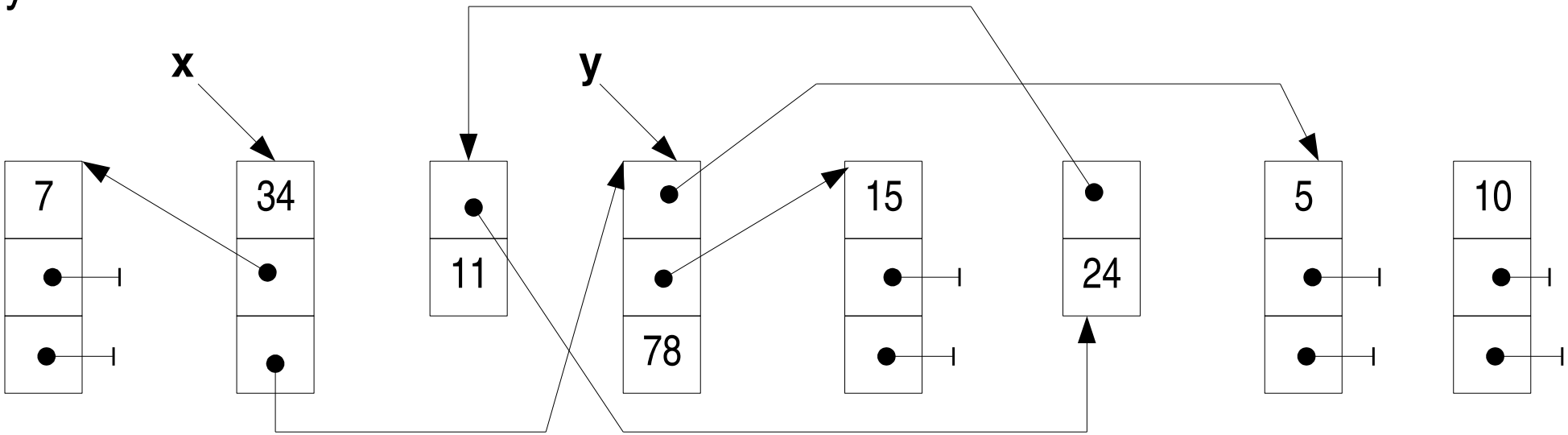
Odśmiecanie sterty (garbage collection)

- Zagadnienia do analizy:
  - Kiedy należy uruchamiać odśmiecanie ?
  - Jak określić nieużywane rekordy (jak powinny wyglądać rekordy na stercie?)
  - fragmentacja sterty
  - wykorzystanie zwolnionych bloków
- Pojęcia rekordu "żywego" i "osiągalnego"
- Przykładowa sterta



Mark-And-Sweep

- Dwie fazy: oznaczanie (mark) i zamiatanie (sweep)
- Algorytm oznaczania (przeglądanie wgłąb)

```
DFS (a):
    Jeżeli a jest wskaźnikiem na stertę:
        Jeżeli rekord a nie jest jeszcze oznaczony:
            Oznacz a
            Dla każdego pola f w rekordzie a:   DFS ( f )
```

- Algorytm zamiatania - wszystkie zmienne wskazujące na stertę są korzeniami

```
Dla każdego korzenia v: DFS(v);     p = pierwszy adres na stercie
Dopóki p < ostatni adres na stercie:
    Jeżeli rekord p jest oznaczony:   Usuń oznaczenie z p
    w przeciwnym wypadku:
        f = pierwsze pole w p
        p.f = lista_wolnych
        lista_wolnych = p
    p = p + rozmiar rekordu p
```

Liczniki referencji

- W każdym rekordzie składujemy informację o ilości referencji na ten rekord
- Za każdym razem, kiedy wykonujemy przypisanie, uzupełniamy liczniki (rekurencyjnie!)
- Dwa podejścia do rekurencyjnego uzupełniania liczników:
  - W momencie wkładania rekordu na listę zwolnionych bloków
  - W momencie ponownej alokacji bloku
- Wady
  - Liczniki referencji NIE ZWALNIAJĄ pętli
  - Duży koszt (przykład sekwencji intrukcji podczas podstawienia)
- Ulepszenia
  - Optymalizacja kodu (propagacja stałych)
  - Przerywanie pętli przez programistę
  - Okresowe wywoływanie "mark-and-sweep"

Odśmiecanie kopiujące

- Sterta jest podzielona na dwa obszary: from-space (FS) i to-space (TS)
- **next** - pierwszy wolny adres (do alokacji)
- limit - maksymalny adres, po przekroczeniu którego rozpoczynamy kopiowanie
- Funkcja kopiująca:

Forward (**p**):

Jeżeli **p** wskazuje na **FS**:

f = pierwsze pole **p**

Jeżeli **f** wskazuje na **TS**: return **f**

w przeciwnym wypadku:

Dla każdego pola **g** z rekordu **p**: do lokacji **next + g** zapisz wartość **p.g**

p.f = next

next = next + rozmiar rekordu **p**

return **p.f**

w przeciwnym wypadku: return **p**

Odśmiecanie kopiujące c.d.

- Algorytm Cheney'a

next = **scan** = początek **TS**
Dla każdego korzenia **r**: **r** = Forward(**r**)
Dopóki **scan** < **next**:
    Dla każdego pola **f** rekordu wskazywanego przez **scan**:
        scan.f = Forward (**scan.f**)
    scan = **scan** + rozmiar rekordu wskazywanego przez **scan**

- Lokalizacja referencji

Forward(**p**):
  Jeżeli **p** wskazuje na **FS**:
    f = pierwsze pole **p**
    Jeżeli **f** wskazuje na **TS**:
      return **f**
    w p.p: Chase(**p**); return **p.f**
  else return **p**

Chase(**p**):
  powtarzaj:
    q = **next**; **next** = **next** + rozmiar **p**; **r** = NULL
    Dla każdego **f** z rekordu **p**:
      w lokacji **q + f** wpisz **p.f**
      Jeśli **q** wskaz. na **FS** i **q.f.f1** nie wskaz. na **TS**:
        r = **q.f**
    p.f1 = **q**; **p** = **r**
  dopóki **p** nie stanie się NULL

Odśmiecanie inkrementacyjne

- Collector - proces zajmujący się odśmiecaniem
- Mutator - działający program, który zmienia graf wskaźników na stercie
- Rodzaje algorytmów
  - inkrementacyjny - collector wywoływany przez mutator
  - współbieżny - collector i mutator działające naprzemiennie
- Oznaczanie trójkolorowe (tricolor marking)
  - Biały - blok nie odwiedzany przez odśmiecacz (oznaczanie, kopiowanie)
  - Szary - blok odwiedzony, ale jego dzieci jeszcze nie
  - Czarny - blok i jego dzieci odwiedzone

Dopóki są **szare** obiekty:
    Wybierz **szary** blok **p**
    Dla każdego pola **f** z bloku **p**
        Jeżeli **p.f** wskazuje na **biały** blok
            Oznacz **p.f** jako **szary**
    Oznacz **p** jako **czarny**

- Żadne pole w czarnym bloku nie wskazuje na blok biały
- Każdy szary obiekt znajduje się w zbiorze operacyjnym odśmiecacza

Odśmiecanie inkrementacyjne c.d.

- Algorytmy
  - *Dijkstra, Lamport*: Jeżeli program (mutator) zapisuje "biały wskaźnik a" w "czarnym bloku b", koloruje **a** na szaro
  - *Steele*: Jeżeli program (mutator) zapisuje "biały wskaźnik a" w "czarnym bloku b", koloruje **b** na szaro
  - *Boehm, Demers, Shenker*: Strony w pamięci wirtualnej, w których wszystkie bloki są czarne zostają oznaczone jako "tylko do odczytu". Błąd zapisu do takiej strony, koloruje wszystkie jej bloki na szaro.
  - *Baker*: Jeżeli program odczytuje wskaźnik do białego bloku, to koloruje go na szaro (przed pobraniem)
  - *Appel, Ellis, Li*: Jeżeli program pobiera wskaźnik b ze strony pamięci, w której istnieje szary lub biały blok, kolorujemy każdy obiekt na tej stronie na czarno

Różne

- Odśmiecanie generacyjne
- Dodatkowe dane potrzebne do odśmiecania
  - pole wskazujące na deskryptor typu (lista pól i określenie, które z nich są wskaźnikami)
  - opis zmiennych programu (łącznie z tymczasowymi)