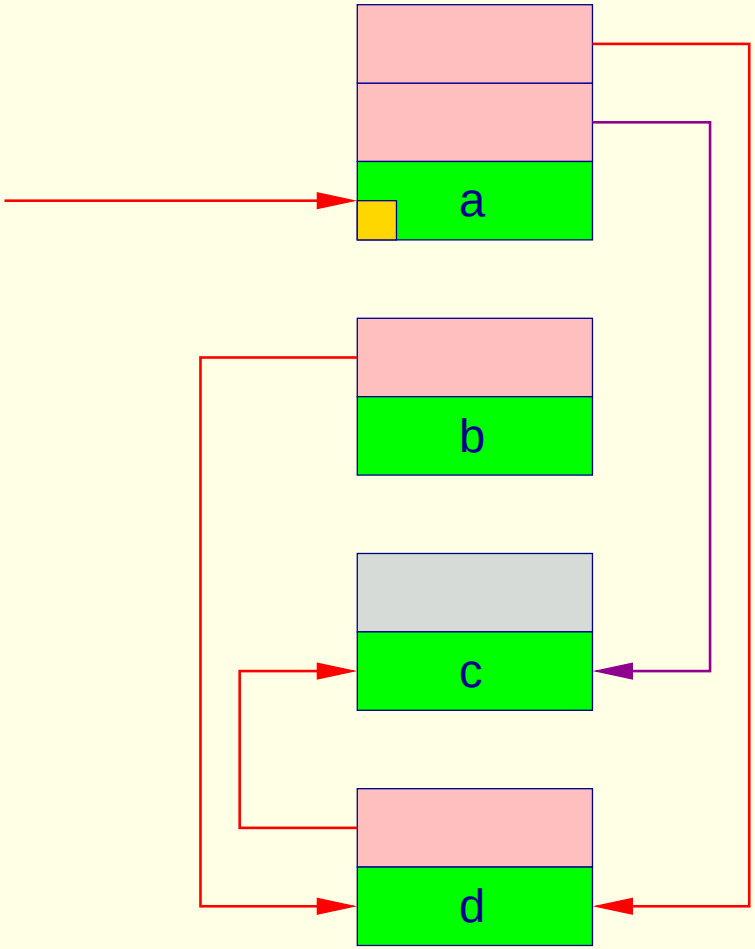
Prügikoristus

Rekursiivne märkimine:



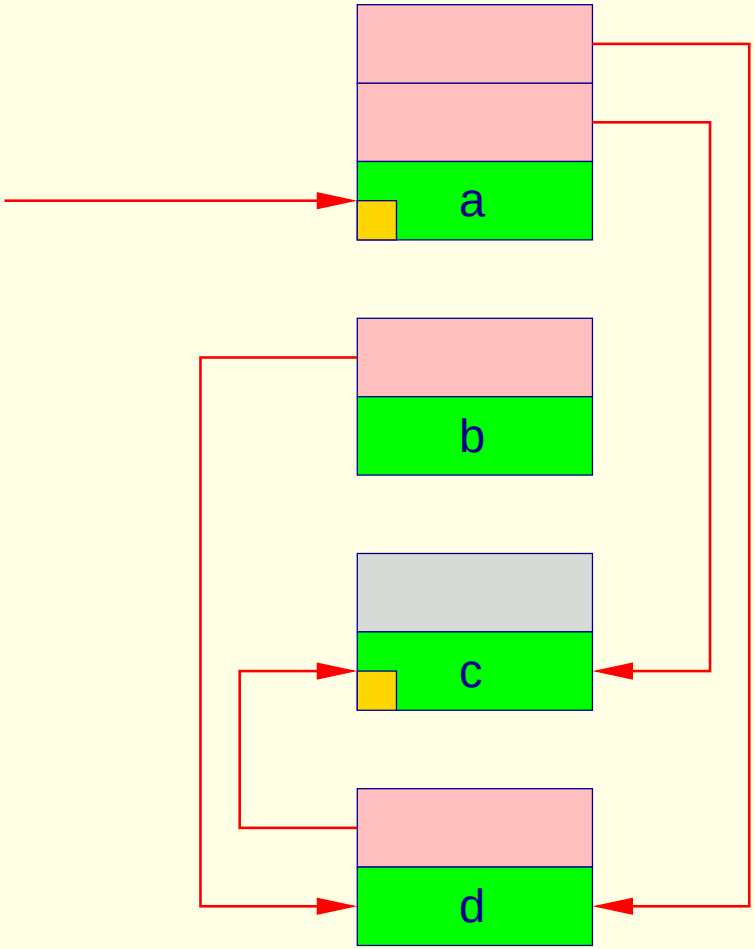
Prügikoristus

Rekursiivne märkimine:



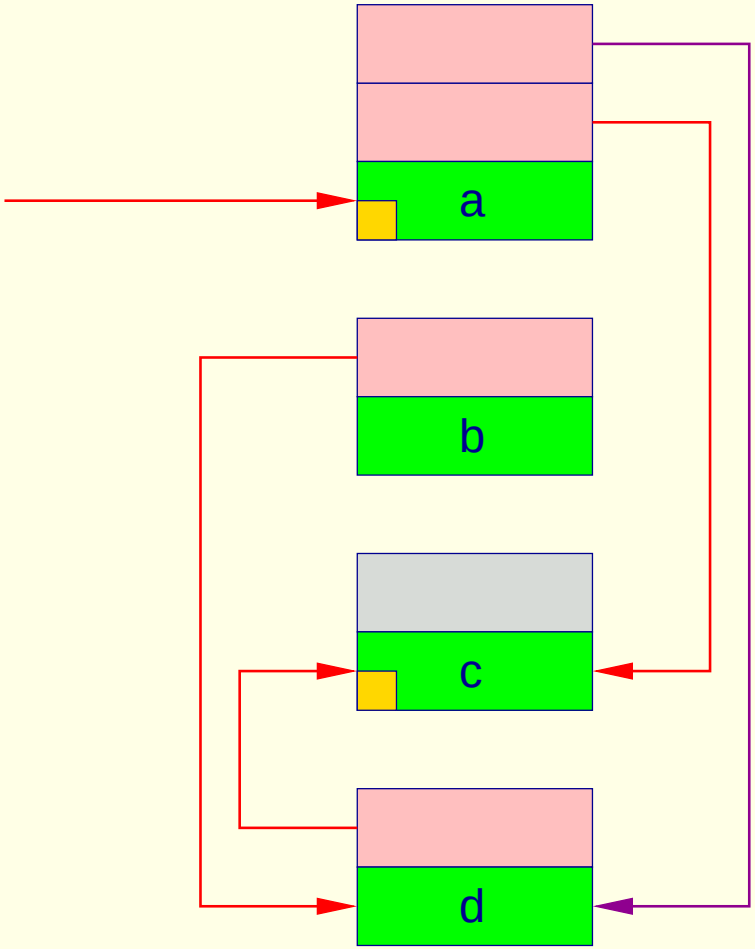
Prügikoristus

Rekursiivne märkimine:



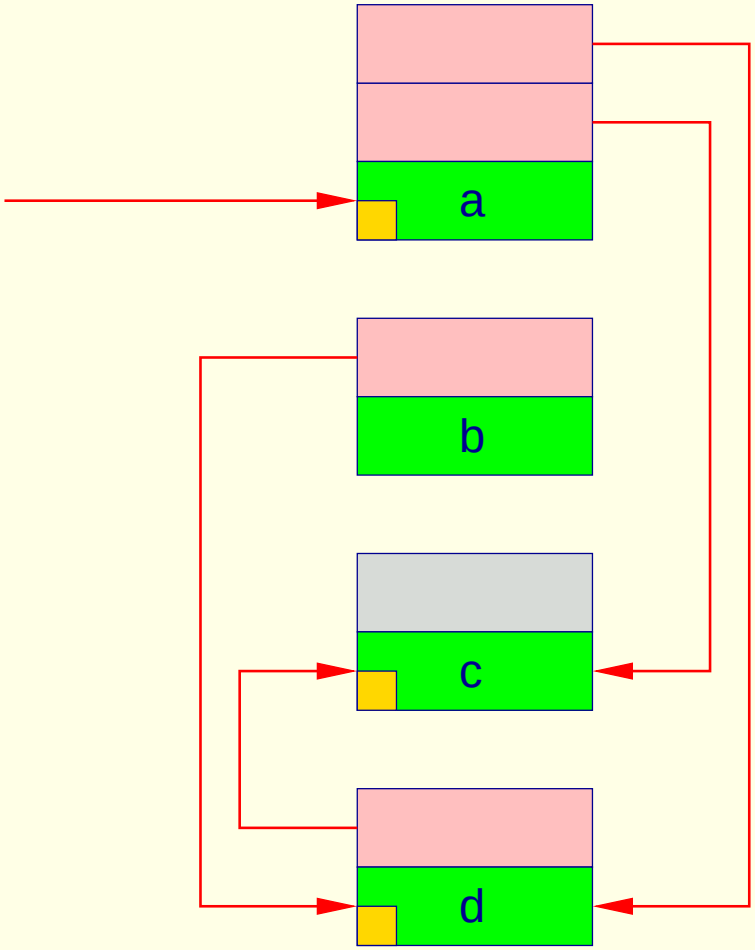
Prügikoristus

Rekursiivne märkimine:



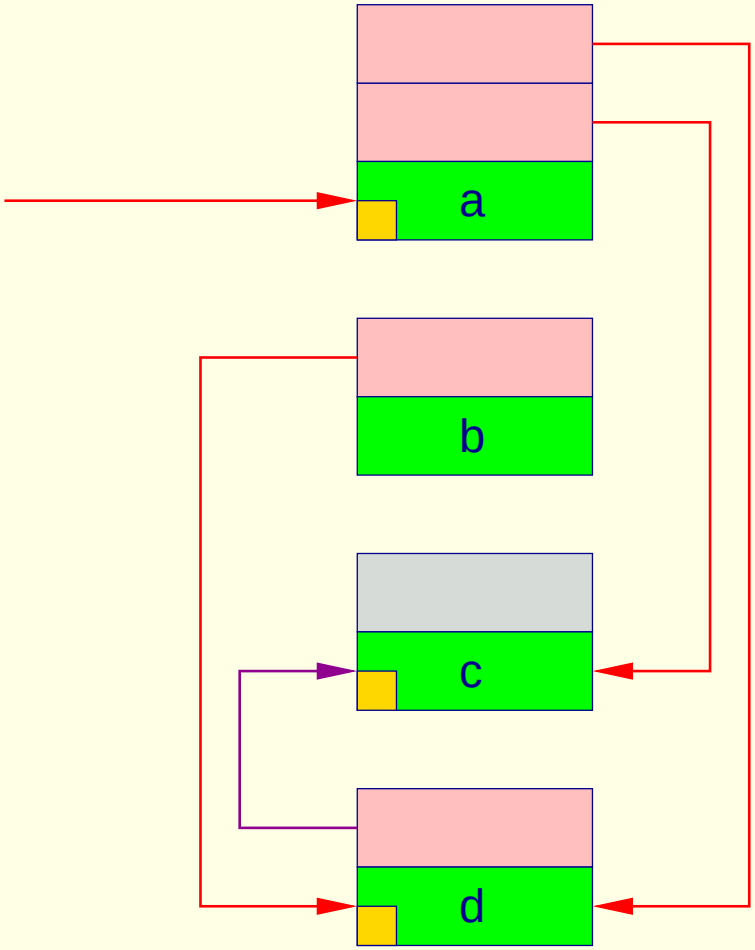
Prügikoristus

Rekursiivne märkimine:



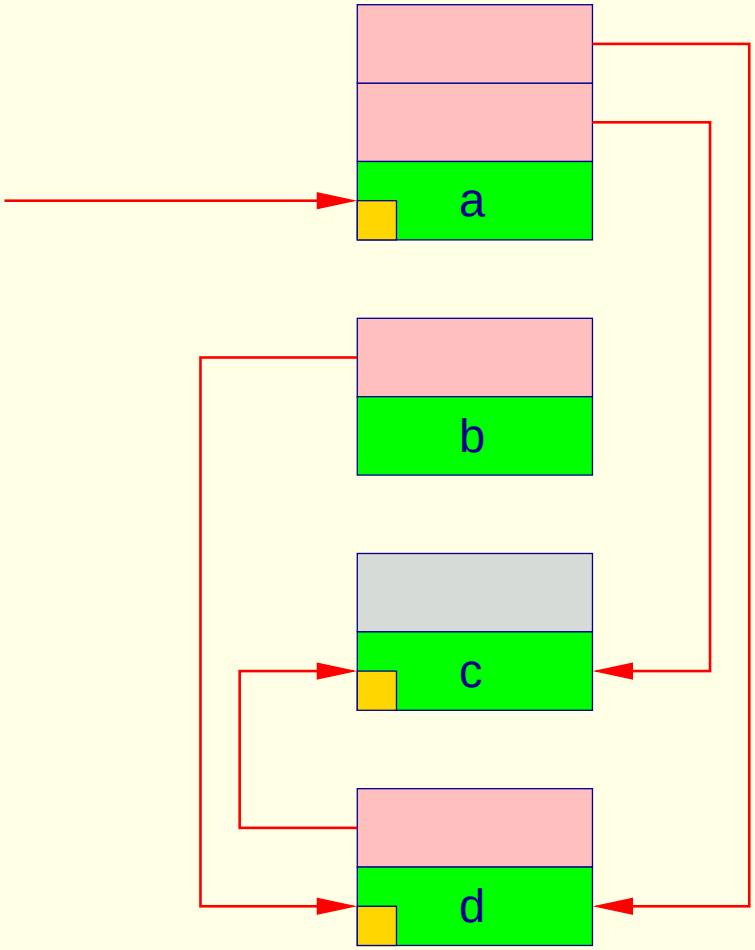
Prügikoristus

Rekursiivne märkimine:



Prügikoristus

Rekursiivne märkimine:



Prügikoristus

- Protseduur `collect()` vaatab läbi kõik kuhjaobjektid ning lisab märgendamata objektid vabade objektide listi.

```
void collect (void) {
  freelist = NIL;
  foreach x ∈ objects() do
    if (x→mark == 0) {
      x→next = freelist;
      freelist = x;
    }
    else x→mark = 0;
  end;
}
```

Prügikoristus

”Mark-scan” prügikoristuse puudused:

- Objektide märkimine toimub rekursiivselt.
  - Rekursioonimagasin kasvab halvimal juhul linearselt kuhja suurusega!!
  - Võimalik lahendus: Schorr-White’i ”viitade pööramise” algoritm.
- Elusad objektid on kuhjas vabade mälupiirkondadega segamini.
  - Mälu fragmenteerumine.
  - Võimalik lahendus: ”mark-compact” prügikoristus.

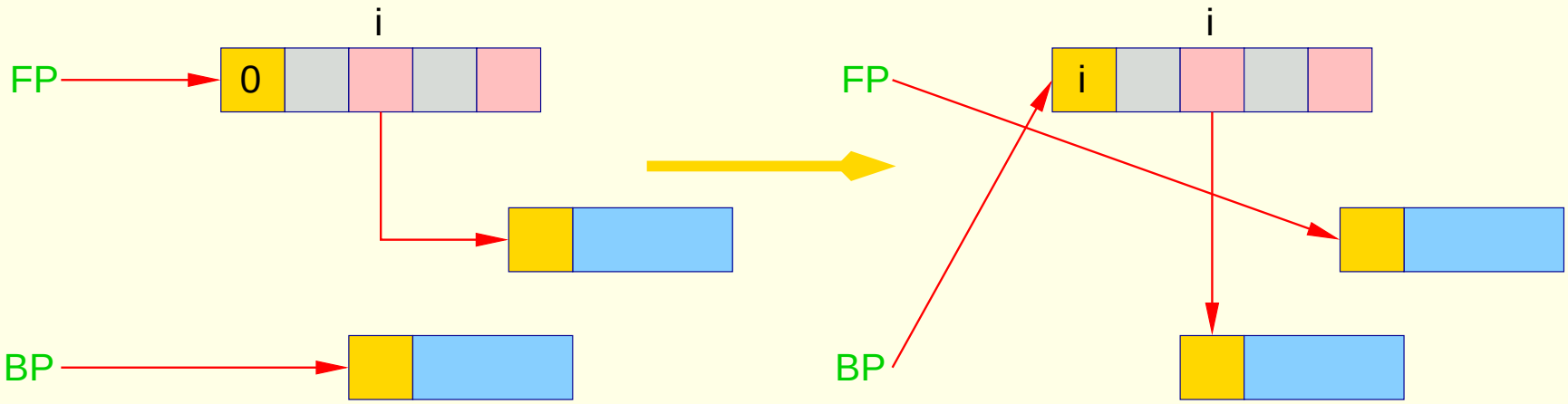
Prügikoristus

Schorr-White'i "viitade pööramise" (pointer reversal) algoritm:

```
void mark (ref x) {
  FP = x; BP = NIL;
  while (FP→mark ≠ -1 || BP ≠ NIL) {
    if (FP→mark == 0) {
      FP→mark = i = nextidx(FP);
      if (i ≠ -1) {
        tmp = FP; FP = tmp[i];
        tmp[i] = BP; BP = tmp;
      }
    } else { // FP→mark == -1
      ...
    }
  }
}
```

Prügikoristus

Schorr-White'i "viitade pööramise" algoritm:



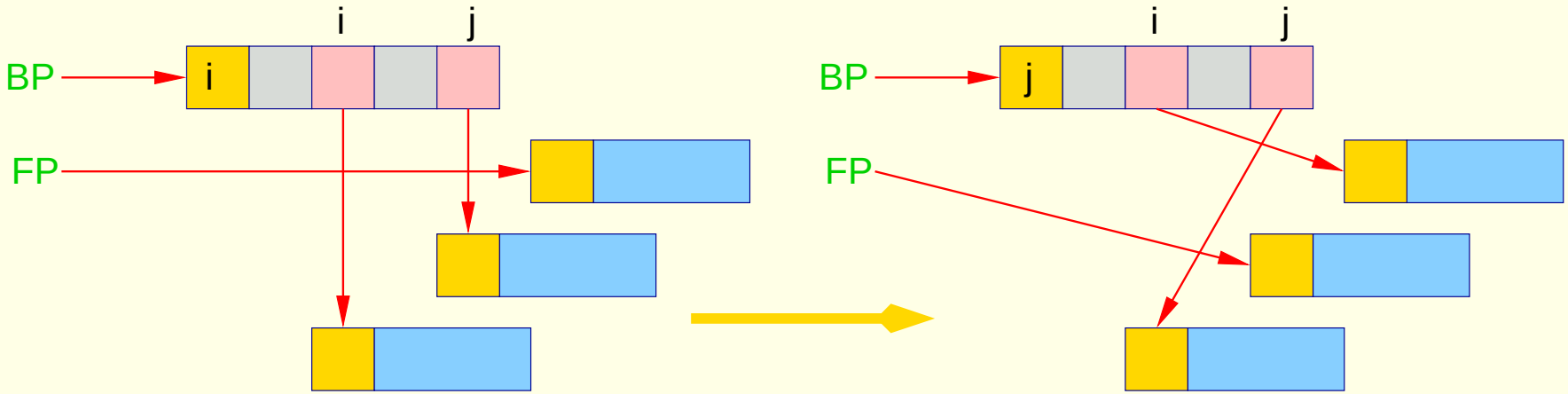
Prügikoristus

Schorr-White'i "viitade pööramise" algoritm:

```
...
} else { // FP→mark == -1
  i = nextidx(BP);
  if (i ≠ -1) {
    tmp = FP; FP = BP[i]; BP[i] = BP[BP→mark];
    BP[BP→mark] = tmp; BP→mark = i;
  } else {
    tmp = FP; FP = BP; BP = FP[FP→mark];
    FP[FP→mark] = tmp; FP→mark = i;
  }
}
...
```

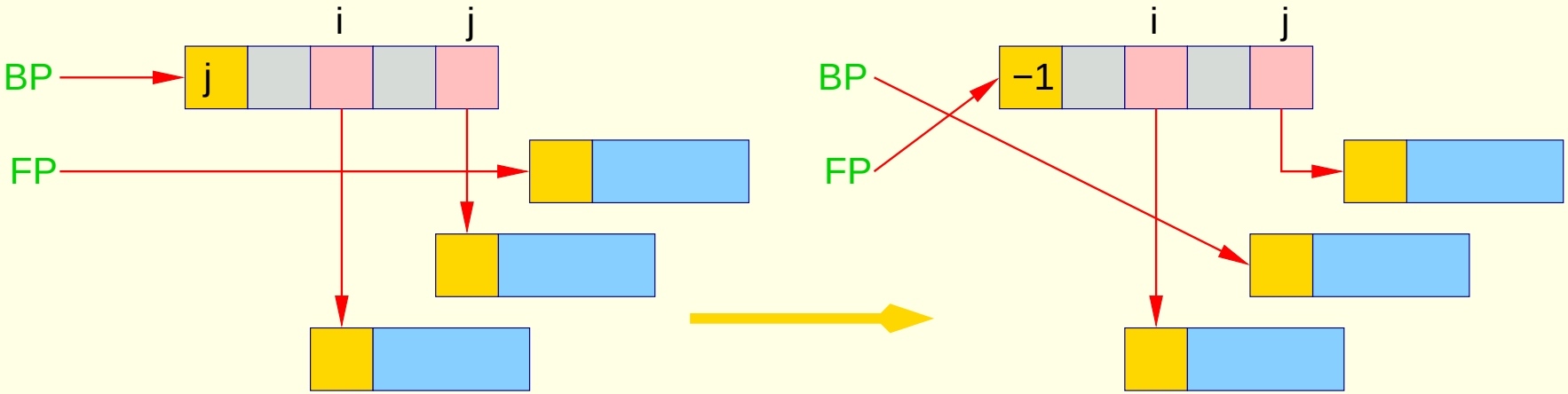
Prügikoristus

Schorr-White'i "viitade pööramise" algoritm:



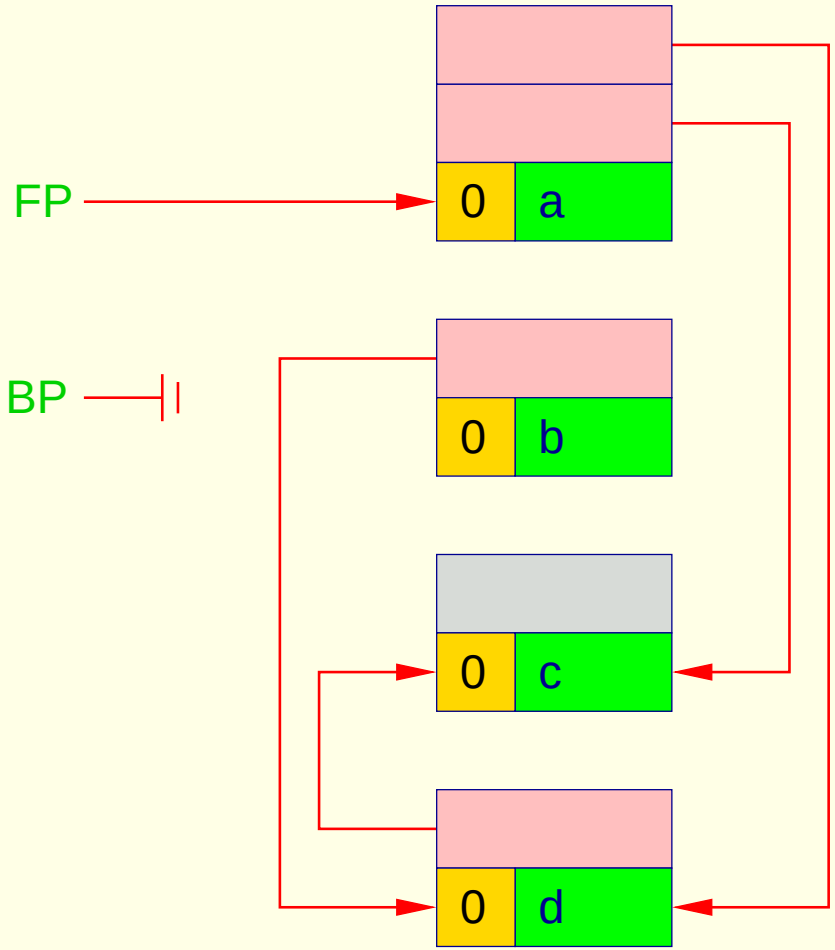
Prügikoristus

Schorr-White'i "viitade pööramise" algoritm:



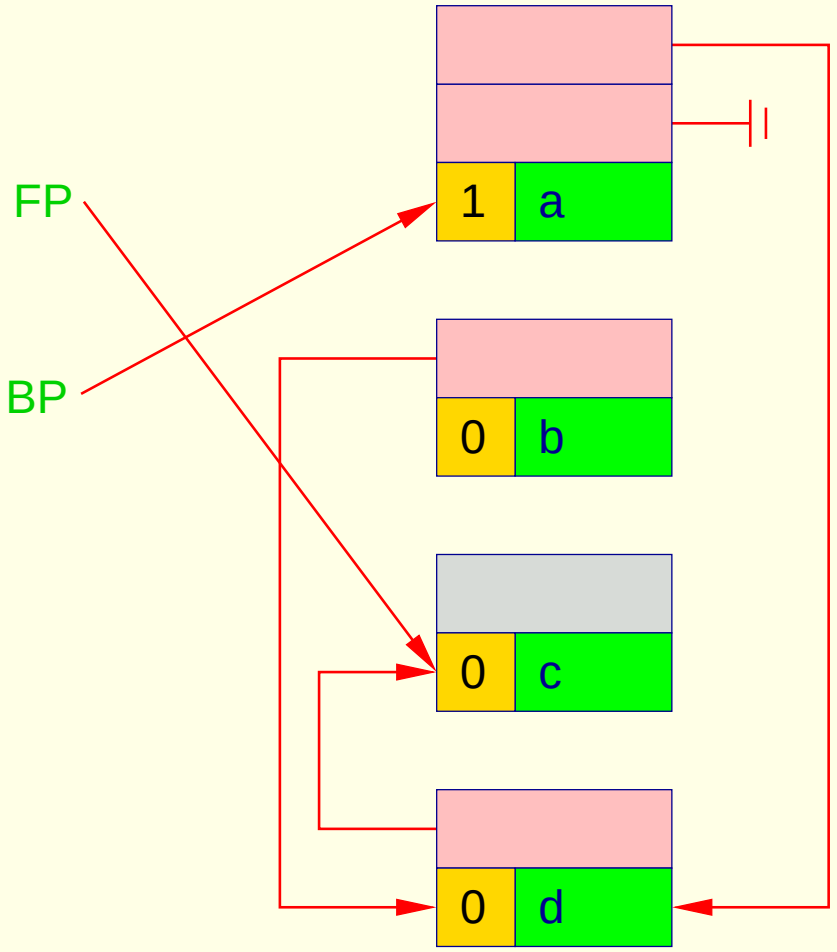
Prügikoristus

Näide:



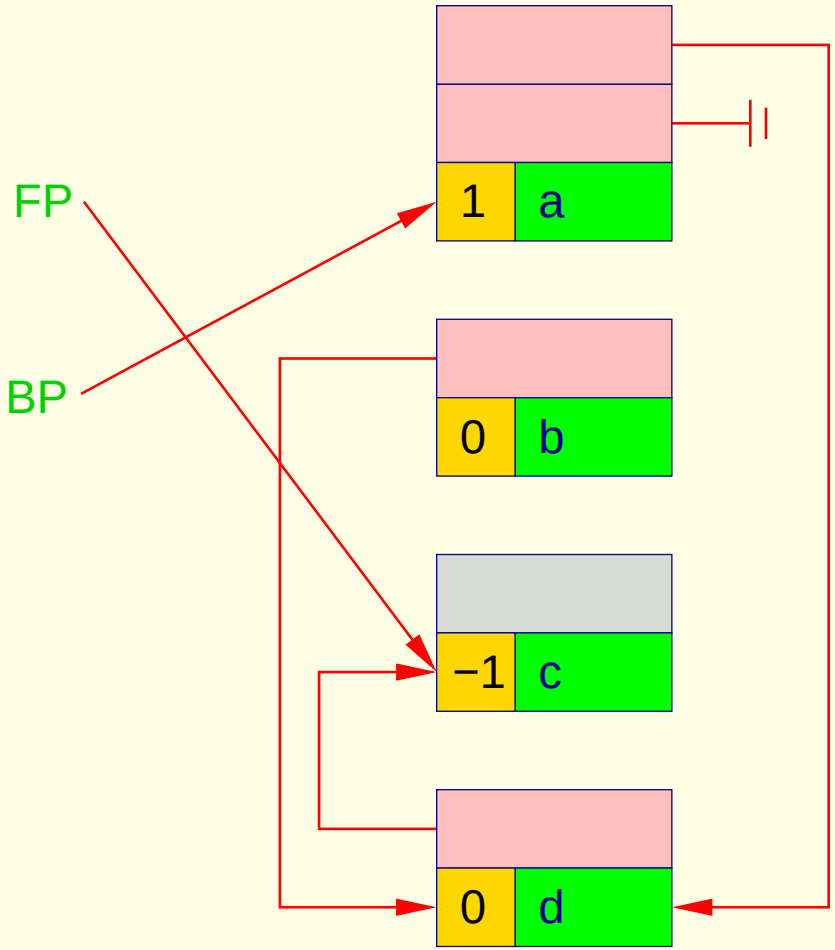
Prügikoristus

Näide:



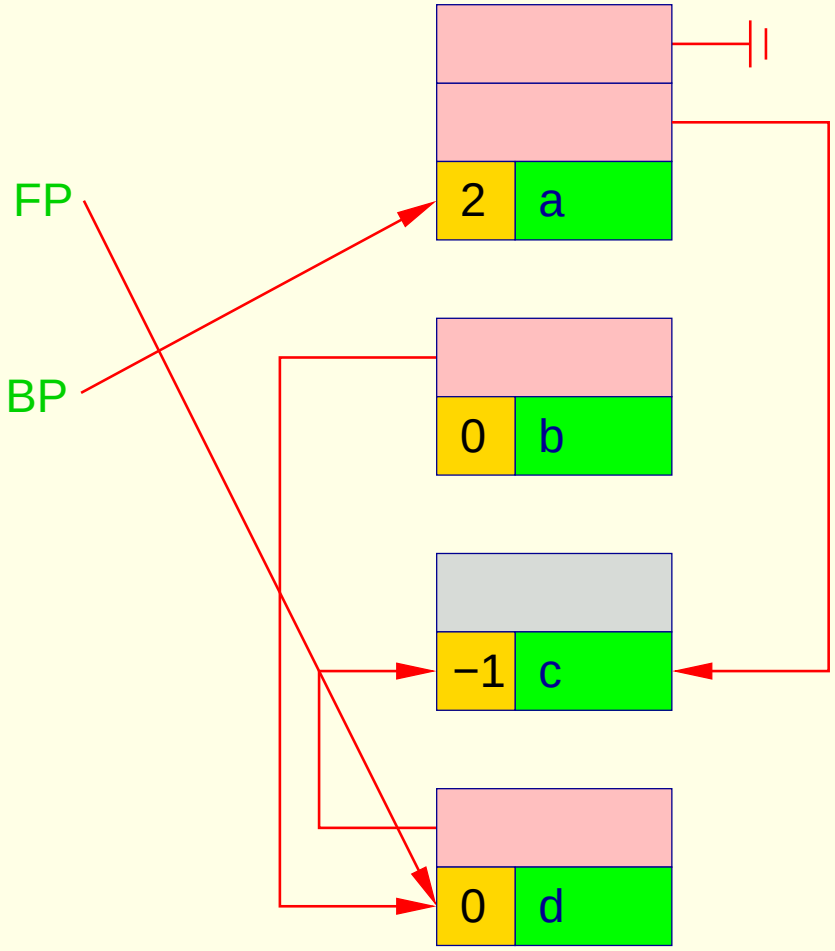
Prügikoristus

Näide:



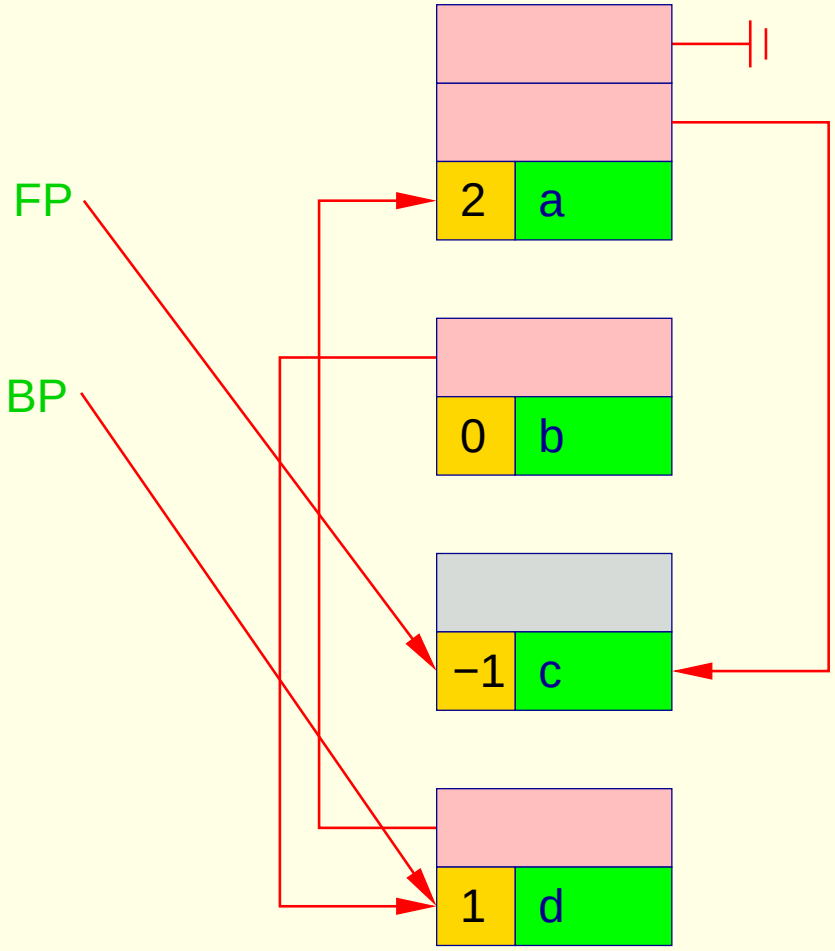
Prügikoristus

Näide:



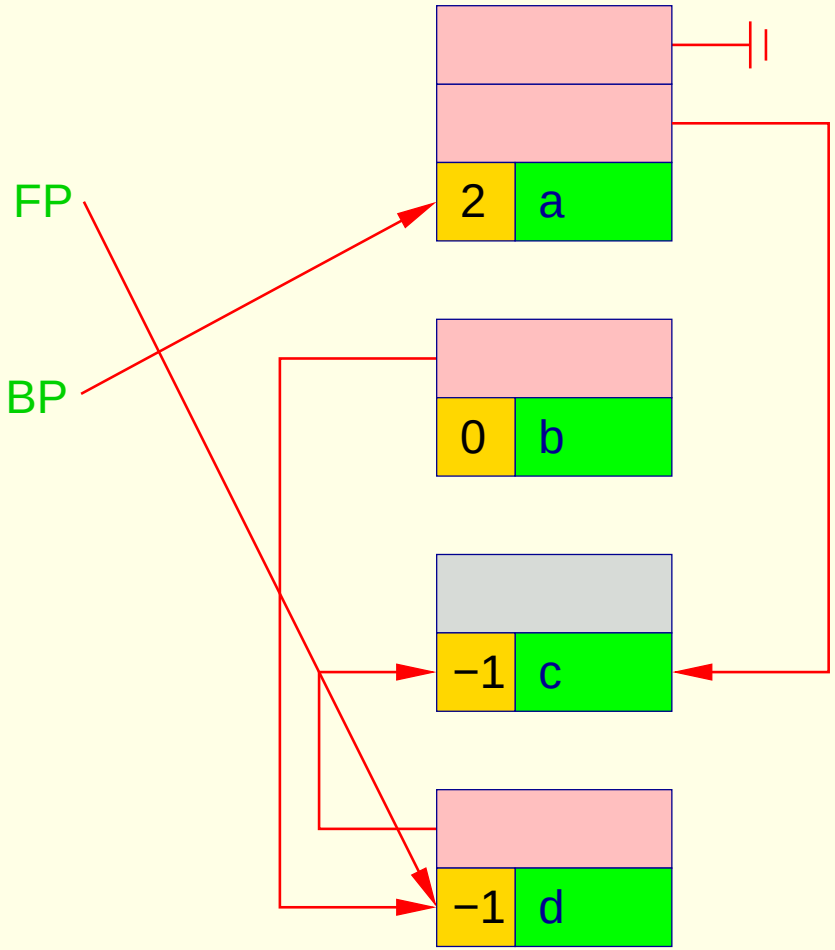
Prügikoristus

Näide:



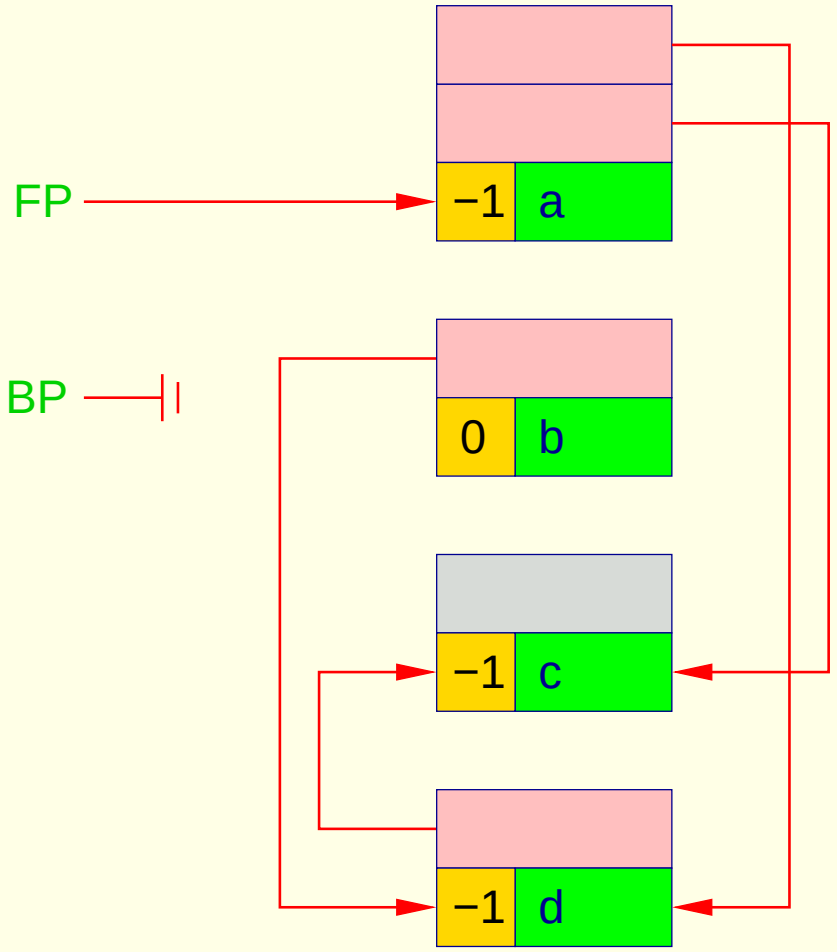
Prügikoristus

Näide:



Prügikoristus

Näide:



Prügikoristus

- ”Mark-compact” prügikoristus toimub kolmes faasis:
  - esimese faasis, lähtudes juurtest, märgendatakse kõik kättesaadavad objektid (analoogselt ”mark-scan” prügikoristusega);
  - teises faasis teostatakse kuhja täisläbivaatus ning arvutatakse märgendatud objektide uued aadressid;
  - kolmandas faasis nihutatakse märgendatud objektide uude asukohta ning viitadele antakse uued väärtused.

\+ Prügikoristuse tulemusena paikneb kogu vaba mälu kompaktselt kuhja lõpus.

– On suhteliselt aeglane, kuna kuhjaobjekte tuleb läbida palju kordi.