MC-202
Gerenciamento de Memória

Rafael C. S. Schouery
rafael@ic.unicamp.br

Universidade Estadual de Campinas

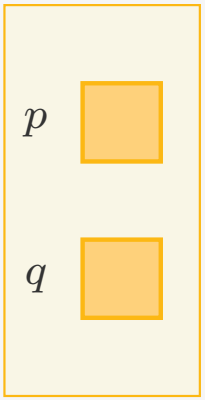
Atualizado em: 2025-08-04 14:58

Como a memória está organizada?

Como a memória está organizada?

Em dois grandes blocos de memória:

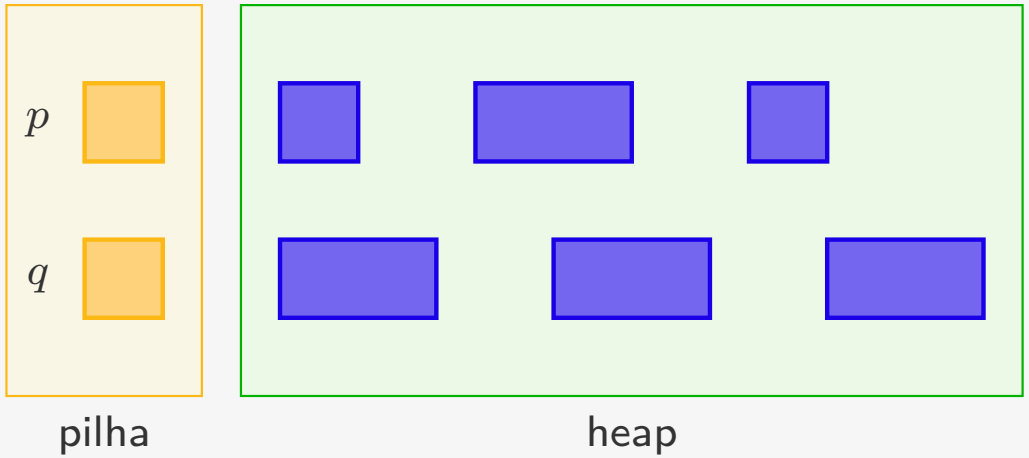
Como a memória está organizada?



Em dois grandes blocos de memória:

- pilha: guardamos as variáveis locais

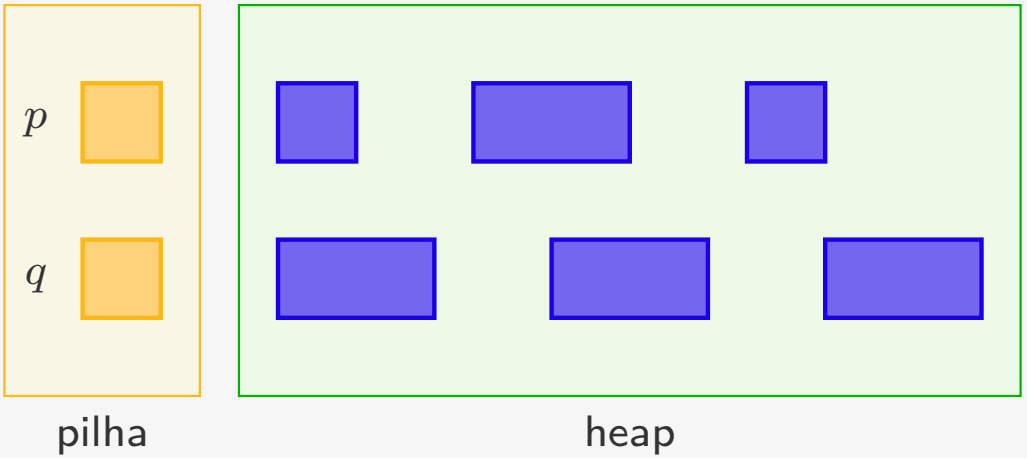
Como a memória está organizada?



Em dois grandes blocos de memória:

- pilha: guardamos as variáveis locais
- heap: criamos nós dinamicamente

Como a memória está organizada?

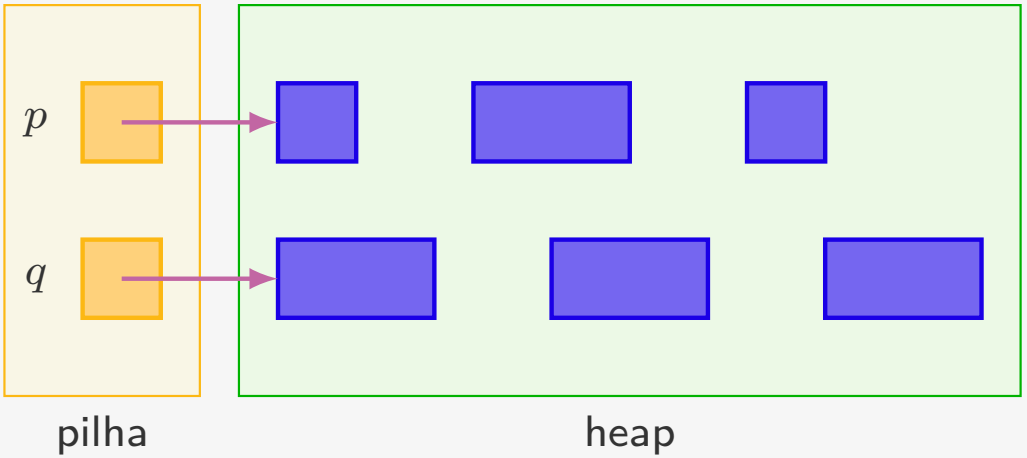


Em dois grandes blocos de memória:

- pilha: guardamos as variáveis locais
- heap: criamos nós dinamicamente

Como acessamos nós no heap?

Como a memória está organizada?



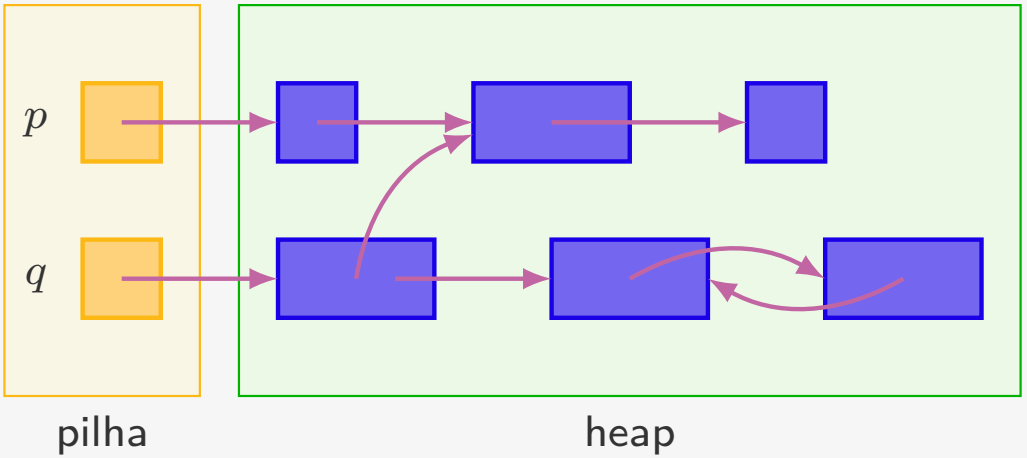
Em dois grandes blocos de memória:

- pilha: guardamos as variáveis locais
- heap: criamos nós dinamicamente

Como acessamos nós no heap?

- diretamente com ponteiros na pilha

Como a memória está organizada?



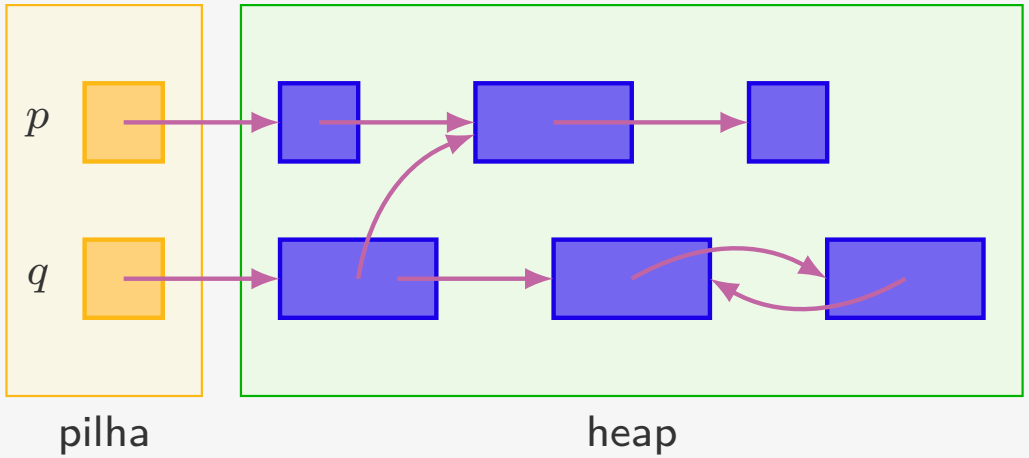
Em dois grandes blocos de memória:

- pilha: guardamos as variáveis locais
- heap: criamos nós dinamicamente

Como acessamos nós no heap?

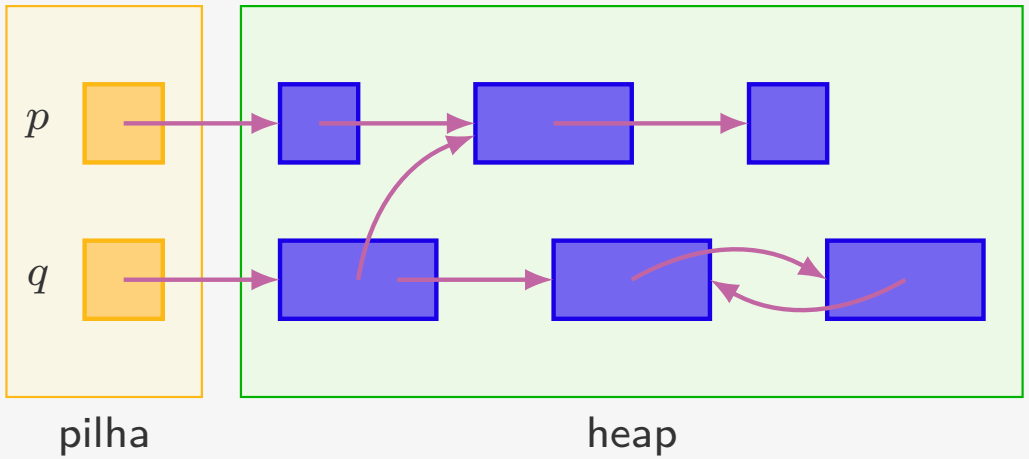
- diretamente com ponteiros na pilha
- indiretamente com ponteiros nos nós

Alocando e liberando nós



Como implementar `malloc` e `free`?

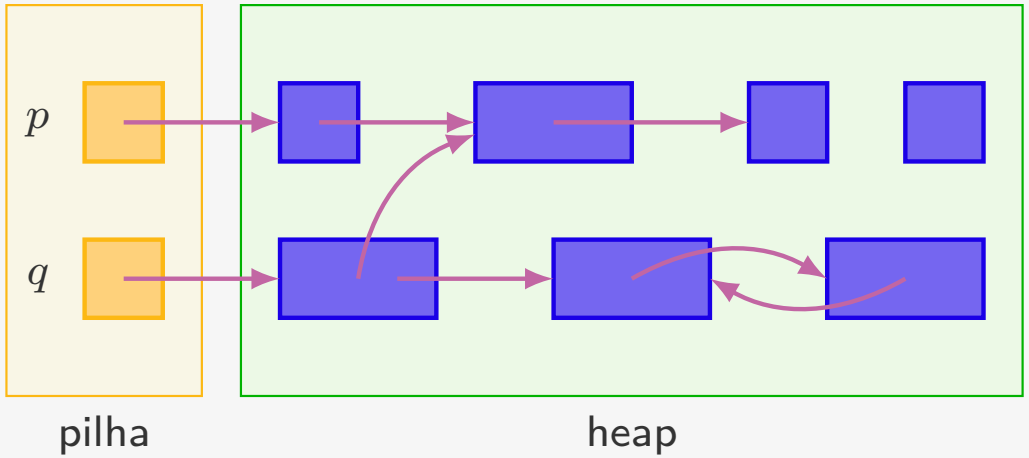
Alocando e liberando nós



Como implementar `malloc` e `free`?

- reservando novos blocos de memória

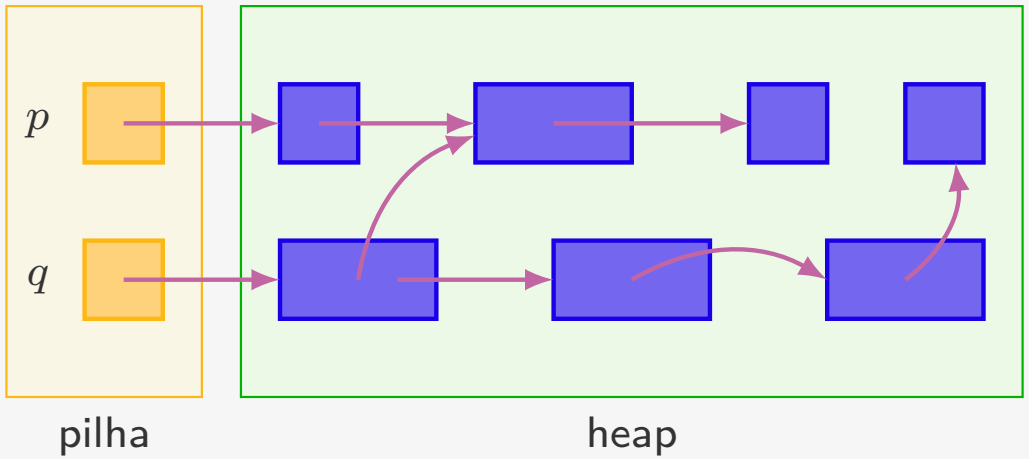
Alocando e liberando nós



Como implementar `malloc` e `free`?

- reservando novos blocos de memória

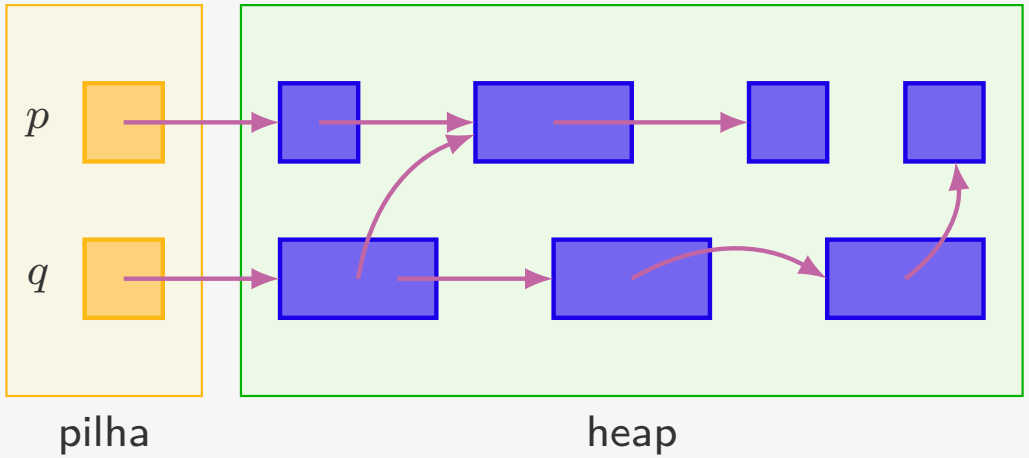
Alocando e liberando nós



Como implementar `malloc` e `free`?

- reservando novos blocos de memória

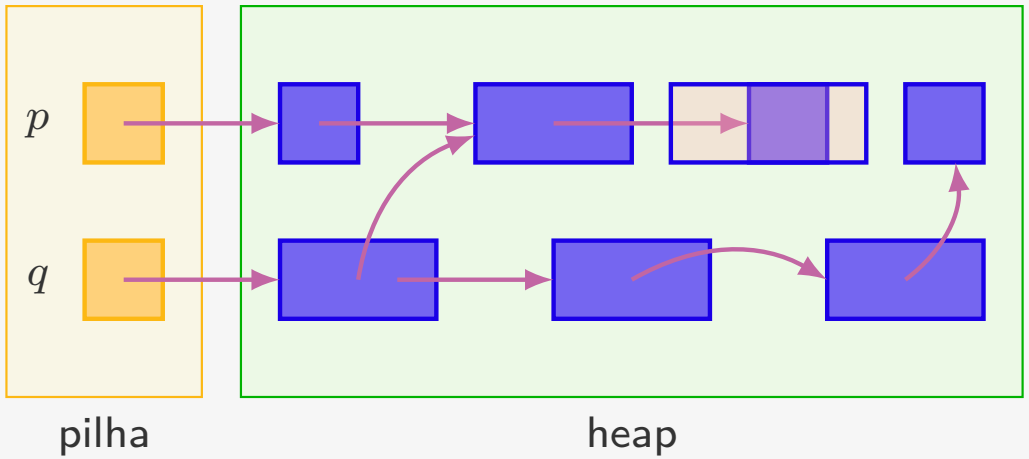
Alocando e liberando nós



Como implementar `malloc` e `free`?

- reservando novos blocos de memória
- sem sobrescrever nós existentes

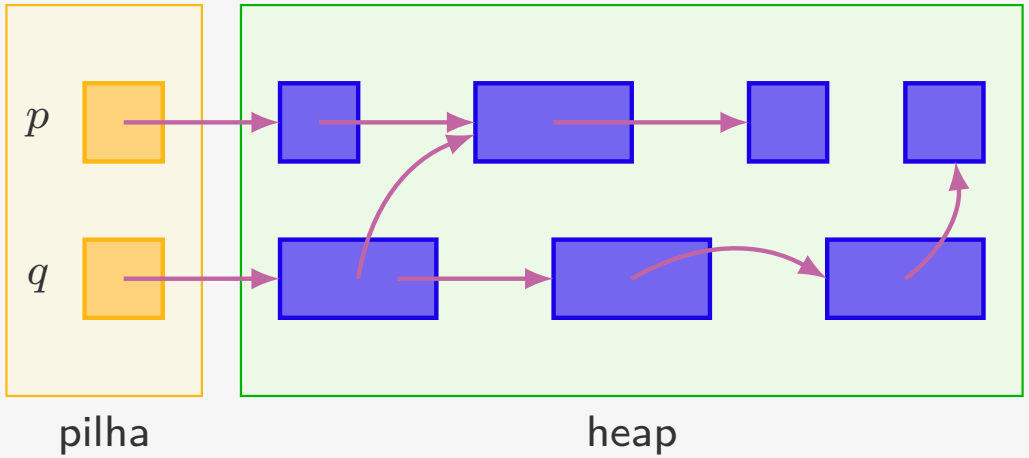
Alocando e liberando nós



Como implementar `malloc` e `free`?

- reservando novos blocos de memória
- sem sobrescrever nós existentes

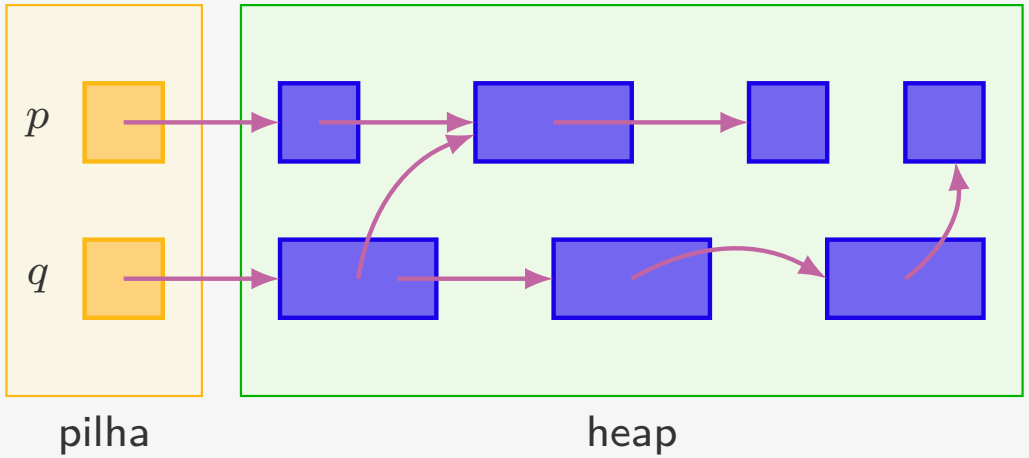
Alocando e liberando nós



Como implementar `malloc` e `free`?

- reservando novos blocos de memória
- sem sobrescrever nós existentes

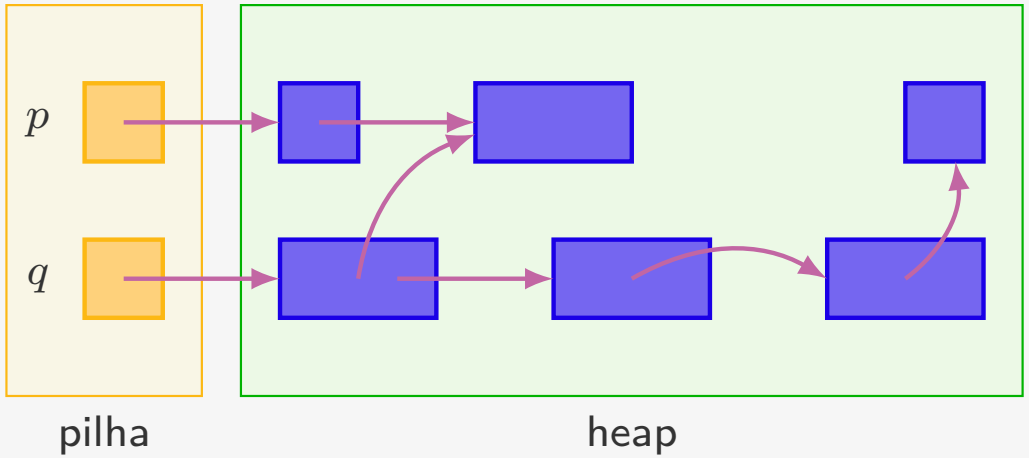
Alocando e liberando nós



Como implementar `malloc` e `free`?

- reservando novos blocos de memória
- sem sobrescrever nós existentes
- liberando e evitando desperdiçar espaço

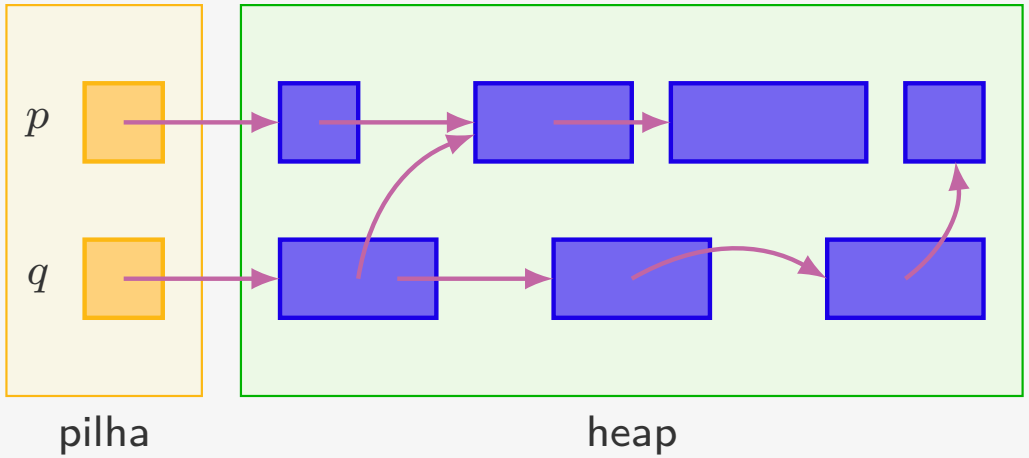
Alocando e liberando nós



Como implementar `malloc` e `free`?

- reservando novos blocos de memória
- sem sobrescrever nós existentes
- liberando e evitando desperdiçar espaço

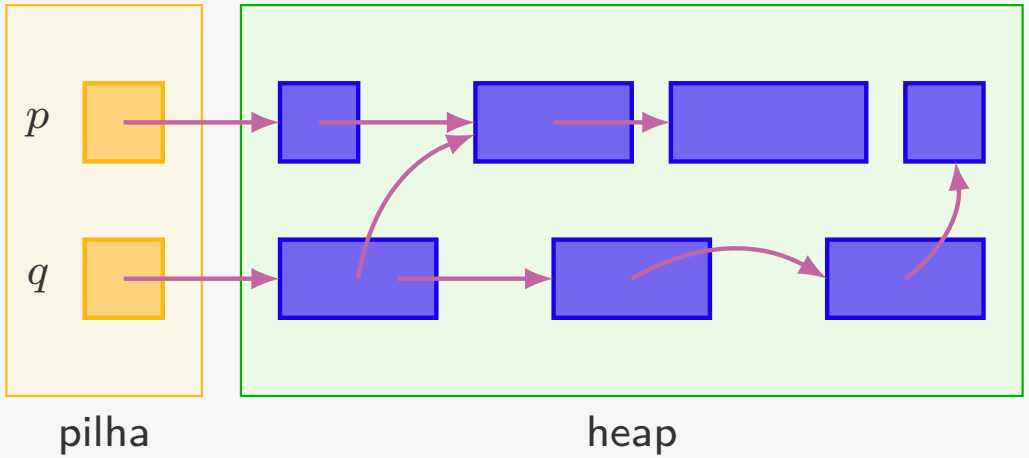
Alocando e liberando nós



Como implementar `malloc` e `free`?

- reservando novos blocos de memória
- sem sobrescrever nós existentes
- liberando e evitando desperdiçar espaço

Alocando e liberando nós



Como implementar `malloc` e `free`?

- reservando novos blocos de memória
- sem sobrescrever nós existentes
- liberando e evitando desperdiçar espaço
- tudo isso eficientemente

Lista de blocos livres

O heap é um espaço contíguo de memória:

Lista de blocos livres

O heap é um espaço contíguo de memória:

Lista de blocos livres

O heap é um espaço contíguo de memória:

- alguns blocos estão reservados para nós

Lista de blocos livres

O heap é um espaço contíguo de memória:

- alguns blocos estão reservados para nós
- o resto do espaço está dividido em blocos livres

Lista de blocos livres

O heap é um espaço contíguo de memória:

- alguns blocos estão reservados para nós
- o resto do espaço está dividido em blocos livres

Podemos criar uma lista de blocos livres

Lista de blocos livres

O heap é um espaço contíguo de memória:

- alguns blocos estão reservados para nós
- o resto do espaço está dividido em blocos livres

Podemos criar uma lista de blocos livres

- mas não podemos alocar memória

Lista de blocos livres

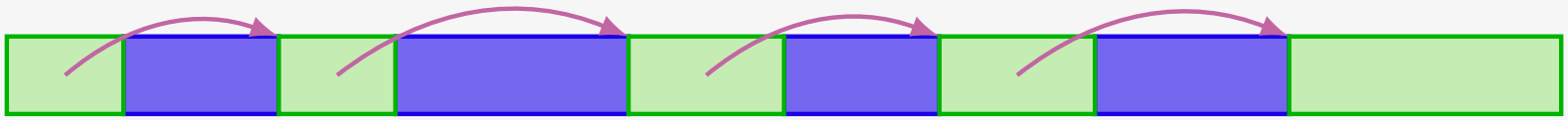
O heap é um espaço contíguo de memória:

- alguns blocos estão reservados para nós
- o resto do espaço está dividido em blocos livres

Podemos criar uma lista de blocos livres

- mas não podemos alocar memória
- onde guardar os nós da lista?

Lista de blocos livres



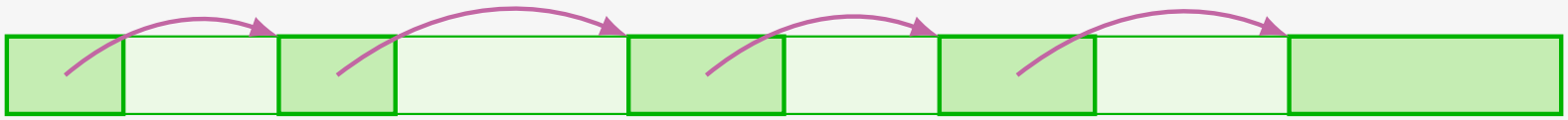
O heap é um espaço contíguo de memória:

- alguns blocos estão reservados para nós
- o resto do espaço está dividido em blocos livres

Podemos criar uma lista de blocos livres

- mas não podemos alocar memória
- onde guardar os nós da lista?
- guardamos ponteiros e tamanho nos próprios blocos

Lista de blocos livres



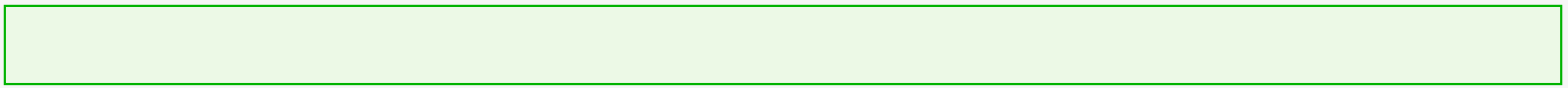
O heap é um espaço contíguo de memória:

- alguns blocos estão reservados para nós
- o resto do espaço está dividido em blocos livres

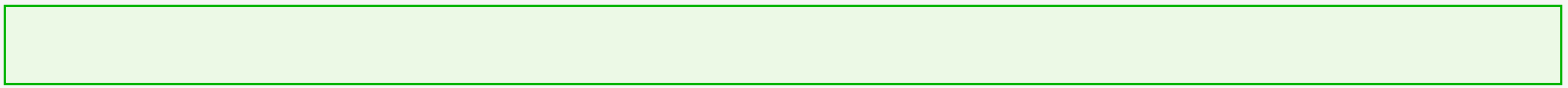
Podemos criar uma lista de blocos livres

- mas não podemos alocar memória
- onde guardar os nós da lista?
- guardamos ponteiros e tamanho nos próprios blocos

Informações adicionais

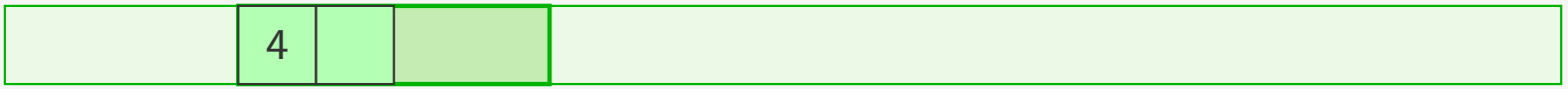


Informações adicionais



Um bloco livre contém:

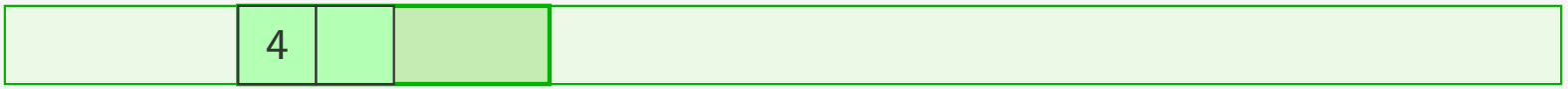
Informações adicionais



Um bloco livre contém:

- tamanho do bloco

Informações adicionais



Um bloco livre contém:

- tamanho do bloco
- ponteiro para o próximo

Informações adicionais

Um bloco livre contém:

- tamanho do bloco
- ponteiro para o próximo

Informações adicionais

Um bloco livre contém:

- tamanho do bloco
- ponteiro para o próximo

Informações adicionais

Um bloco livre contém:

- tamanho do bloco
- ponteiro para o próximo

Informações adicionais

Um bloco livre contém:

- tamanho do bloco
- ponteiro para o próximo

Um bloco reservado contém:

Informações adicionais

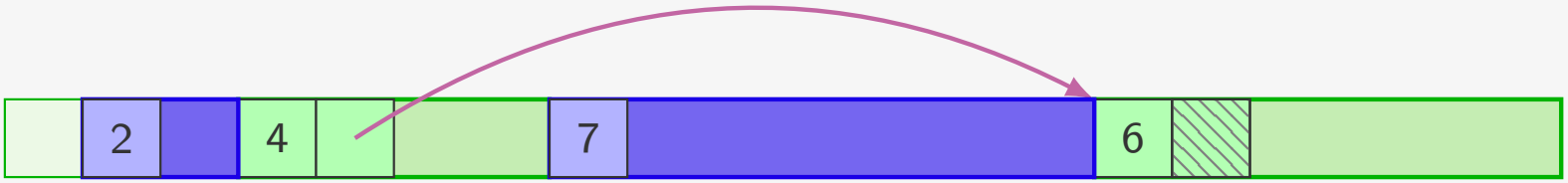
Um bloco livre contém:

- tamanho do bloco
- ponteiro para o próximo

Um bloco reservado contém:

- tamanho do bloco
- flag de bloco reservado

Informações adicionais



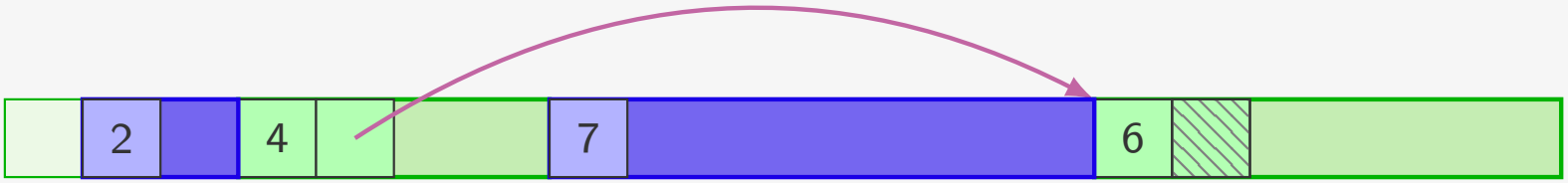
Um bloco livre contém:

- tamanho do bloco
- ponteiro para o próximo

Um bloco reservado contém:

- tamanho do bloco
- flag de bloco reservado

Informações adicionais



Um bloco livre contém:

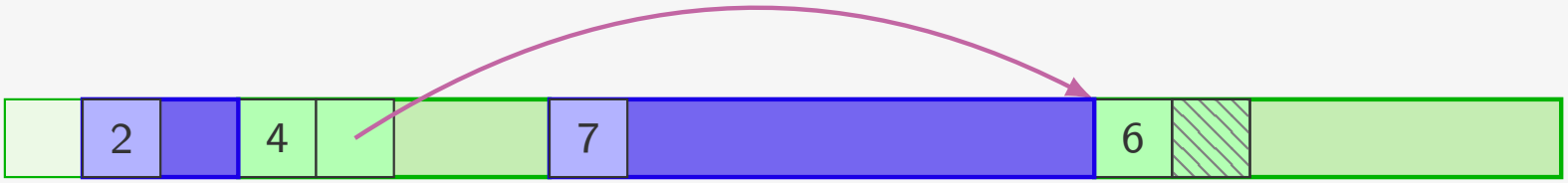
- tamanho do bloco
- ponteiro para o próximo

Um bloco reservado contém:

- tamanho do bloco
- flag de bloco reservado

O heap contém:

Informações adicionais



Um bloco livre contém:

- tamanho do bloco
- ponteiro para o próximo

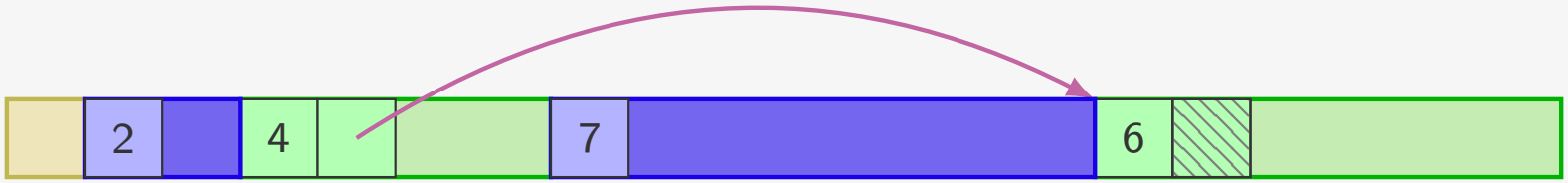
Um bloco reservado contém:

- tamanho do bloco
- flag de bloco reservado

O heap contém:

- o ponteiro para o primeiro bloco livre

Informações adicionais



Um bloco livre contém:

- tamanho do bloco
- ponteiro para o próximo

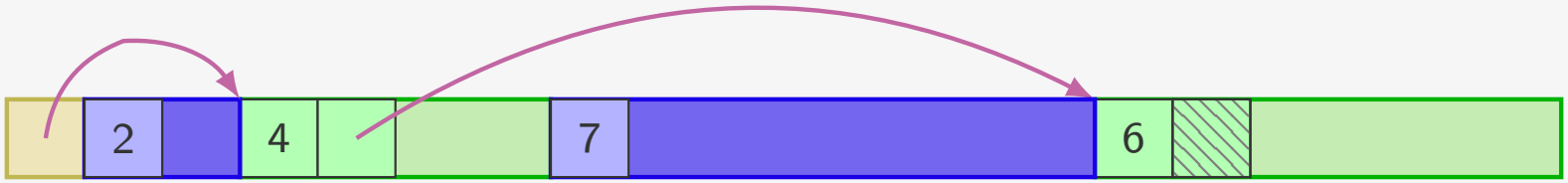
Um bloco reservado contém:

- tamanho do bloco
- flag de bloco reservado

O heap contém:

- o ponteiro para o primeiro bloco livre

Informações adicionais



Um bloco livre contém:

- tamanho do bloco
- ponteiro para o próximo

Um bloco reservado contém:

- tamanho do bloco
- flag de bloco reservado

O heap contém:

- o ponteiro para o primeiro bloco livre

Alocando memória

Como reservar um novo bloco com n bytes?

Alocando memória

Como reservar um novo bloco com n bytes?

- procuramos um bloco com tamanho suficiente

Alocando memória

Como reservar um novo bloco com n bytes?

- procuramos um bloco com tamanho suficiente
- diminuímos tamanho do bloco livre

Alocando memória

Como reservar um novo bloco com n bytes?

- procuramos um bloco com tamanho suficiente
- diminuímos tamanho do bloco livre
- reservamos um bloco (metadados + espaço reservado)

Alocando memória

Como reservar um novo bloco com n bytes?

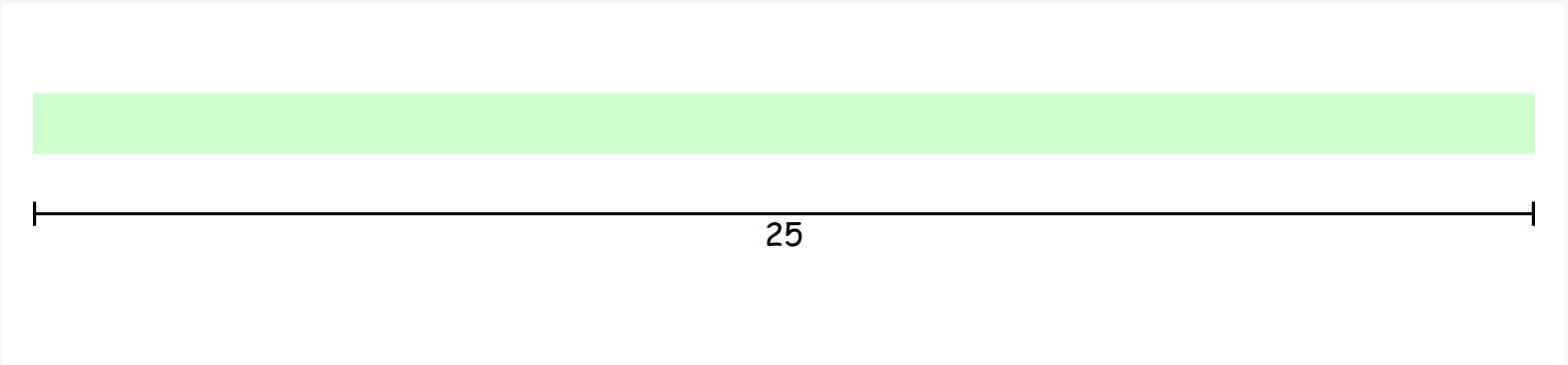
- procuramos um bloco com tamanho suficiente
- diminuímos tamanho do bloco livre
- reservamos um bloco (metadados + espaço reservado)
- devolvemos o endereço do espaço reservado

Primeira alocação

Começamos com um único bloco livre

Primeira alocação

Começamos com um único bloco livre

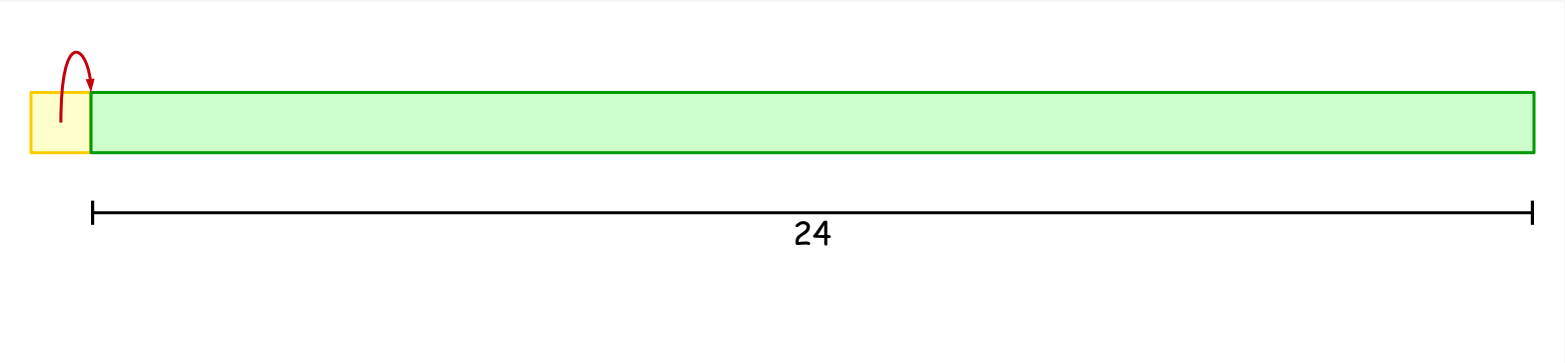


Primeira alocação

Começamos com um único bloco livre

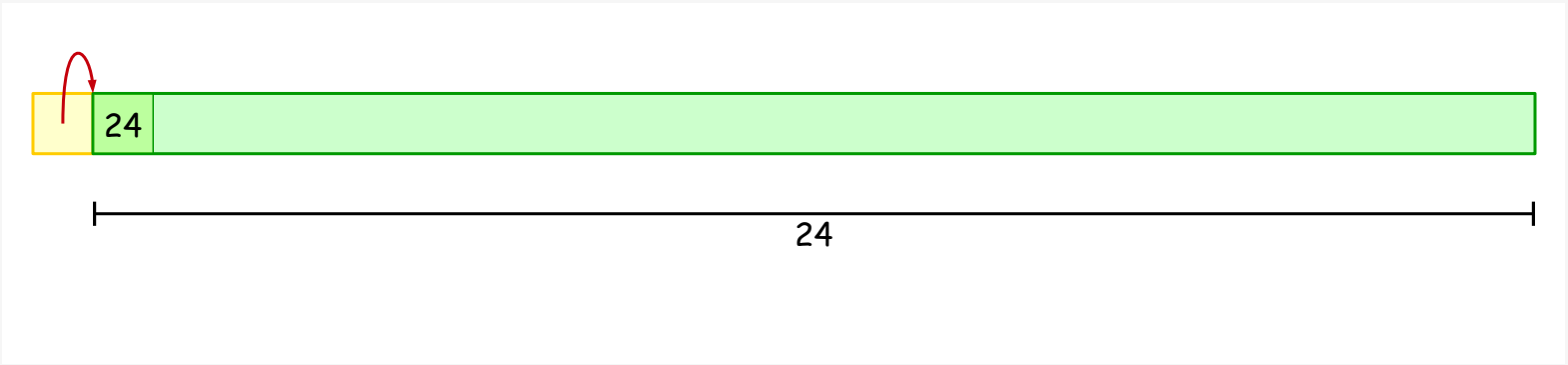
Primeira alocação

Começamos com um único bloco livre



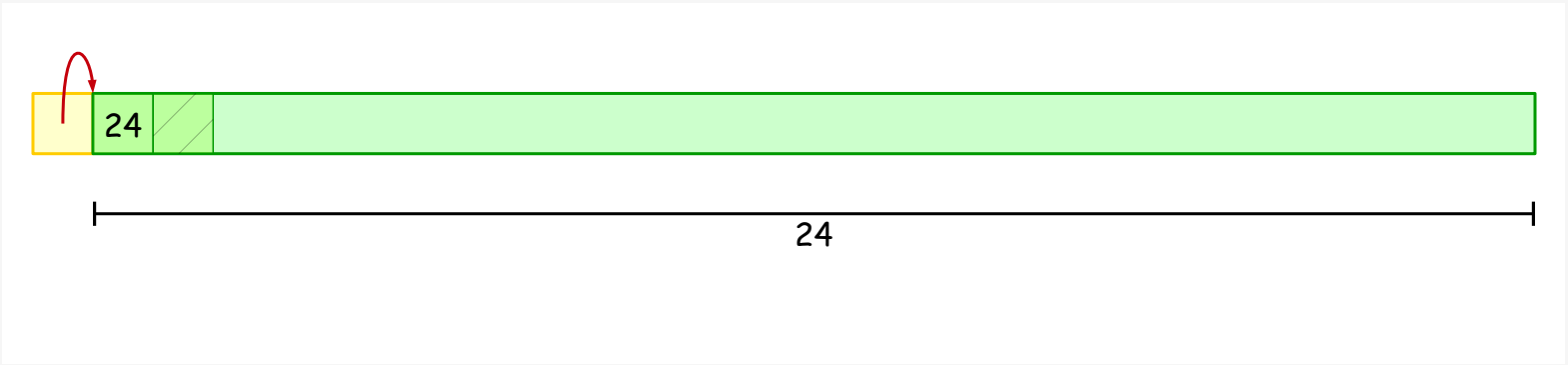
Primeira alocação

Começamos com um único bloco livre



Primeira alocação

Começamos com um único bloco livre

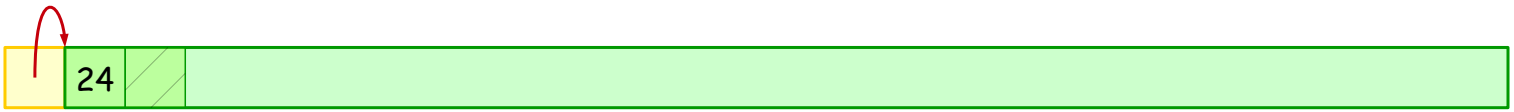


Primeira alocação

Começamos com um único bloco livre

Primeira alocação

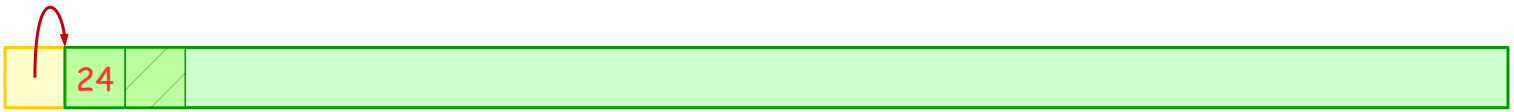
Começamos com um único bloco livre



Alocar 4 bytes

Primeira alocação

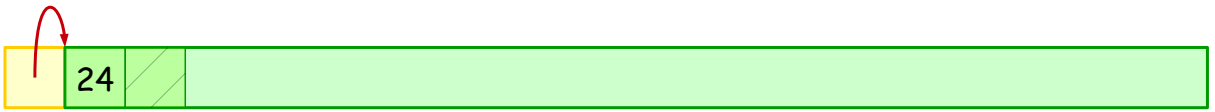
Começamos com um único bloco livre



Alocar 4 bytes

Primeira alocação

Começamos com um único bloco livre



Alocar 4 bytes

Primeira alocação

Começamos com um único bloco livre

Primeira alocação

Começamos com um único bloco livre

Primeira alocação

Começamos com um único bloco livre

Primeira alocação

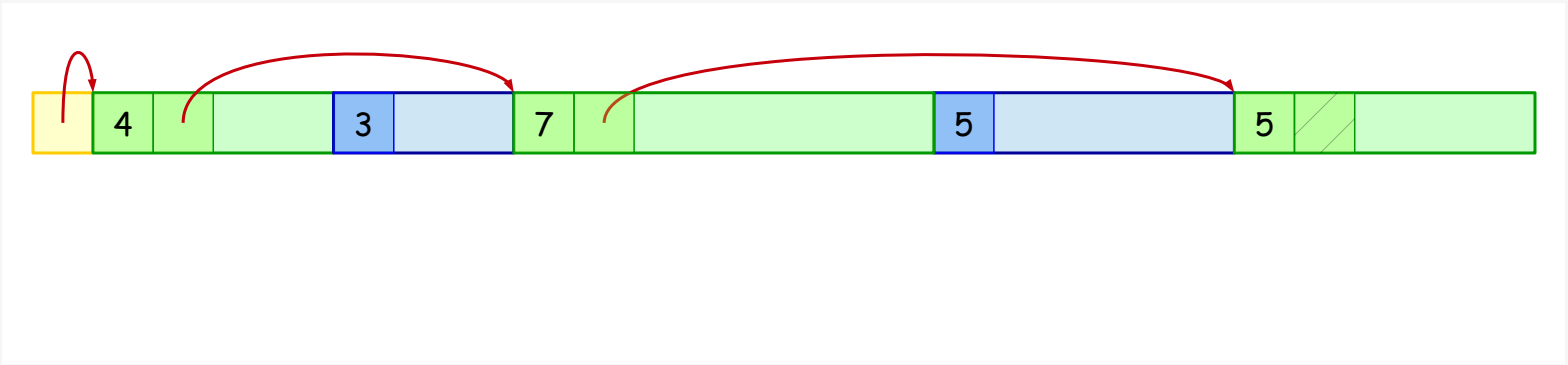
Começamos com um único bloco livre

Primeira alocação

Começamos com um único bloco livre

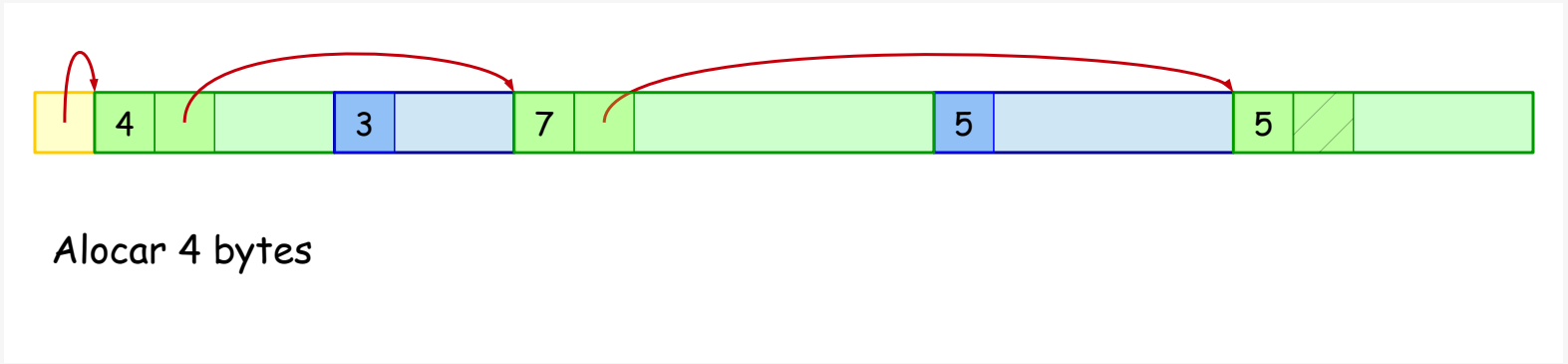
Removendo bloco da lista livre

Removemos da lista os blocos livres muito pequenos



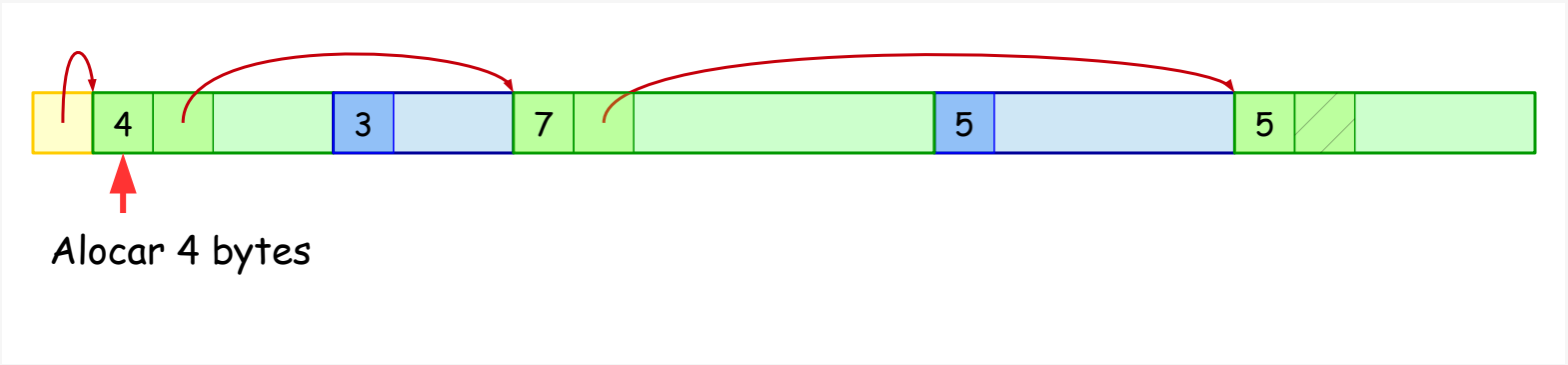
Removendo bloco da lista livre

Removemos da lista os blocos livres muito pequenos



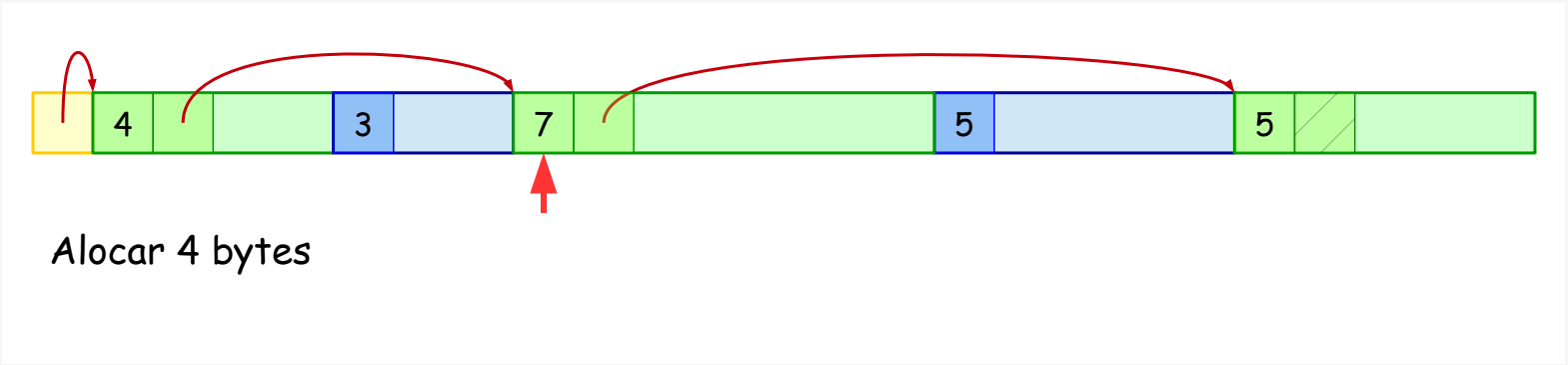
Removendo bloco da lista livre

Removemos da lista os blocos livres muito pequenos



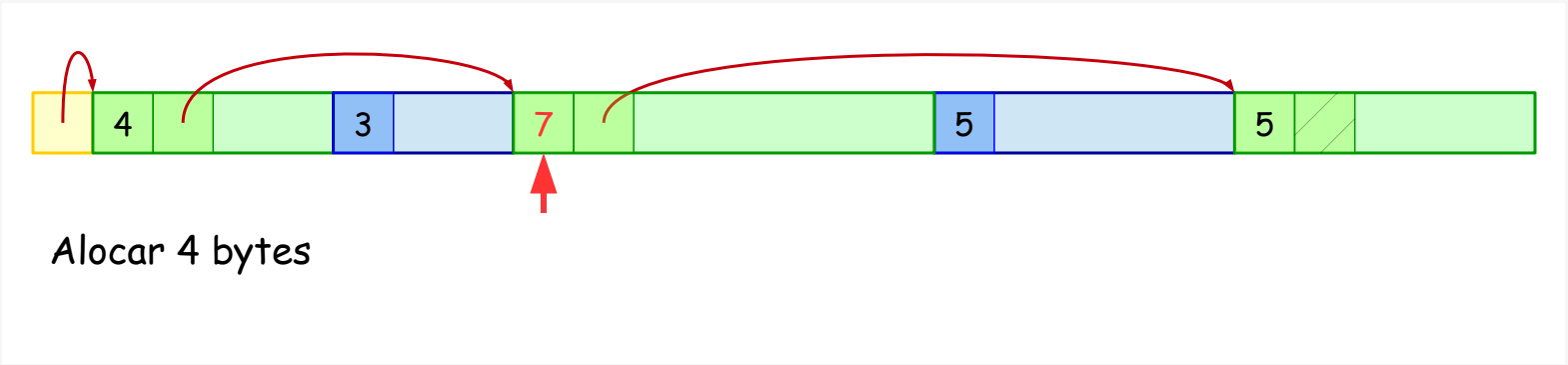
Removendo bloco da lista livre

Removemos da lista os blocos livres muito pequenos



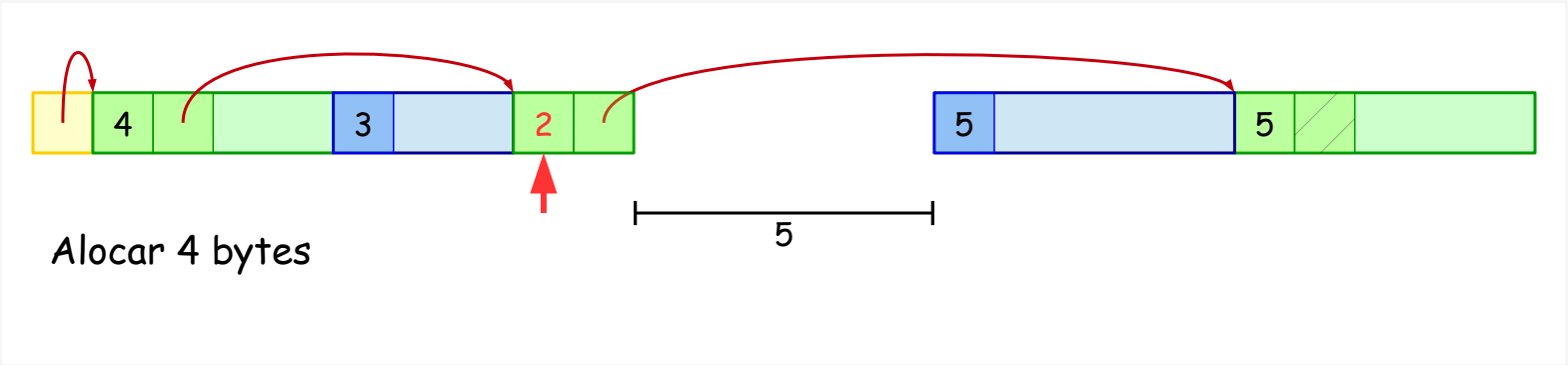
Removendo bloco da lista livre

Removemos da lista os blocos livres muito pequenos



Removendo bloco da lista livre

Removemos da lista os blocos livres muito pequenos

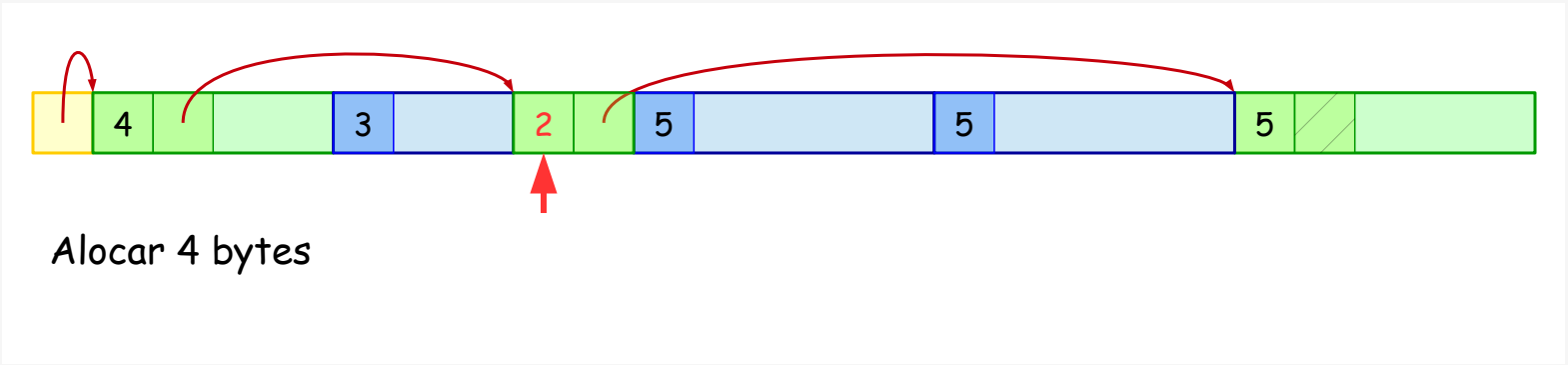


Removendo bloco da lista livre

Removemos da lista os blocos livres muito pequenos

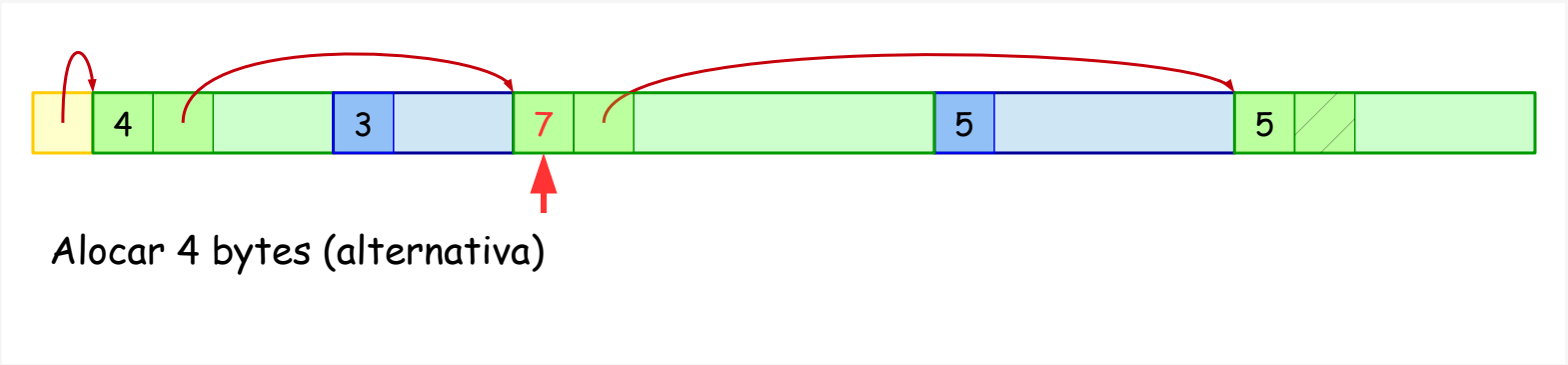
Removendo bloco da lista livre

Removemos da lista os blocos livres muito pequenos



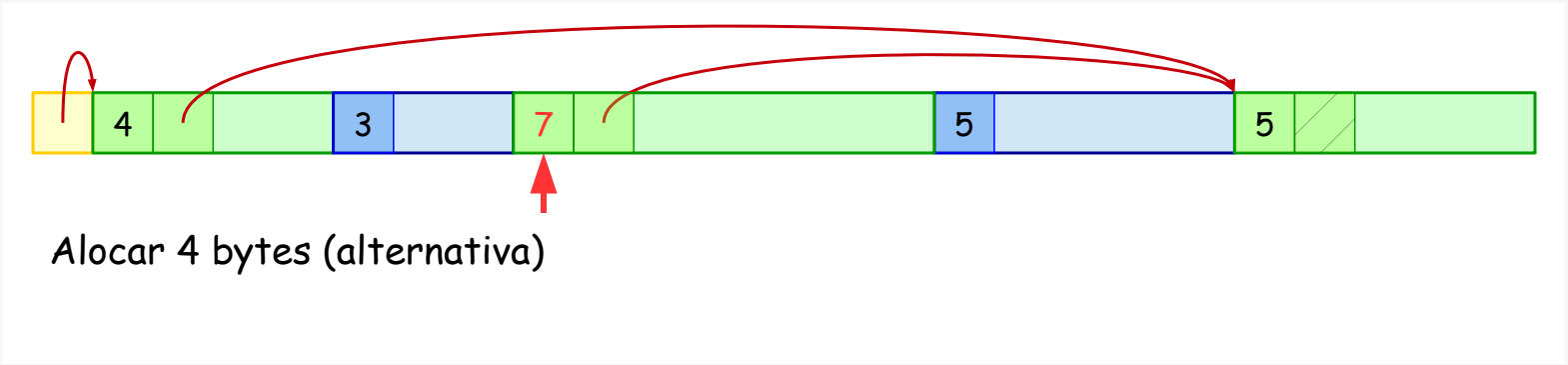
Removendo bloco da lista livre

Removemos da lista os blocos livres muito pequenos



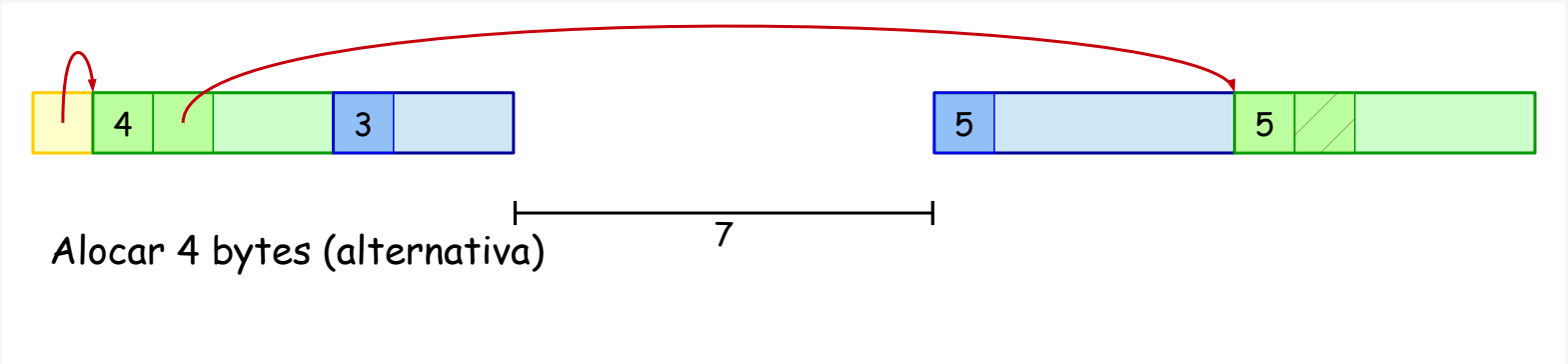
Removendo bloco da lista livre

Removemos da lista os blocos livres muito pequenos



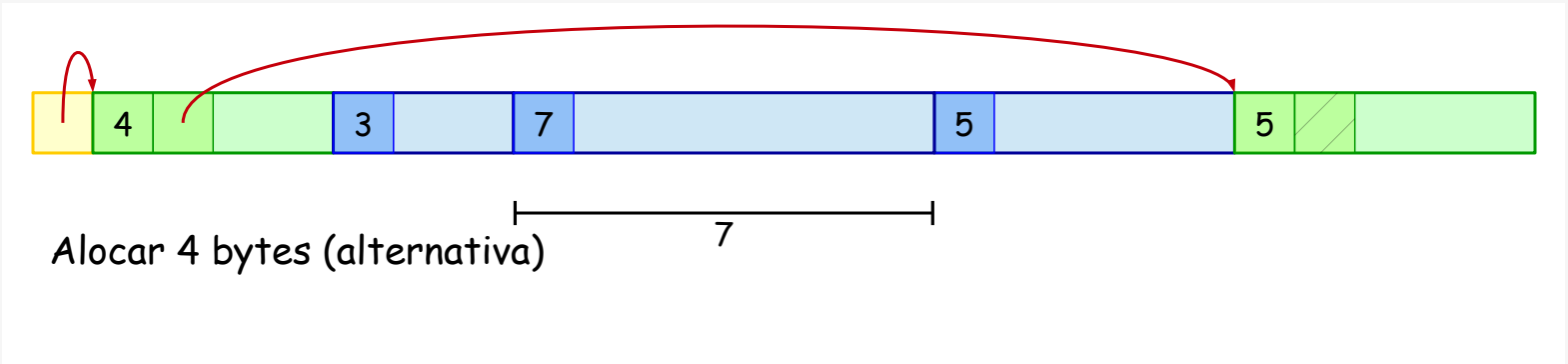
Removendo bloco da lista livre

Removemos da lista os blocos livres muito pequenos



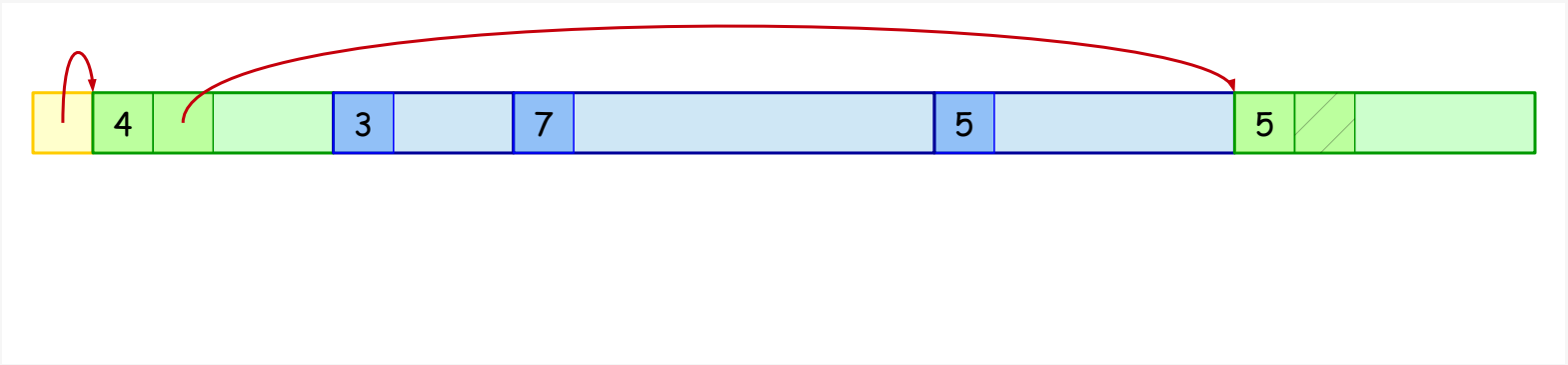
Removendo bloco da lista livre

Removemos da lista os blocos livres muito pequenos



Removendo bloco da lista livre

Removemos da lista os blocos livres muito pequenos



Faltando memória

Pode ser que nenhum bloco livre é grande o suficiente

Faltando memória

Pode ser que nenhum bloco livre é grande o suficiente

- a memória pode estar fragmentada

Faltando memória

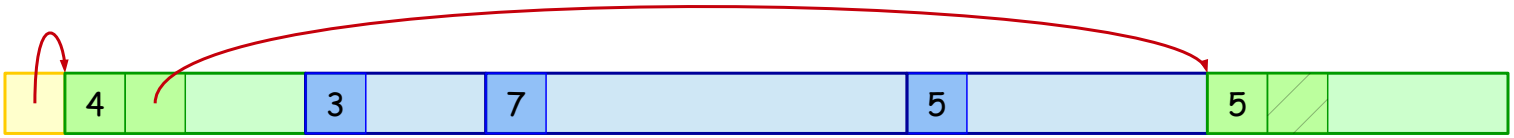
Pode ser que nenhum bloco livre é grande o suficiente

- a memória pode estar fragmentada
- devolvemos `NULL` para indicar o erro

Faltando memória

Pode ser que nenhum bloco livre é grande o suficiente

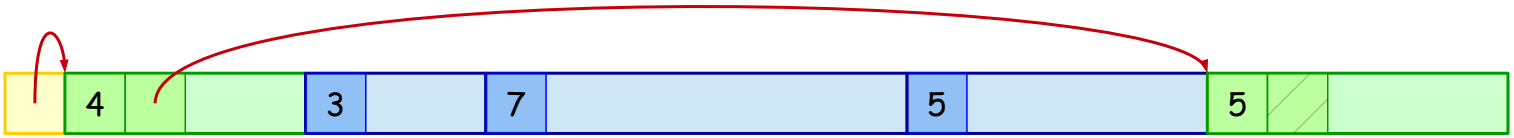
- a memória pode estar fragmentada
- devolvemos `NULL` para indicar o erro



Faltando memória

Pode ser que nenhum bloco livre é grande o suficiente

- a memória pode estar fragmentada
- devolvemos `NULL` para indicar o erro

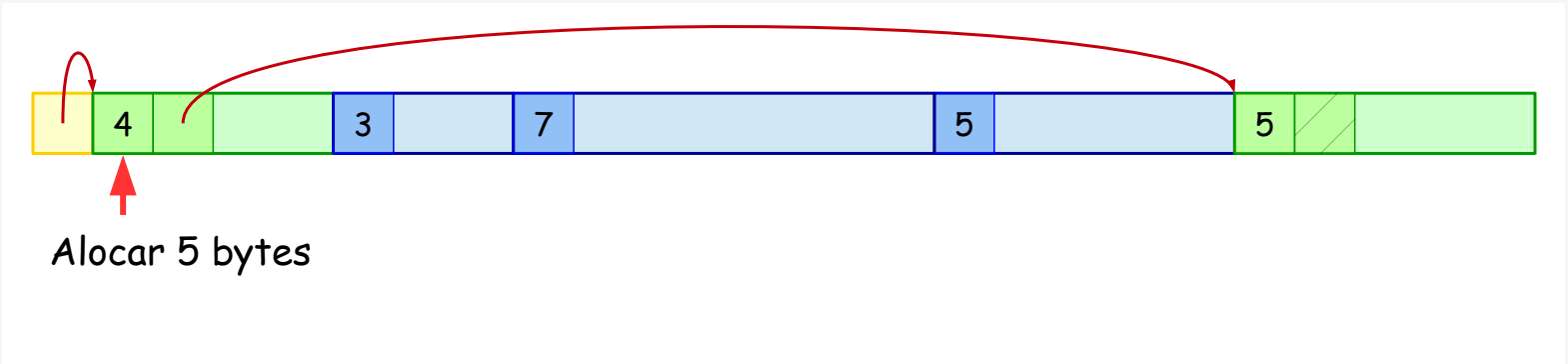


Alocar 5 bytes

Faltando memória

Pode ser que nenhum bloco livre é grande o suficiente

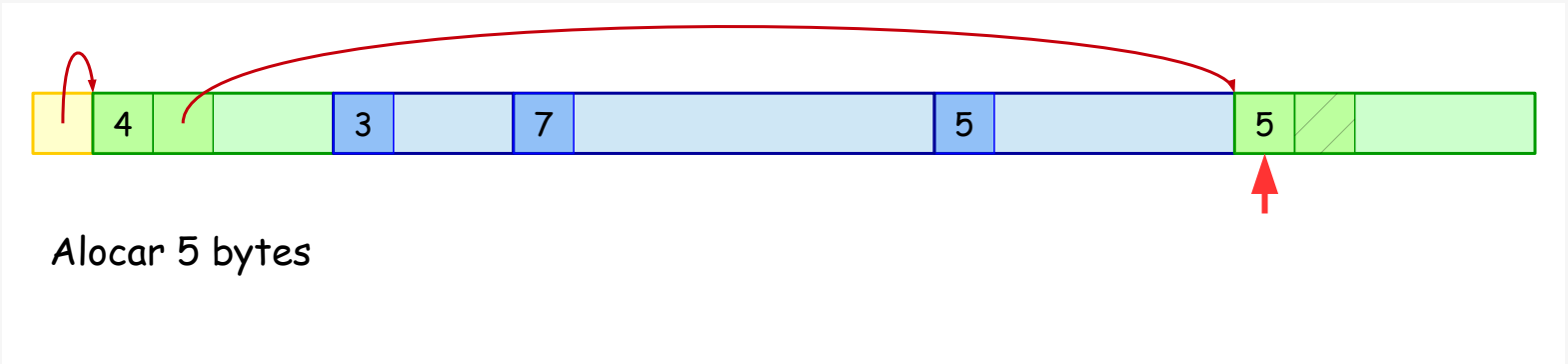
- a memória pode estar fragmentada
- devolvemos `NULL` para indicar o erro



Faltando memória

Pode ser que nenhum bloco livre é grande o suficiente

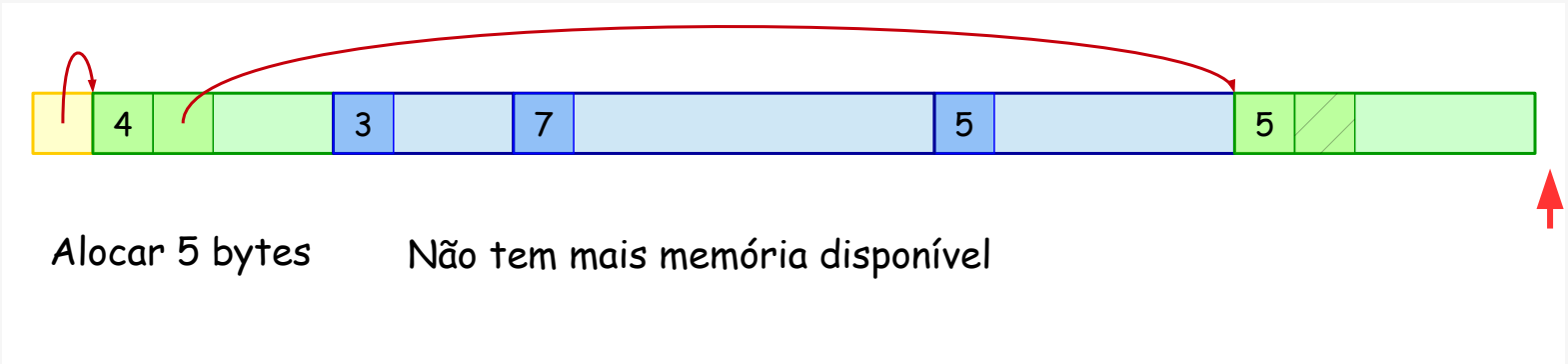
- a memória pode estar fragmentada
- devolvemos `NULL` para indicar o erro



Faltando memória

Pode ser que nenhum bloco livre é grande o suficiente

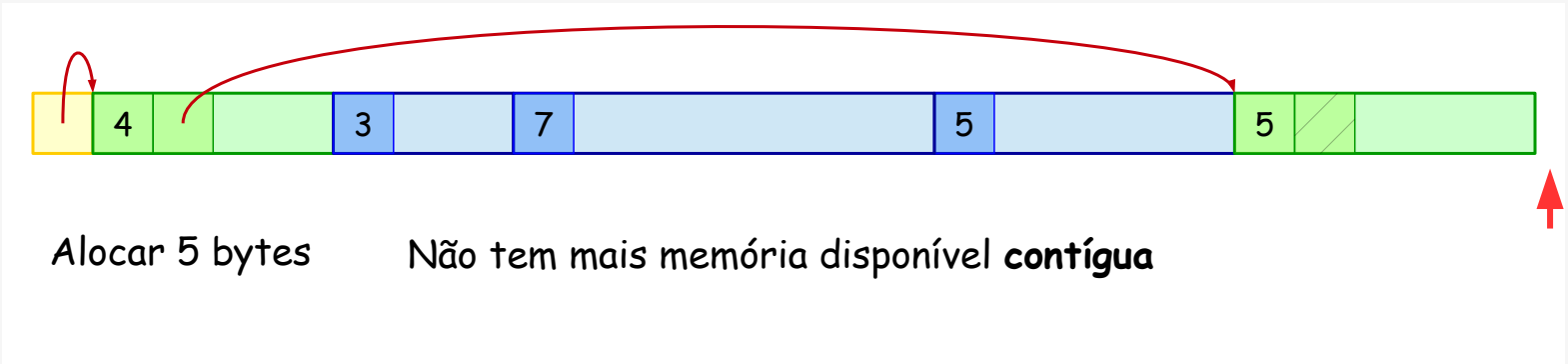
- a memória pode estar fragmentada
- devolvemos `NULL` para indicar o erro



Faltando memória

Pode ser que nenhum bloco livre é grande o suficiente

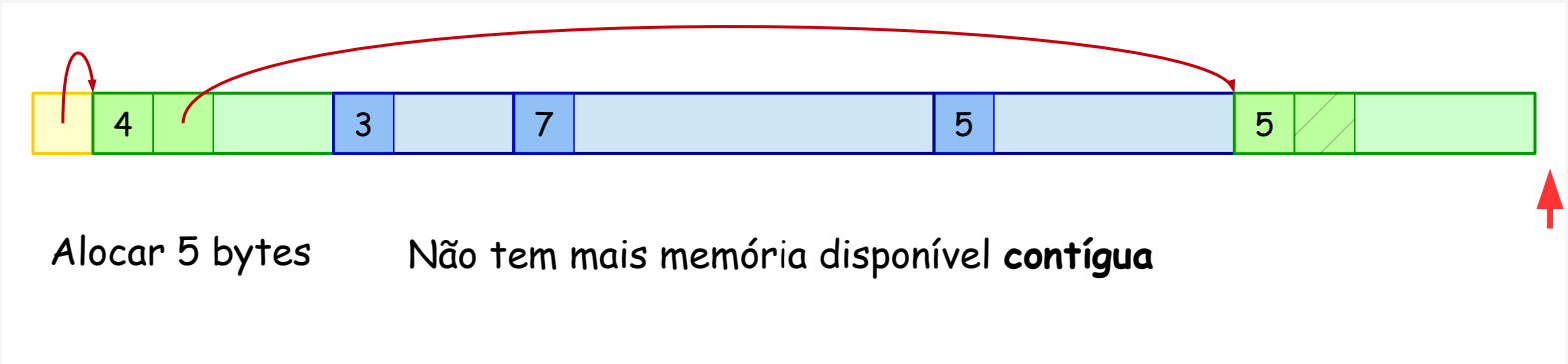
- a memória pode estar fragmentada
- devolvemos `NULL` para indicar o erro



Faltando memória

Pode ser que nenhum bloco livre é grande o suficiente

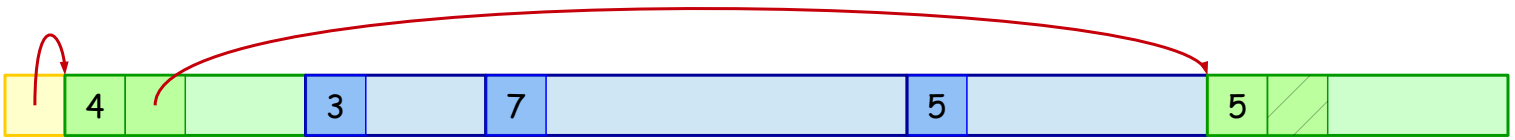
- a memória pode estar fragmentada
- devolvemos `NULL` para indicar o erro



Faltando memória

Pode ser que nenhum bloco livre é grande o suficiente

- a memória pode estar fragmentada
- devolvemos `NULL` para indicar o erro



Alocar 5 bytes

Não tem mais memória disponível **contígua**

→ retornamos erro

Liberando memória

Como liberar um bloco com endereço p?

Liberando memória

Como liberar um bloco com endereço p?

- procuramos o último bloco livre antes de p

Liberando memória

Como liberar um bloco com endereço p?

- procuramos o último bloco livre antes de p
- criamos um novo bloco livre

Liberando memória

Como liberar um bloco com endereço p?

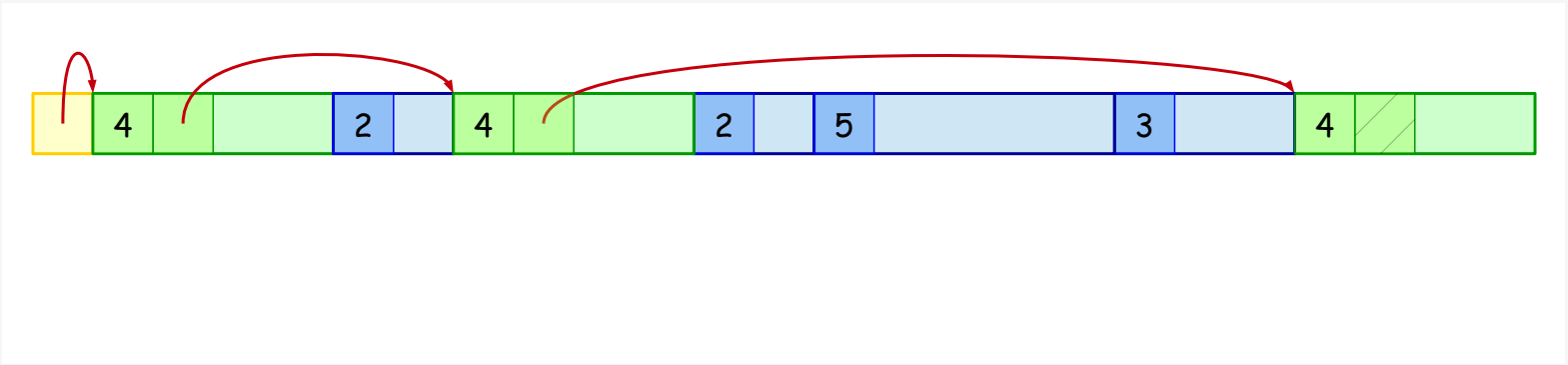
- procuramos o último bloco livre antes de p
- criamos um novo bloco livre
- inserimos após o último bloco livre

Liberando entre blocos reservados

Liberando memória em certa posição

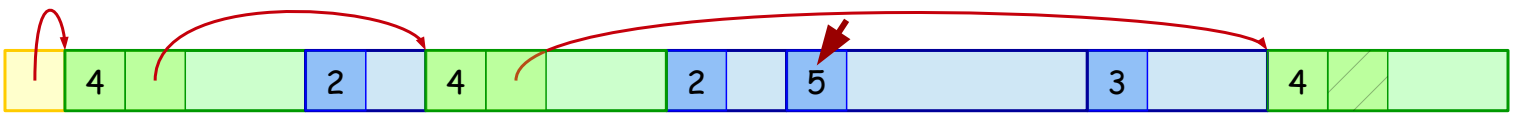
Liberando entre blocos reservados

Liberando memória em certa posição



Liberando entre blocos reservados

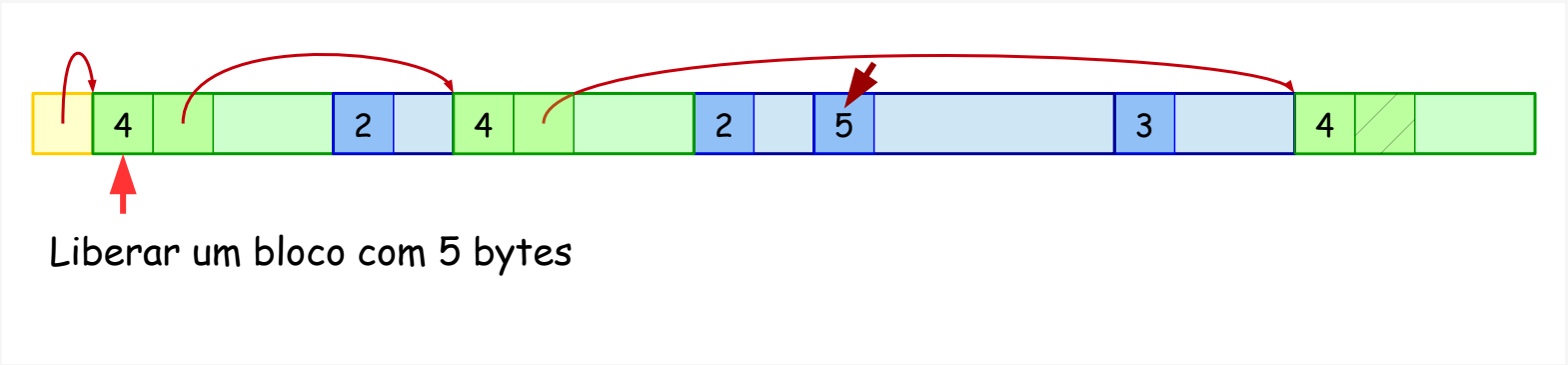
Liberando memória em certa posição



Liberar um bloco com 5 bytes

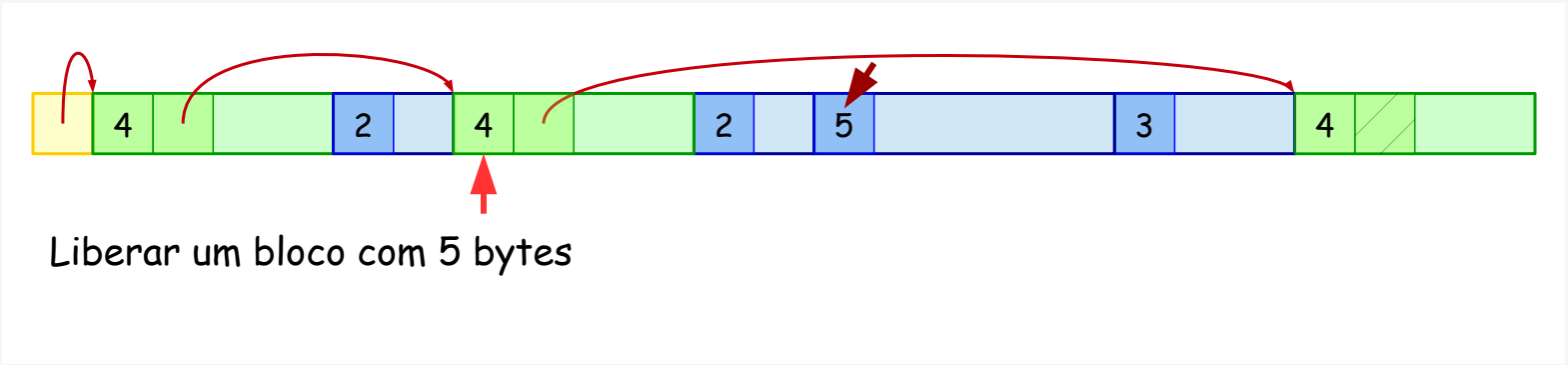
Liberando entre blocos reservados

Liberando memória em certa posição



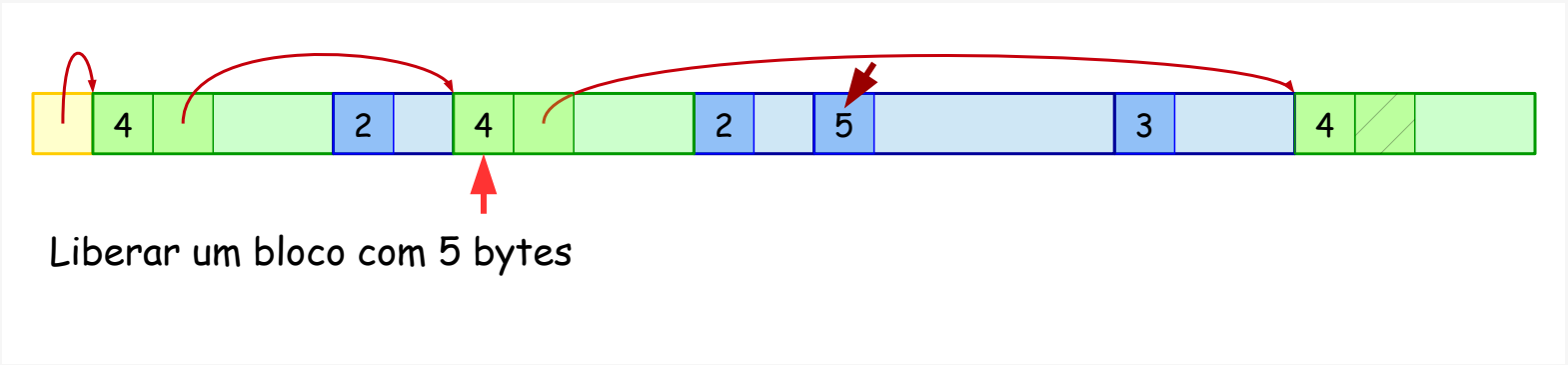
Liberando entre blocos reservados

Liberando memória em certa posição



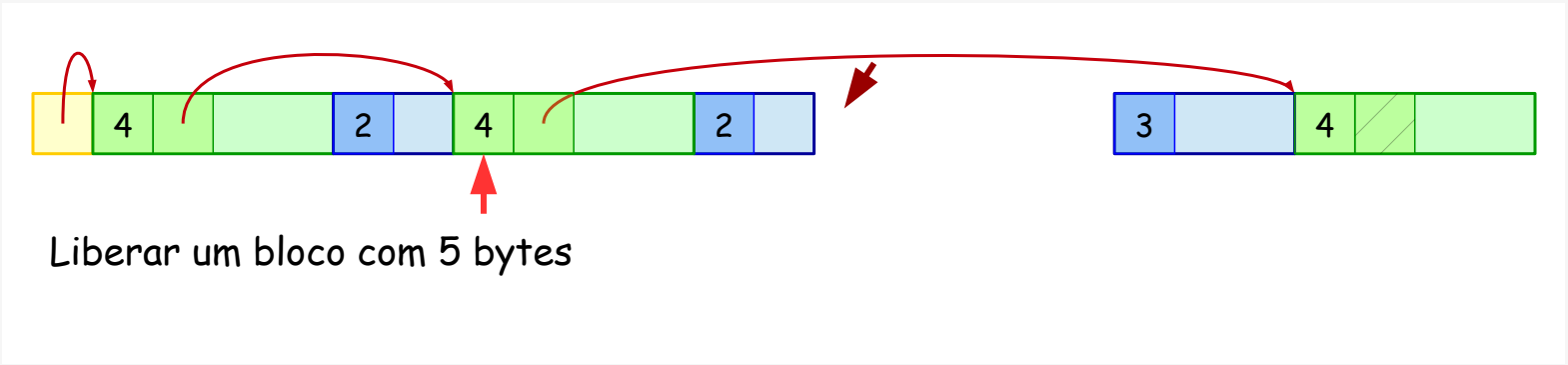
Liberando entre blocos reservados

Liberando memória em certa posição



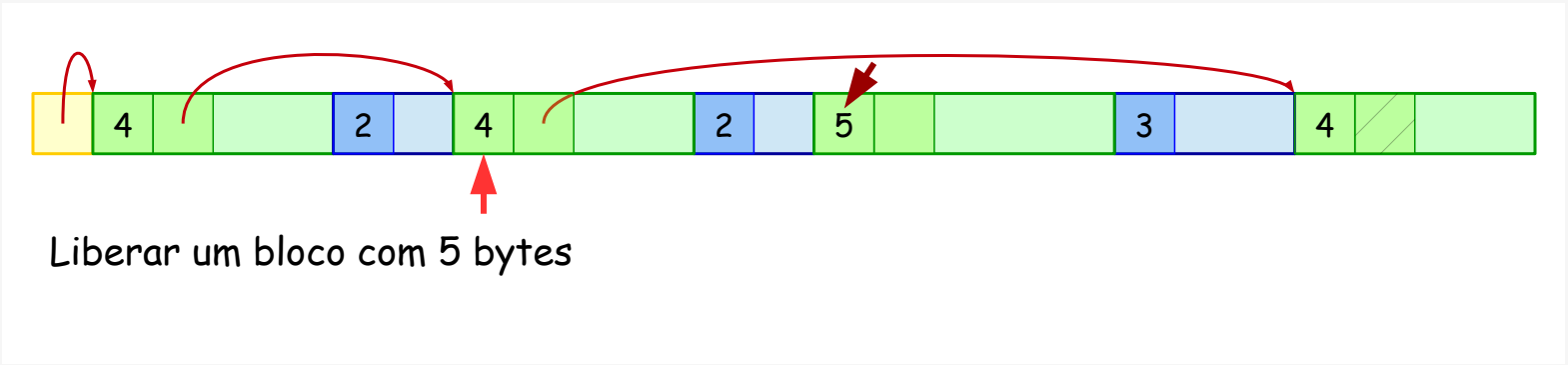
Liberando entre blocos reservados

Liberando memória em certa posição



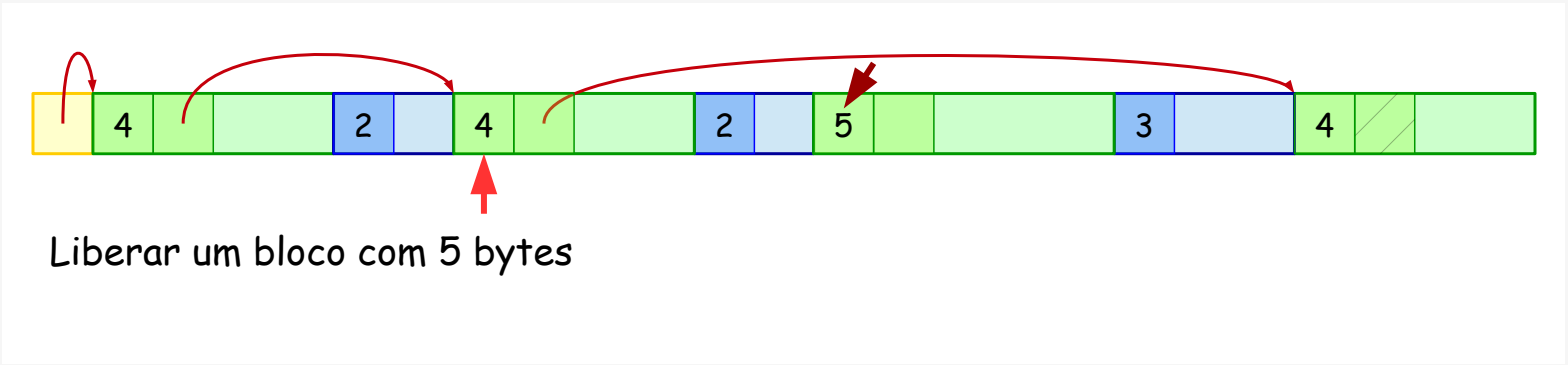
Liberando entre blocos reservados

Liberando memória em certa posição



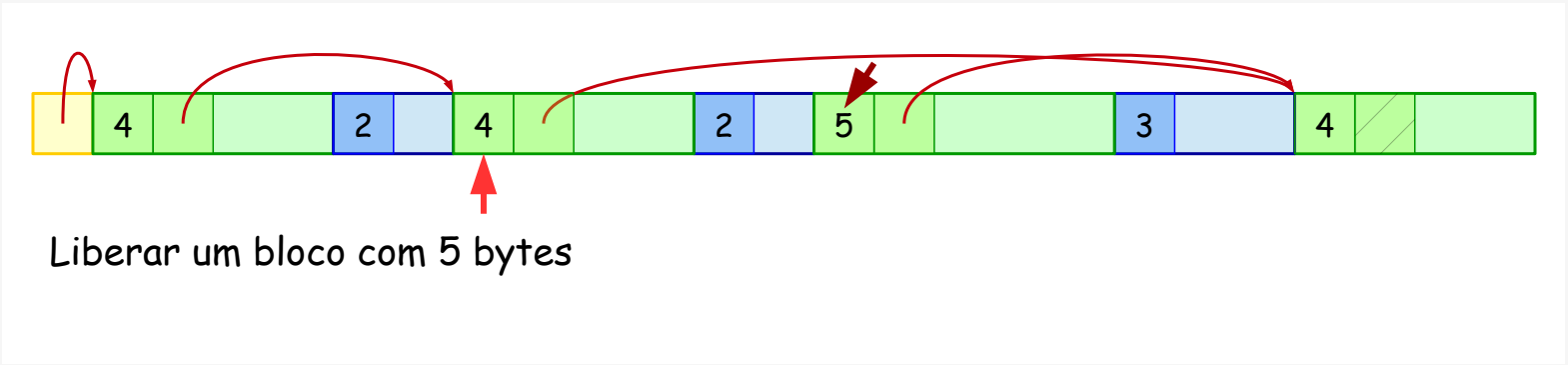
Liberando entre blocos reservados

Liberando memória em certa posição



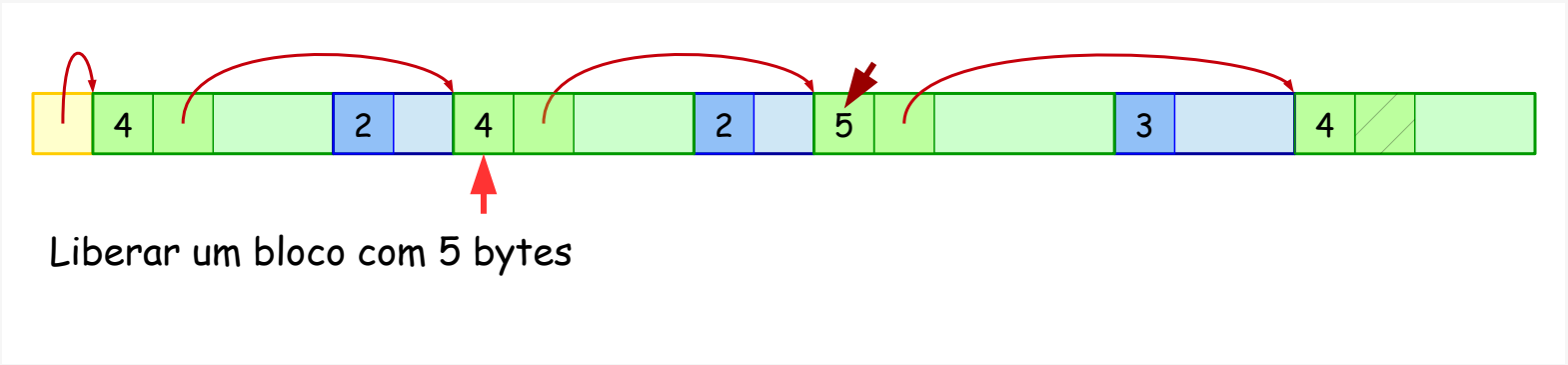
Liberando entre blocos reservados

Liberando memória em certa posição



Liberando entre blocos reservados

Liberando memória em certa posição



Combinando blocos livres

Pode ser que haja blocos livres adjacentes

Combinando blocos livres

Pode ser que haja blocos livres adjacentes

- se o último for adjacente, aumentamos o tamanho

Combinando blocos livres

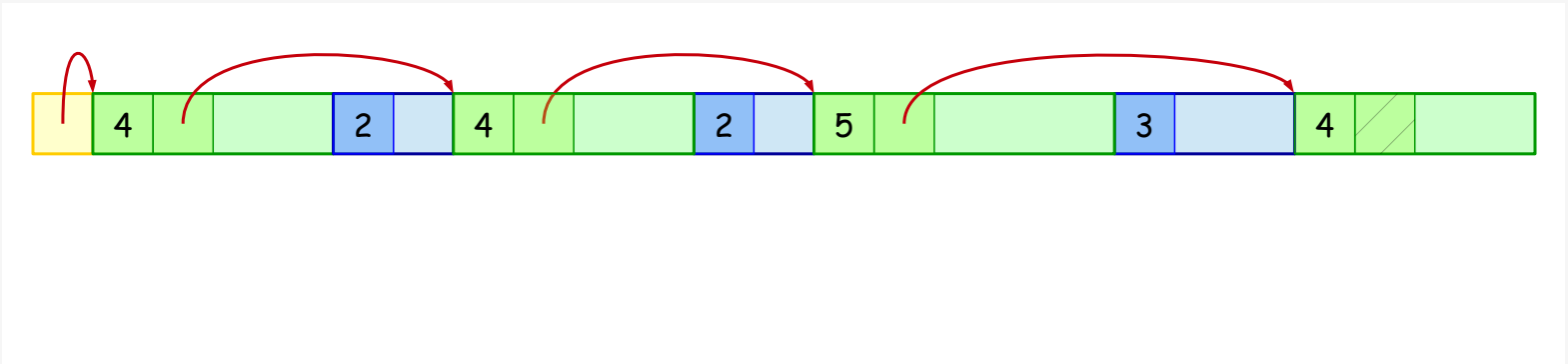
Pode ser que haja blocos livres adjacentes

- se o último for adjacente, aumentamos o tamanho
- se o próximo for adjacente, removemos da lista

Combinando blocos livres

Pode ser que haja blocos livres adjacentes

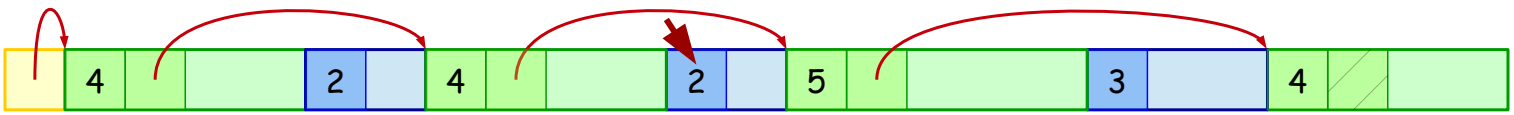
- se o último for adjacente, aumentamos o tamanho
- se o próximo for adjacente, removemos da lista



Combinando blocos livres

Pode ser que haja blocos livres adjacentes

- se o último for adjacente, aumentamos o tamanho
- se o próximo for adjacente, removemos da lista

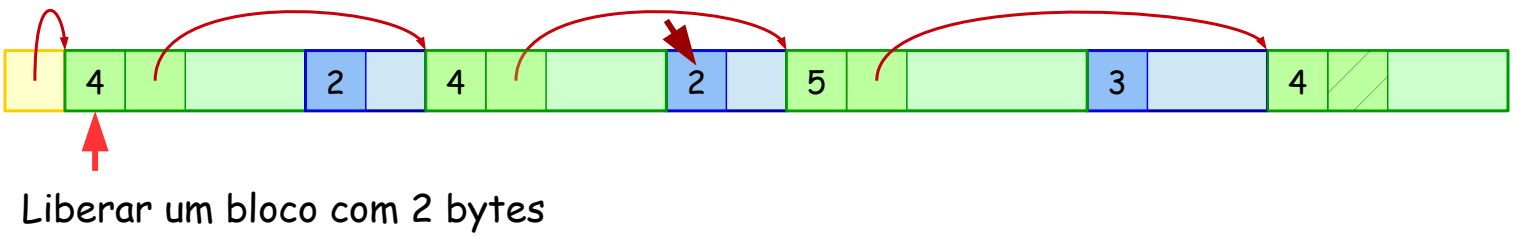


Liberar um bloco com 2 bytes

Combinando blocos livres

Pode ser que haja blocos livres adjacentes

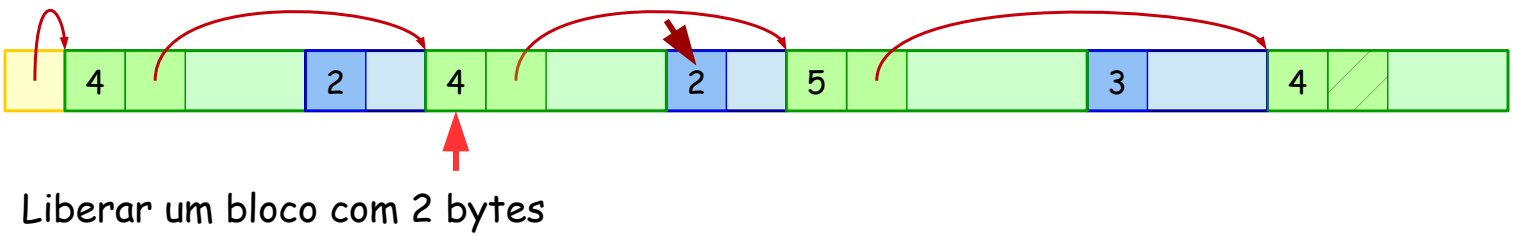
- se o último for adjacente, aumentamos o tamanho
- se o próximo for adjacente, removemos da lista



Combinando blocos livres

Pode ser que haja blocos livres adjacentes

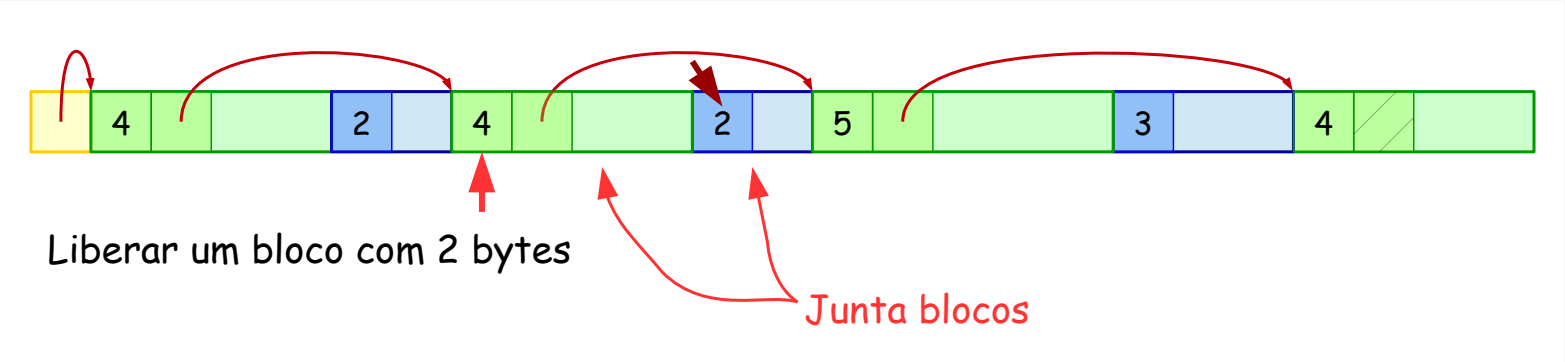
- se o último for adjacente, aumentamos o tamanho
- se o próximo for adjacente, removemos da lista



Combinando blocos livres

Pode ser que haja blocos livres adjacentes

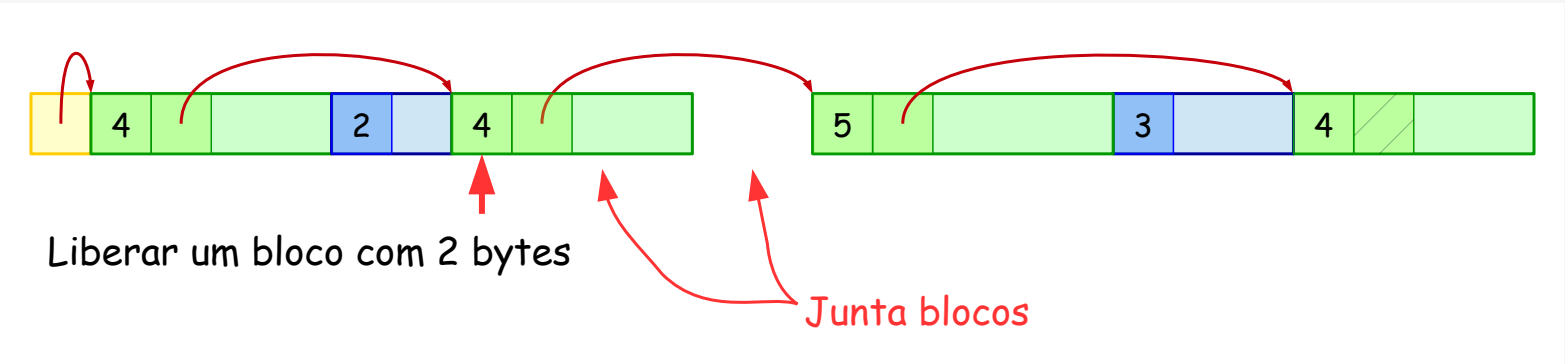
- se o último for adjacente, aumentamos o tamanho
- se o próximo for adjacente, removemos da lista



Combinando blocos livres

Pode ser que haja blocos livres adjacentes

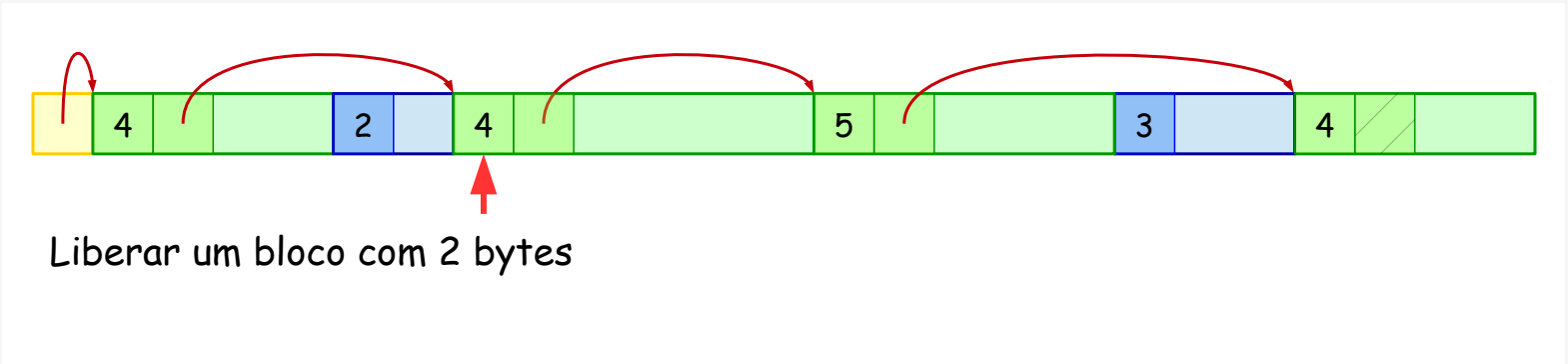
- se o último for adjacente, aumentamos o tamanho
- se o próximo for adjacente, removemos da lista



Combinando blocos livres

Pode ser que haja blocos livres adjacentes

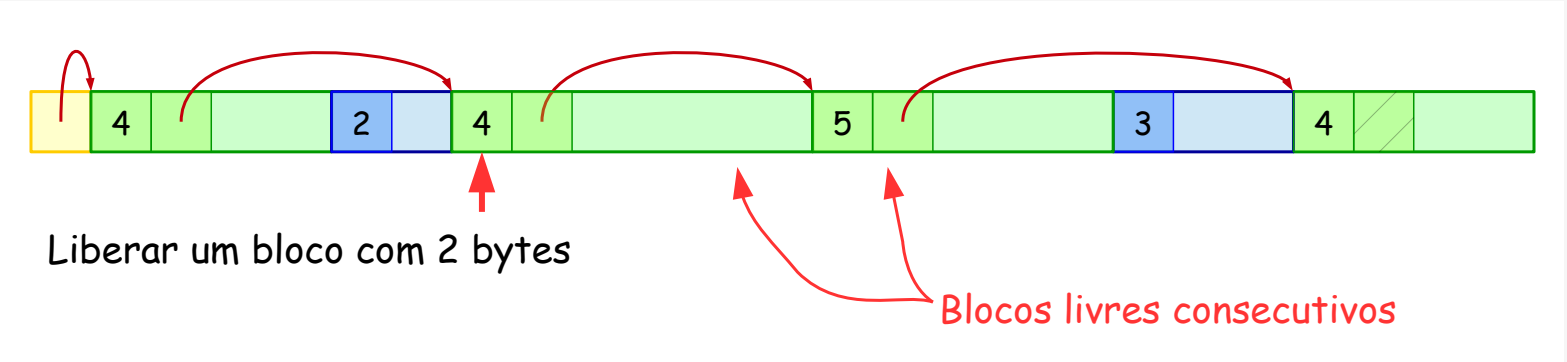
- se o último for adjacente, aumentamos o tamanho
- se o próximo for adjacente, removemos da lista



Combinando blocos livres

Pode ser que haja blocos livres adjacentes

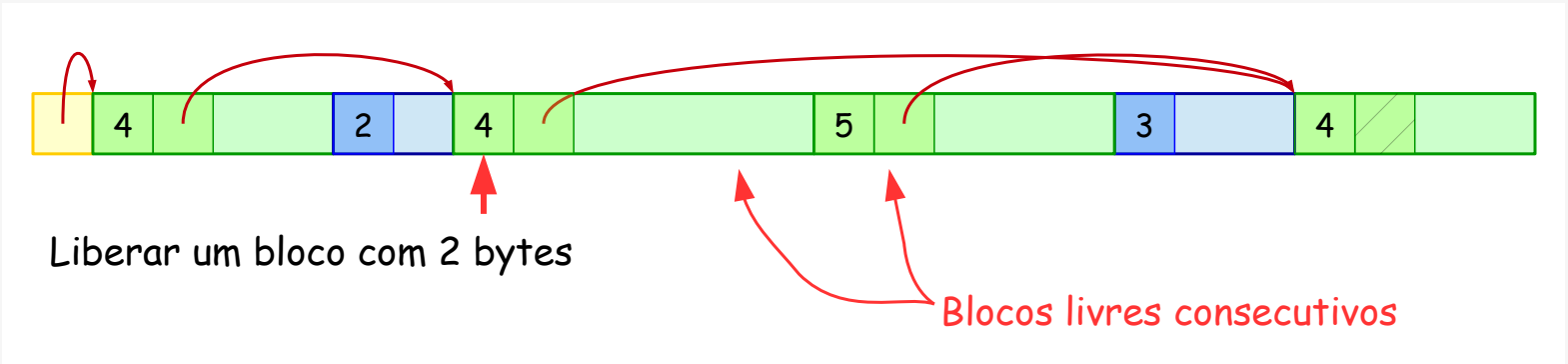
- se o último for adjacente, aumentamos o tamanho
- se o próximo for adjacente, removemos da lista



Combinando blocos livres

Pode ser que haja blocos livres adjacentes

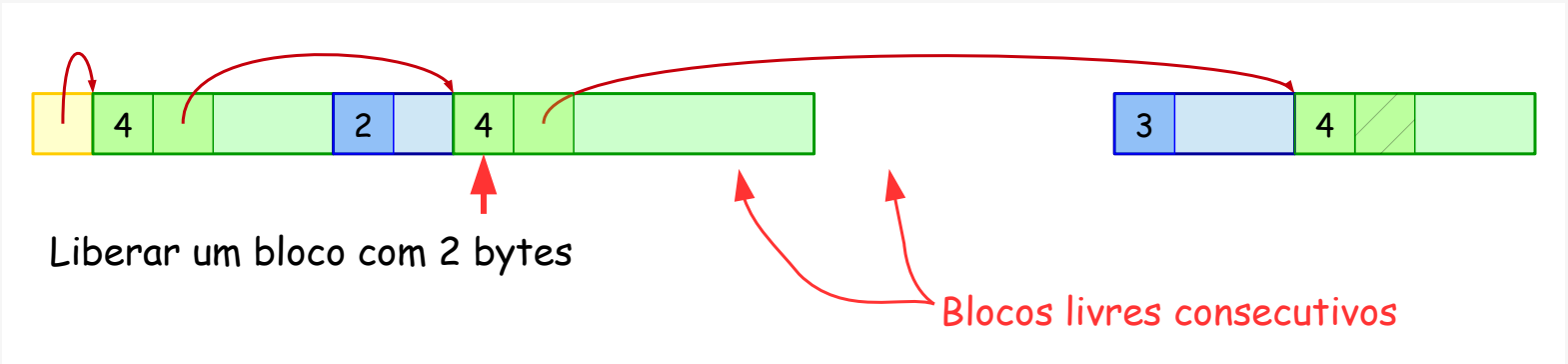
- se o último for adjacente, aumentamos o tamanho
- se o próximo for adjacente, removemos da lista



Combinando blocos livres

Pode ser que haja blocos livres adjacentes

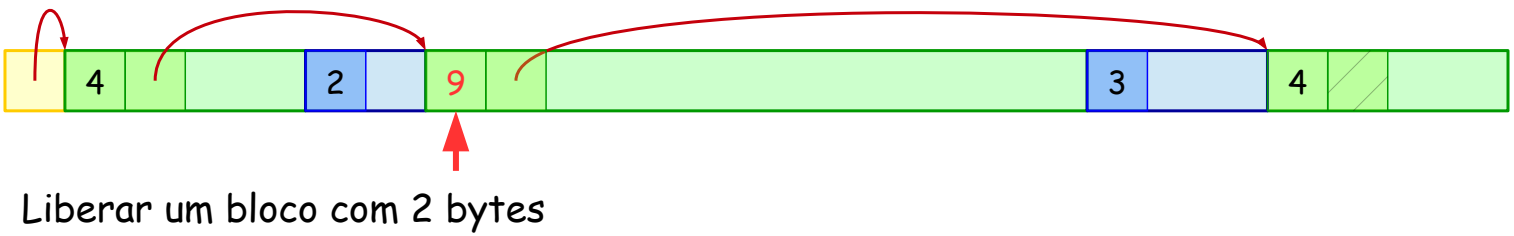
- se o último for adjacente, aumentamos o tamanho
- se o próximo for adjacente, removemos da lista



Combinando blocos livres

Pode ser que haja blocos livres adjacentes

- se o último for adjacente, aumentamos o tamanho
- se o próximo for adjacente, removemos da lista



Analisando

Fatores importantes:

Analisando

Fatores importantes:

- tempo para percorrer e modificar a lista de blocos

Analisando

Fatores importantes:

- tempo para percorrer e modificar a lista de blocos
- fragmentação da memória com alocação desordenada

Analisando

Fatores importantes:

- tempo para percorrer e modificar a lista de blocos
- fragmentação da memória com alocação desordenada

O gerenciamento de memória em C não é seguro

Analisando

Fatores importantes:

- tempo para percorrer e modificar a lista de blocos
- fragmentação da memória com alocação desordenada

O gerenciamento de memória em C não é seguro

- usuário é responsável pelo uso correto

Analisando

Fatores importantes:

- tempo para percorrer e modificar a lista de blocos
- fragmentação da memória com alocação desordenada

O gerenciamento de memória em C não é seguro

- usuário é responsável pelo uso correto
- double free pode corromper a lista

Analisando

Fatores importantes:

- tempo para percorrer e modificar a lista de blocos
- fragmentação da memória com alocação desordenada

O gerenciamento de memória em C não é seguro

- usuário é responsável pelo uso correto
- double free pode corromper a lista
- buffer overflow pode modificar metadados

Analisando

Fatores importantes:

- tempo para percorrer e modificar a lista de blocos
- fragmentação da memória com alocação desordenada

O gerenciamento de memória em C não é seguro

- usuário é responsável pelo uso correto
- double free pode corromper a lista
- buffer overflow pode modificar metadados
- são bugs extremamente difíceis de encontrar

Melhores práticas

Ao construir estruturas de dados, devemos:

Melhores práticas

Ao construir estruturas de dados, devemos:

- não alocar blocos excessivamente

Melhores práticas

Ao construir estruturas de dados, devemos:

- não alocar blocos excessivamente
- evitar estruturas com muito nós pequenos

Melhores práticas

Ao construir estruturas de dados, devemos:

- não alocar blocos excessivamente
- evitar estruturas com muito nós pequenos
- usar boas estimativas para tamanho usado

Melhores práticas

Ao construir estruturas de dados, devemos:

- não alocar blocos excessivamente
- evitar estruturas com muito nós pequenos
- usar boas estimativas para tamanho usado
- liberar cada bloco de memória alocado uma vez

Melhores práticas

Ao construir estruturas de dados, devemos:

- não alocar blocos excessivamente
- evitar estruturas com muito nós pequenos
- usar boas estimativas para tamanho usado
- liberar cada bloco de memória alocado uma vez
- modificar somente o espaço de memória reservado

Gerenciamento de memória

Explícito:

Gerenciamento de memória

Explícito:

- o programador aloca memória

Gerenciamento de memória

Explícito:

- o programador aloca memória
- o programador é responsável por liberar a memória

Gerenciamento de memória

Explícito:

- o programador aloca memória
- o programador é responsável por liberar a memória
- exemplos são C, C++, Pascal

Gerenciamento de memória

Explícito:

- o programador aloca memória
- o programador é responsável por liberar a memória
- exemplos são C, C++, Pascal

Implícito:

Gerenciamento de memória

Explícito:

- o programador aloca memória
- o programador é responsável por liberar a memória
- exemplos são C, C++, Pascal

Implícito:

- o programador cria variáveis de maneira restrita

Gerenciamento de memória

Explícito:

- o programador aloca memória
- o programador é responsável por liberar a memória
- exemplos são C, C++, Pascal

Implícito:

- o programador cria variáveis de maneira restrita
- a linguagem possui mecanismos para liberar a memória

Gerenciamento de memória

Explícito:

- o programador aloca memória
- o programador é responsável por liberar a memória
- exemplos são C, C++, Pascal

Implícito:

- o programador cria variáveis de maneira restrita
- a linguagem possui mecanismos para liberar a memória
- exemplos são Python, Javascript, Java, Lisp, Matlab

Liberando memória automaticamente

Como saber se um nó não será mais usado?

Liberando memória automaticamente

Como saber se um nó não será mais usado?

- não é possível determinar isso!

Liberando memória automaticamente

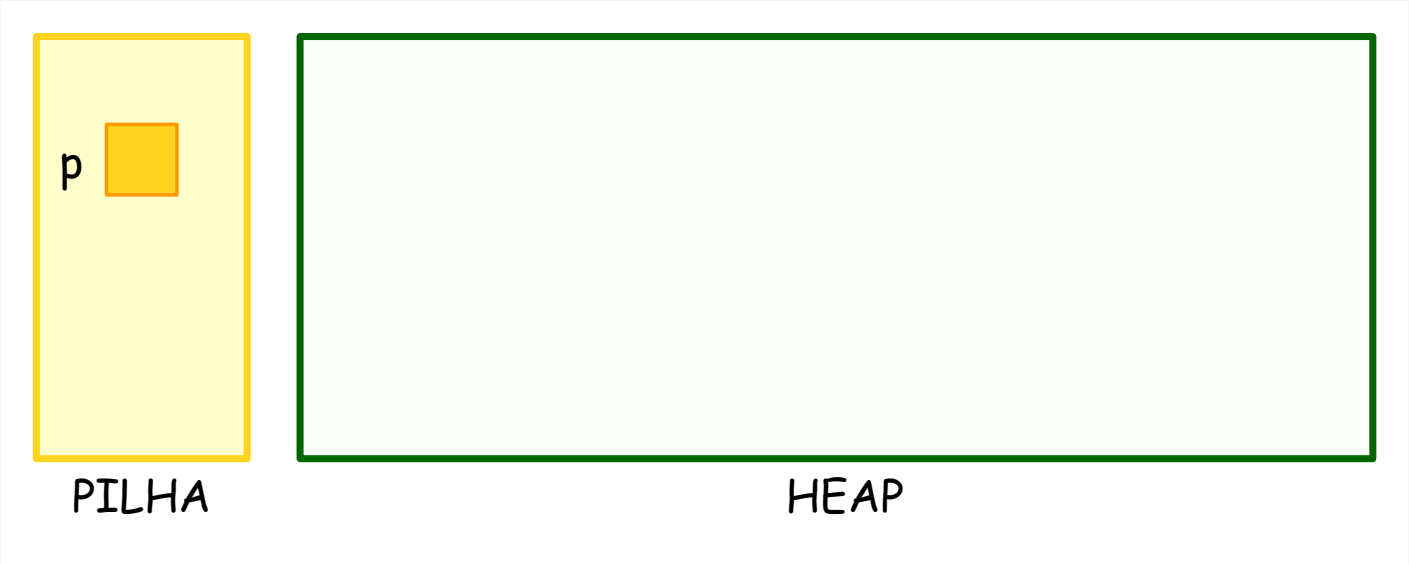
Como saber se um nó não será mais usado?

- não é possível determinar isso!
- mas podemos deduzir quais nós são acessíveis

Liberando memória automaticamente

Como saber se um nó não será mais usado?

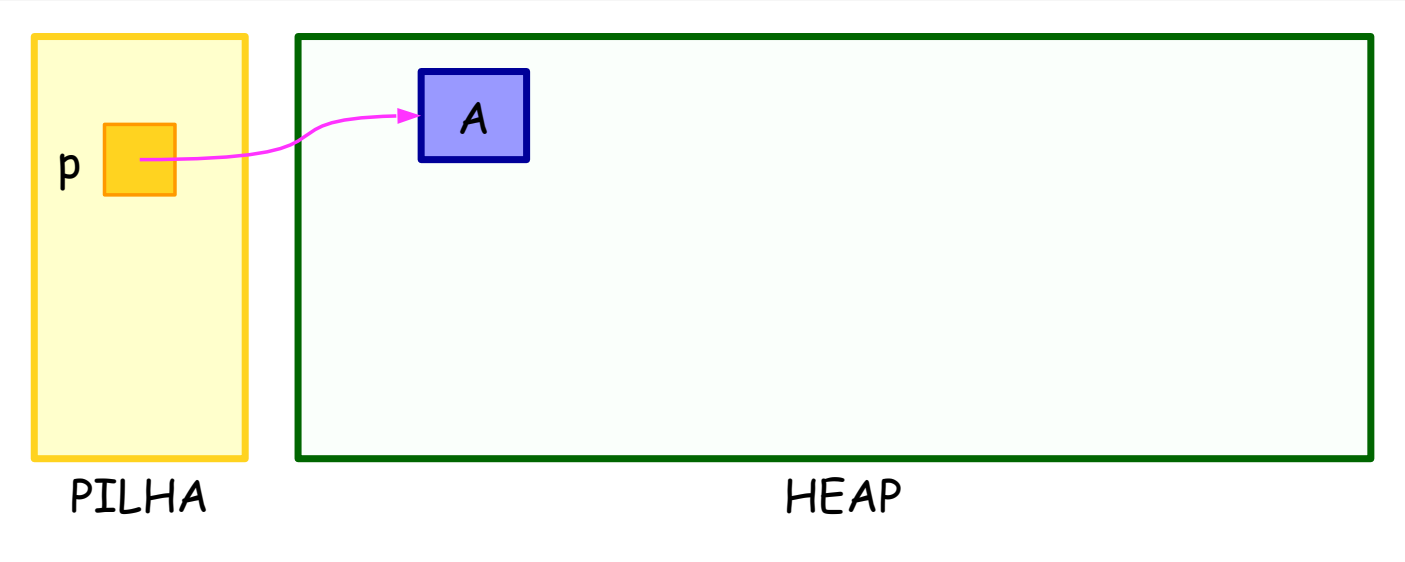
- não é possível determinar isso!
- mas podemos deduzir quais nós são acessíveis



Liberando memória automaticamente

Como saber se um nó não será mais usado?

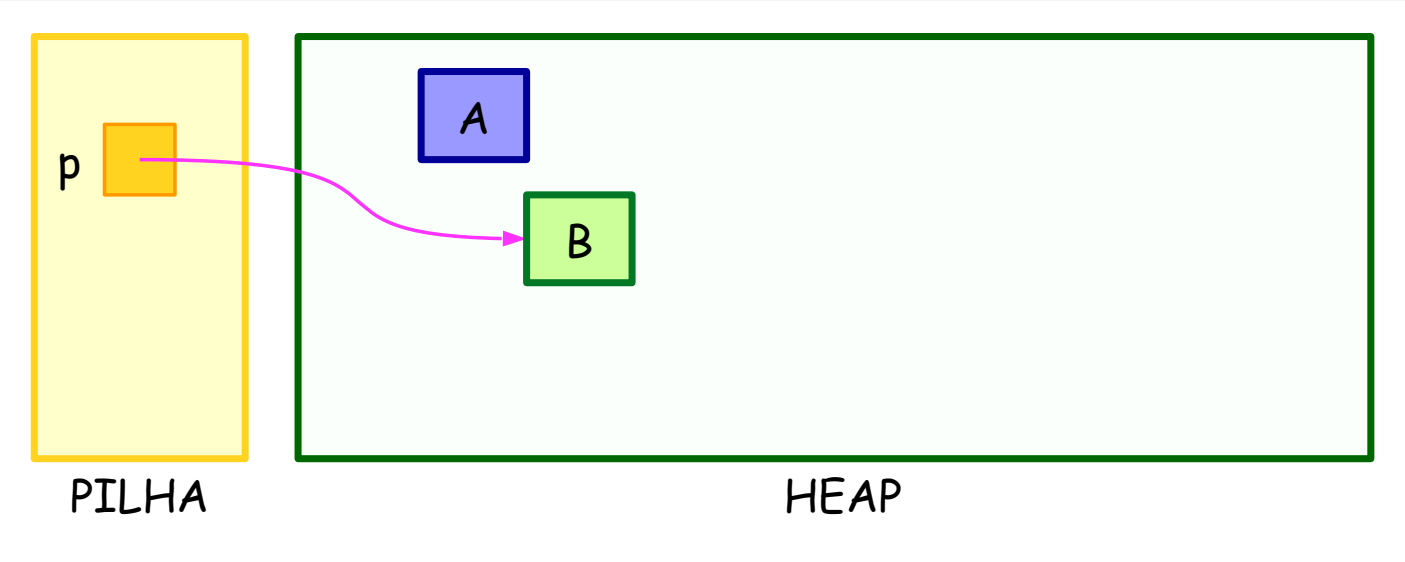
- não é possível determinar isso!
- mas podemos deduzir quais nós são acessíveis



Liberando memória automaticamente

Como saber se um nó não será mais usado?

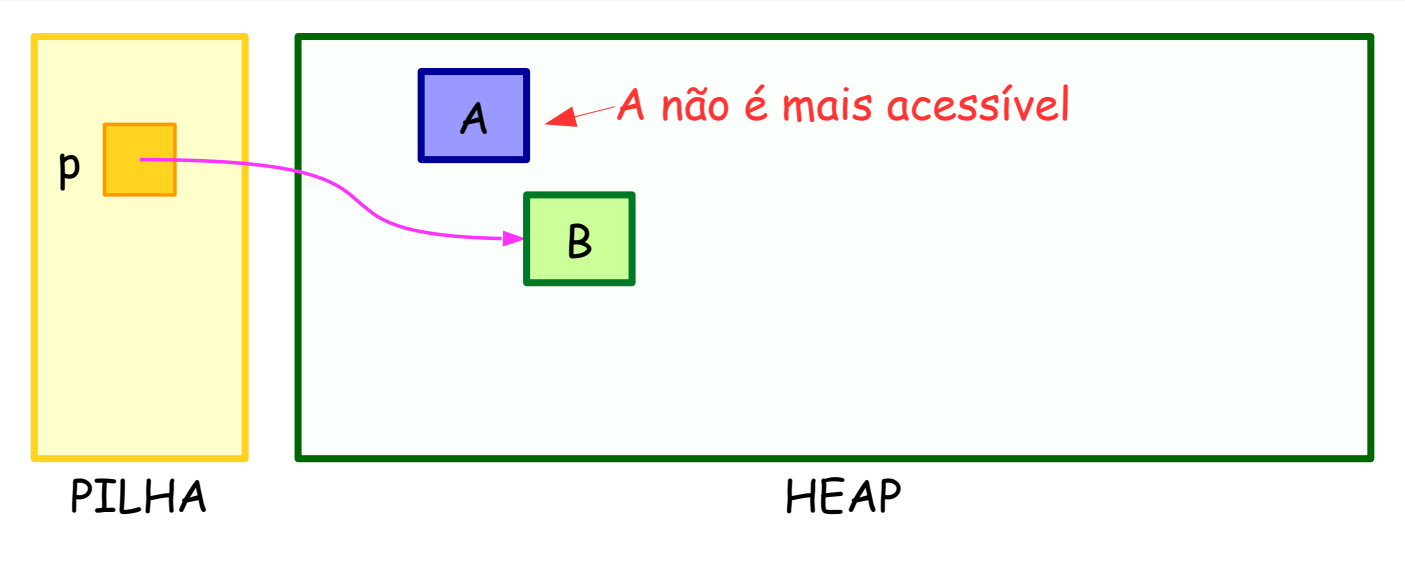
- não é possível determinar isso!
- mas podemos deduzir quais nós são acessíveis



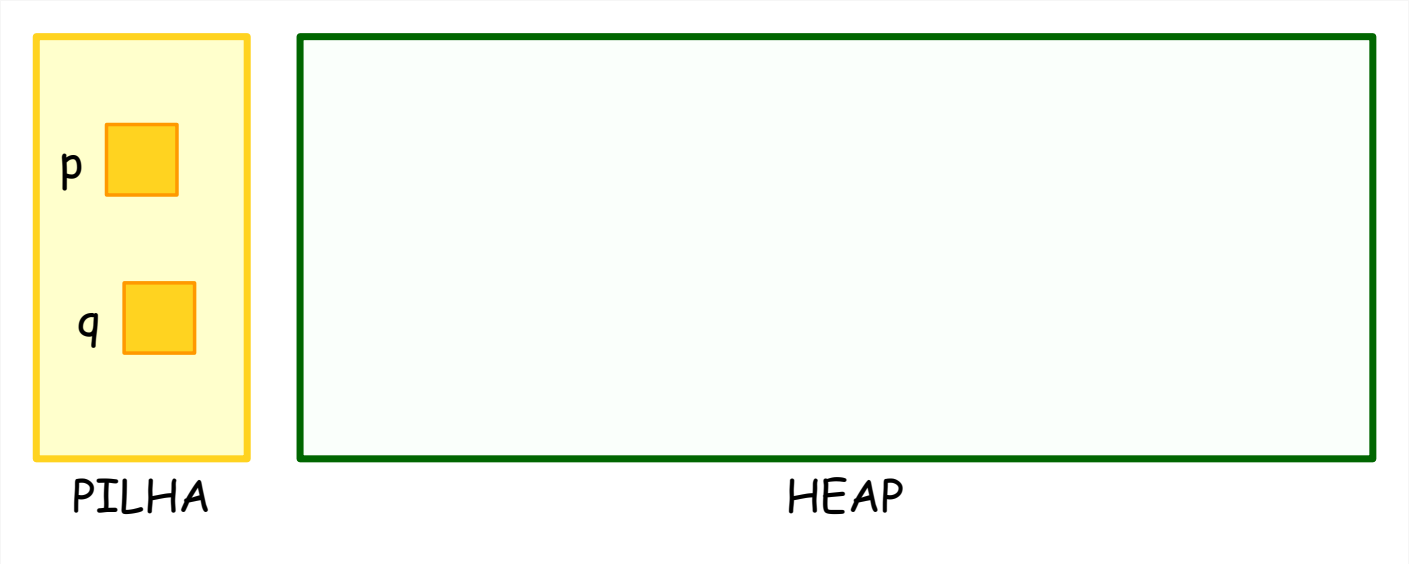
Liberando memória automaticamente

Como saber se um nó não será mais usado?

- não é possível determinar isso!
- mas podemos deduzir quais nós são acessíveis

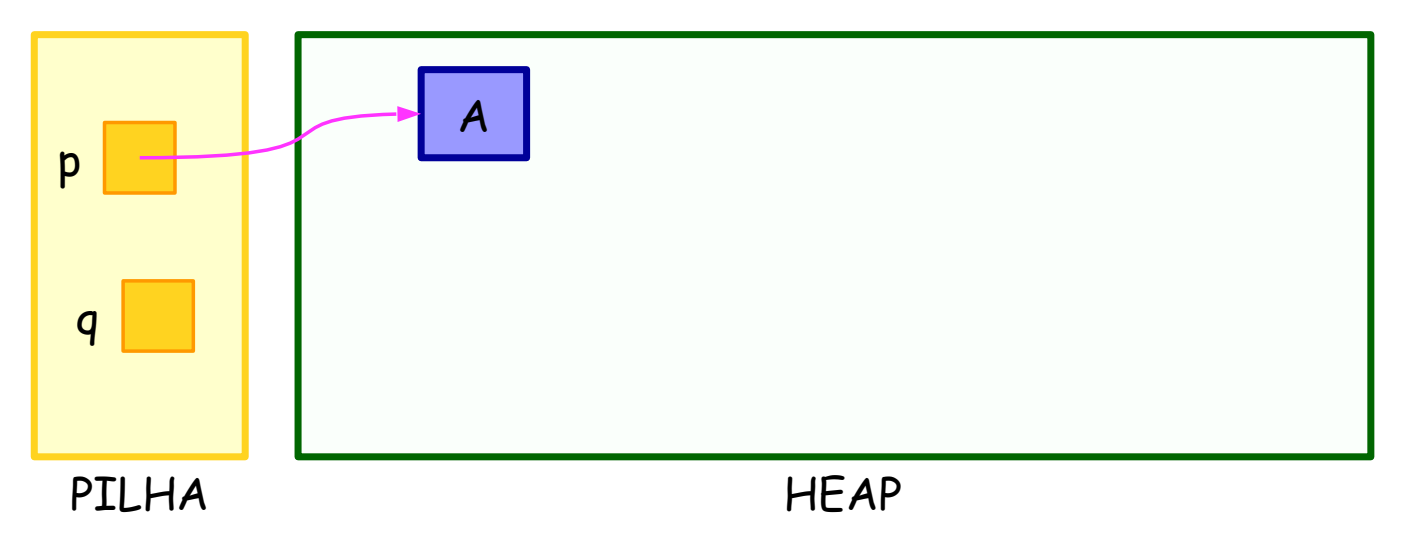


Um exemplo mais completo



Criando uma lista:

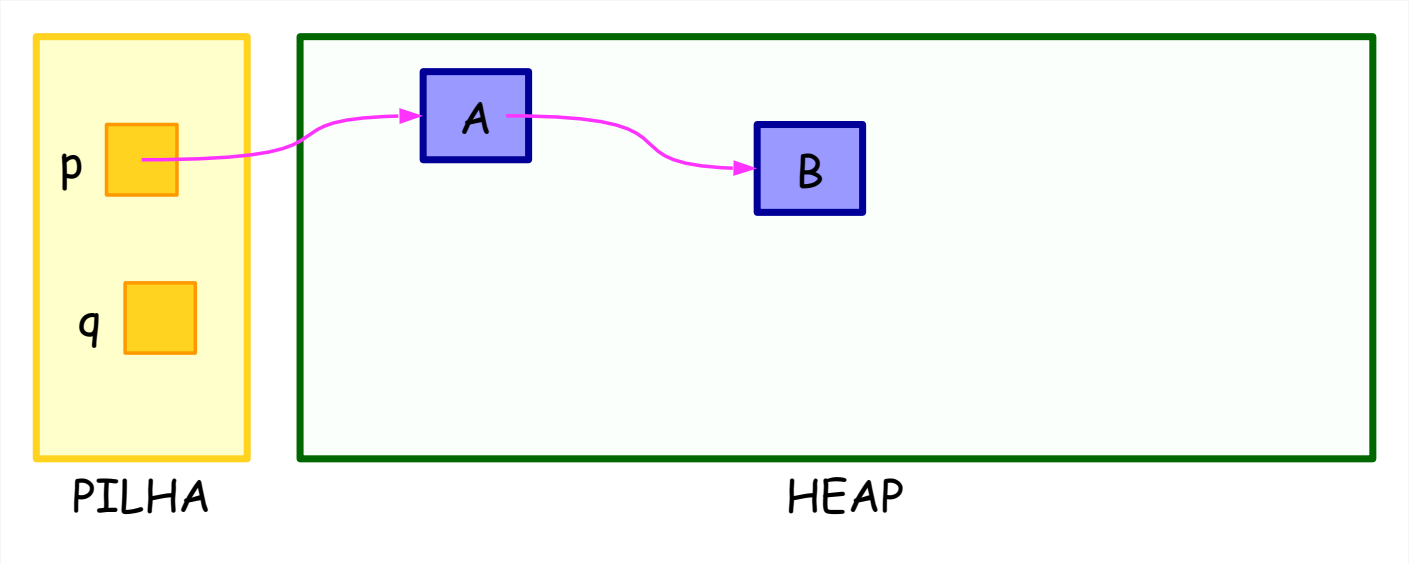
Um exemplo mais completo



Criando uma lista:

1. $p \leftarrow$ cria nó A

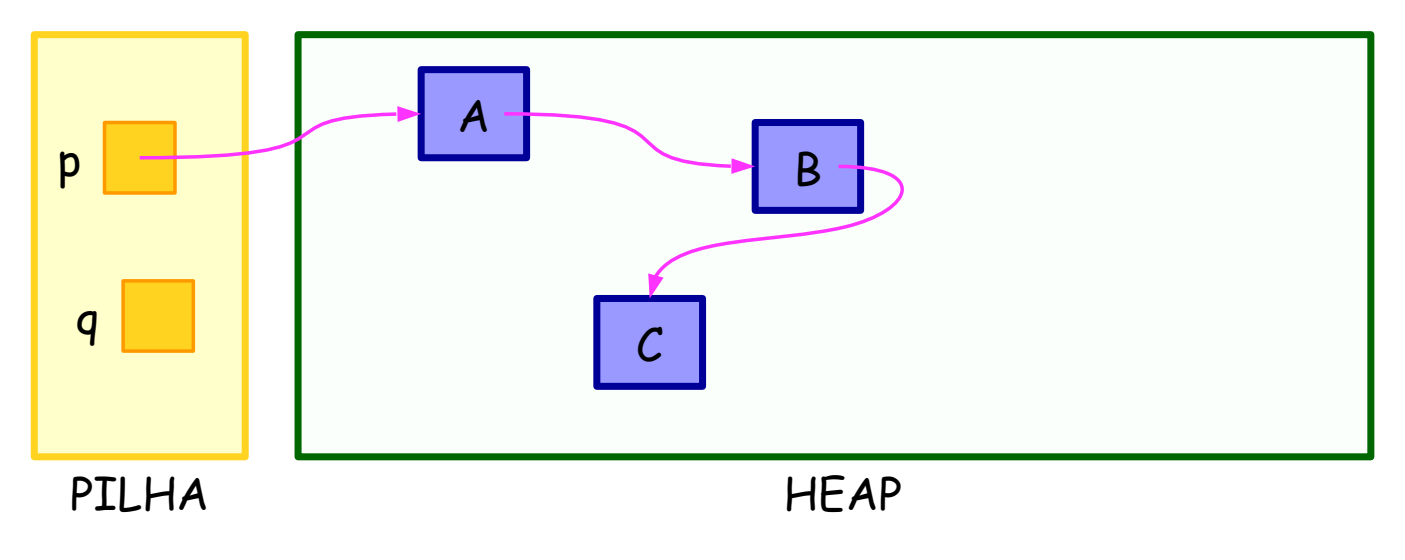
Um exemplo mais completo



Criando uma lista:

1. $p \leftarrow$ cria nó A
2. $p.prox \leftarrow$ cria nó B

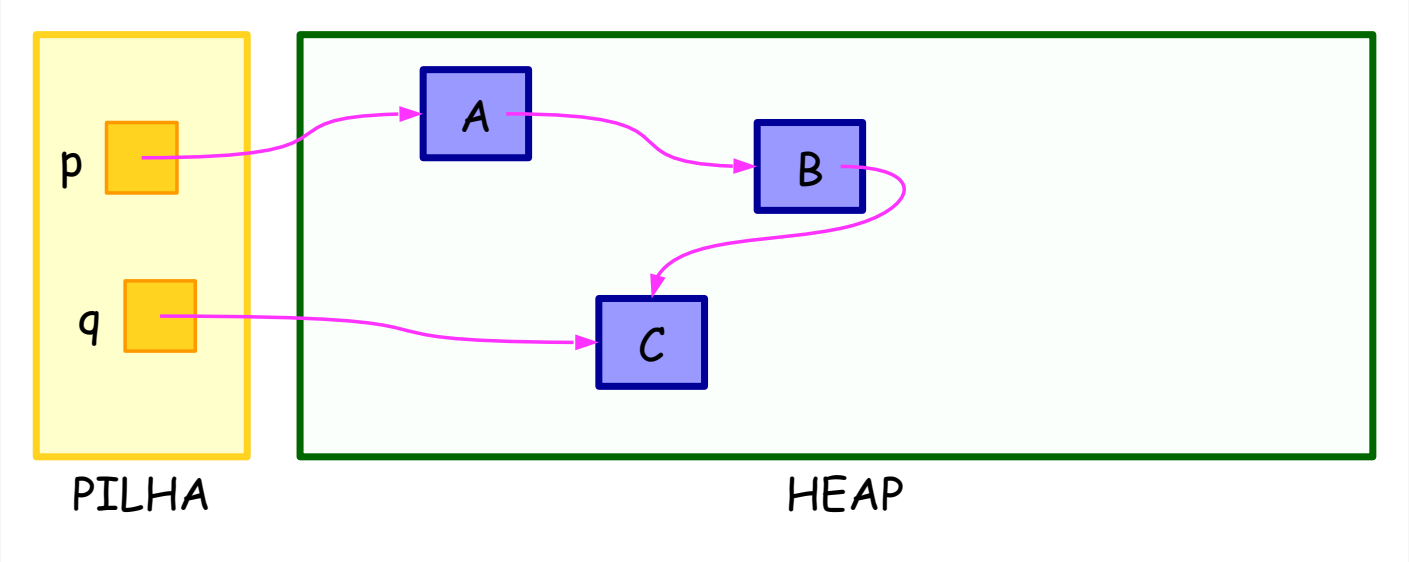
Um exemplo mais completo



Criando uma lista:

1. $p \leftarrow$ cria nó A
2. $p.prox \leftarrow$ cria nó B
3. $p.prox.prox \leftarrow$ cria nó C

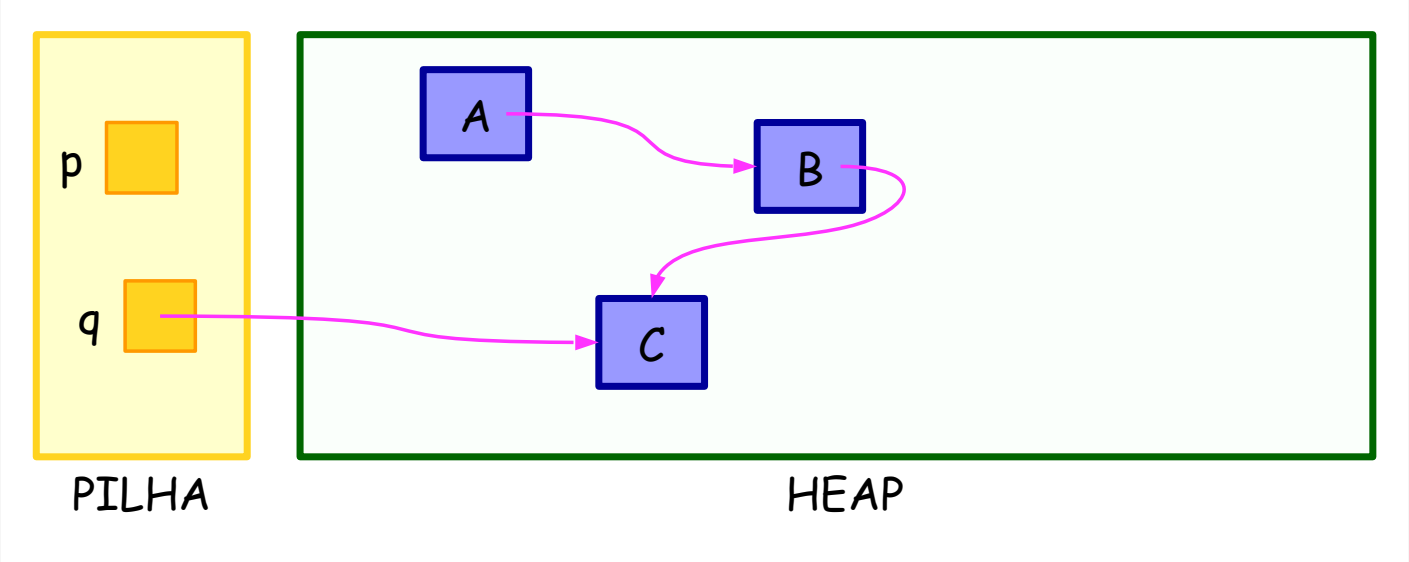
Um exemplo mais completo



Criando uma lista:

1. $p \leftarrow$ cria nó A
2. $p.prox \leftarrow$ cria nó B
3. $p.prox.prox \leftarrow$ cria nó C
4. $q \leftarrow p.prox.prox$

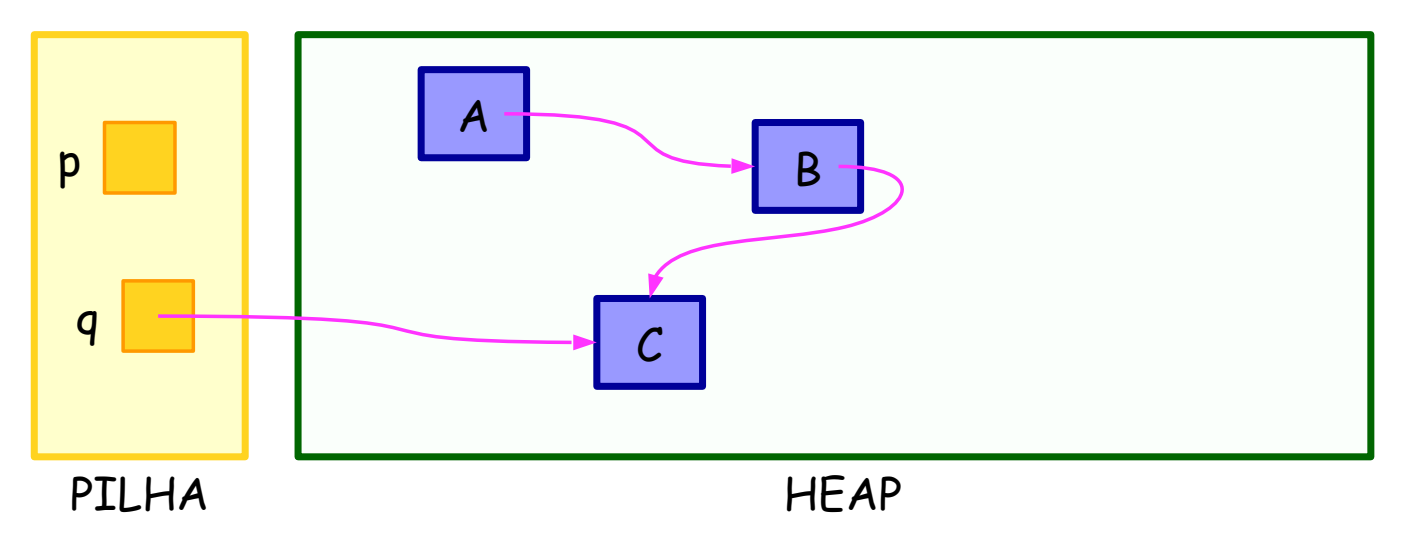
Um exemplo mais completo



Criando uma lista:

1. $p \leftarrow$ cria nó A
2. $p.prox \leftarrow$ cria nó B
3. $p.prox.prox \leftarrow$ cria nó C
4. $q \leftarrow p.prox.prox$
5. $p \leftarrow$ `NULL`

Um exemplo mais completo



Criando uma lista:

1. $p \leftarrow$ cria nó A
2. $p.prox \leftarrow$ cria nó B
3. $p.prox.prox \leftarrow$ cria nó C
4. $q \leftarrow p.prox.prox$
5. $p \leftarrow$ `NULL`

A partir desse momento, a memória de A e B pode ser liberada.

Coleta de lixo

Coleta de lixo

Coleta de lixo

Coleta de lixo

- é o algoritmo executado para liberar nós inacessíveis

Coleta de lixo

Coleta de lixo

- é o algoritmo executado para liberar nós inacessíveis
- a implementação depende da linguagem de programação

Coleta de lixo

Coleta de lixo

- é o algoritmo executado para liberar nós inacessíveis
- a implementação depende da linguagem de programação

Há duas estratégias principais:

Coleta de lixo

Coleta de lixo

- é o algoritmo executado para liberar nós inacessíveis
- a implementação depende da linguagem de programação

Há duas estratégias principais:

1. mark-and-sweep

Coleta de lixo

Coleta de lixo

- é o algoritmo executado para liberar nós inacessíveis
- a implementação depende da linguagem de programação

Há duas estratégias principais:

1. mark-and-sweep
2. contagem de referência

Algoritmo mark-and-sweep

Principais etapas:

Algoritmo mark-and-sweep

Principais etapas:

1. Marcar nós acessíveis

Algoritmo mark-and-sweep

Principais etapas:

1. Marcar nós acessíveis
2. Liberar nós não marcados

Algoritmo mark-and-sweep

Principais etapas:

1. Marcar nós acessíveis
2. Liberar nós não marcados
3. Compactar os blocos reservados

Algoritmo mark-and-sweep

Principais etapas:

1. Marcar nós acessíveis
2. Liberar nós não marcados
3. Compactar os blocos reservados

Observações:

Algoritmo mark-and-sweep

Principais etapas:

1. Marcar nós acessíveis
2. Liberar nós não marcados
3. Compactar os blocos reservados

Observações:

- pode ser executado em qualquer momento do programa

Algoritmo mark-and-sweep

Principais etapas:

1. Marcar nós acessíveis
2. Liberar nós não marcados
3. Compactar os blocos reservados

Observações:

- pode ser executado em qualquer momento do programa
- o algoritmo também precisa de memória para executar

Algoritmo mark-and-sweep

Principais etapas:

1. Marcar nós acessíveis
2. Liberar nós não marcados
3. Compactar os blocos reservados

Observações:

- pode ser executado em qualquer momento do programa
- o algoritmo também precisa de memória para executar
- percorremos o grafo de nós a partir das variáveis da pilha

Algoritmo mark-and-sweep

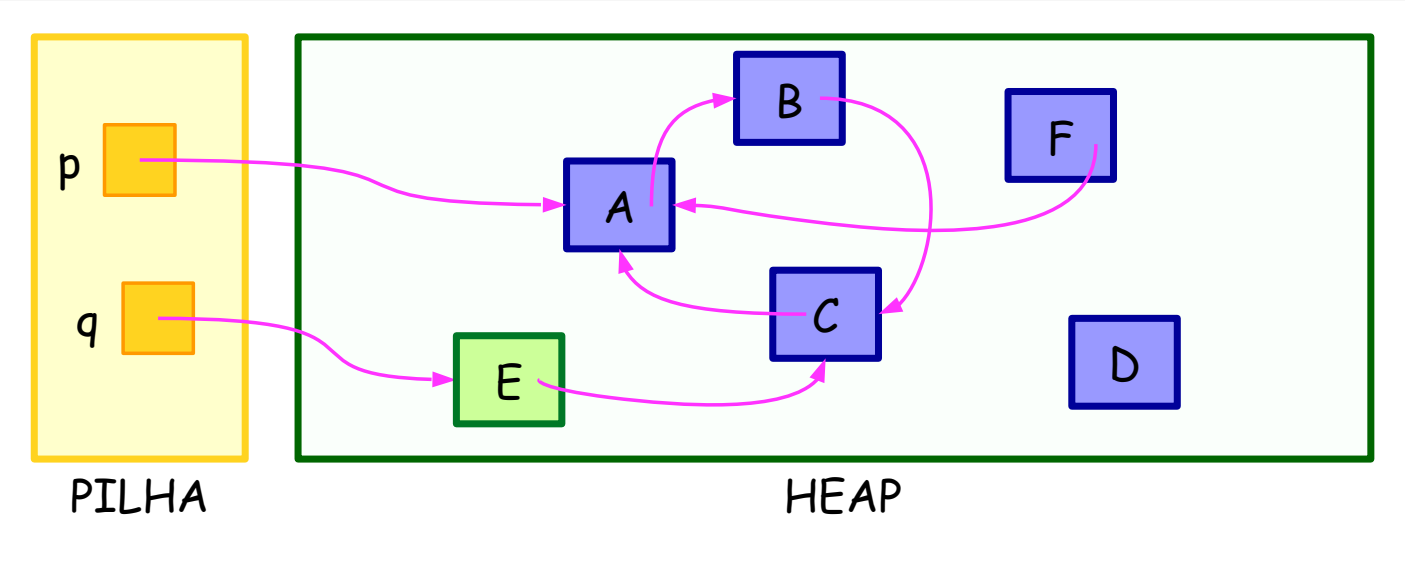
Principais etapas:

1. Marcar nós acessíveis
2. Liberar nós não marcados
3. Compactar os blocos reservados

Observações:

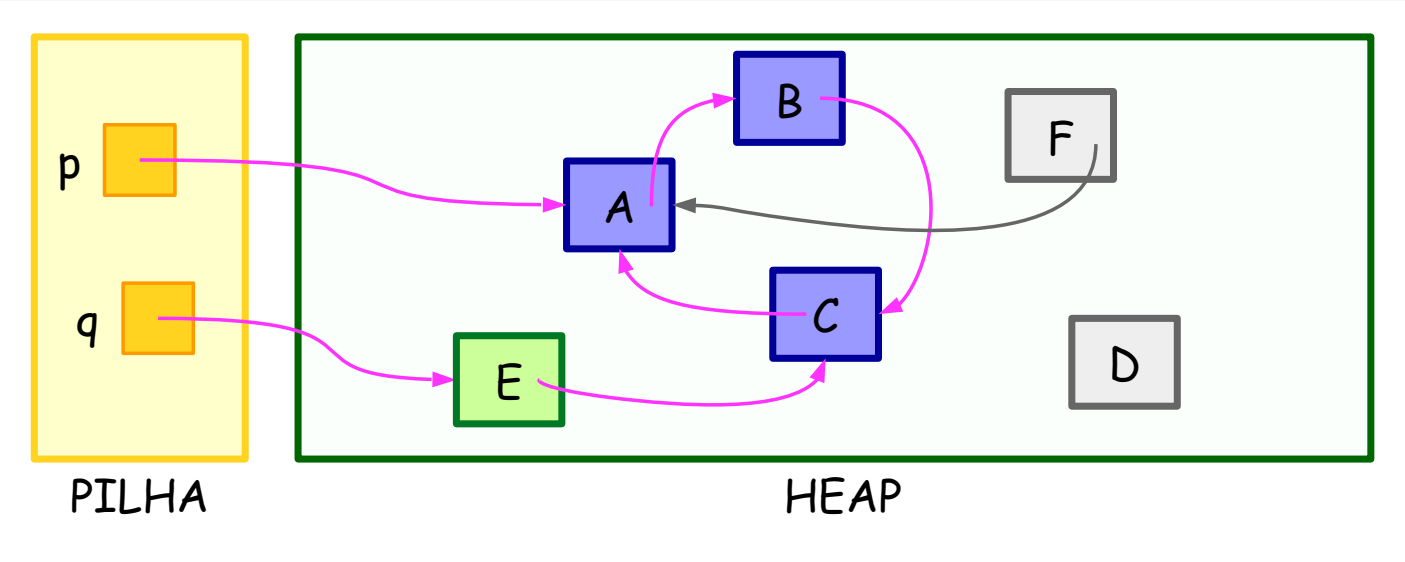
- pode ser executado em qualquer momento do programa
- o algoritmo também precisa de memória para executar
- percorremos o grafo de nós a partir das variáveis da pilha
- ao final da compactação, teremos um grande bloco livre

Exemplo de coleta de lixo



Procedimentos

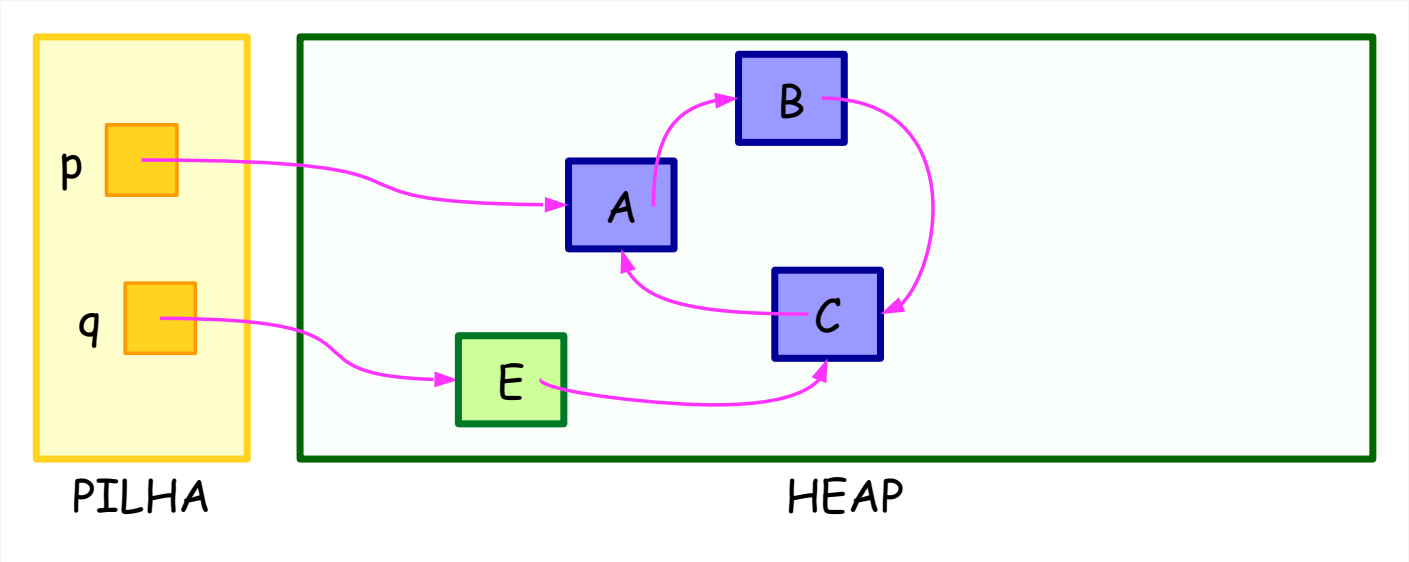
Exemplo de coleta de lixo



Procedimentos

1. Marcar nós acessíveis

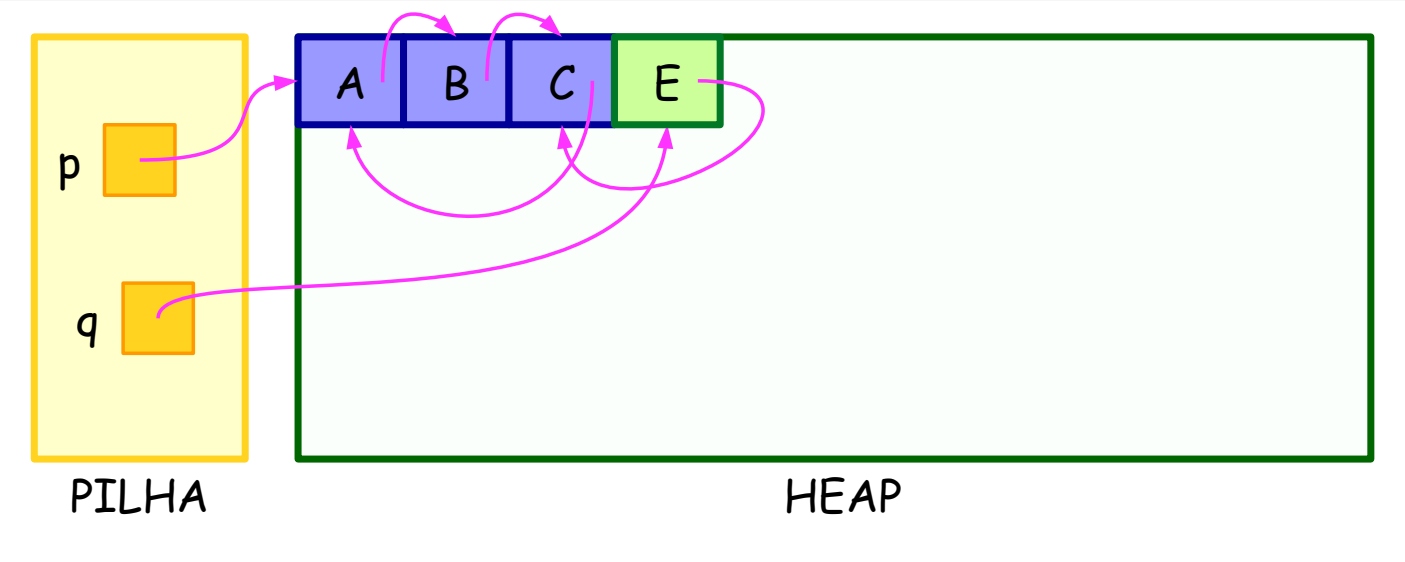
Exemplo de coleta de lixo



Procedimentos

1. Marcar nós acessíveis
2. Liberar nós inacessíveis

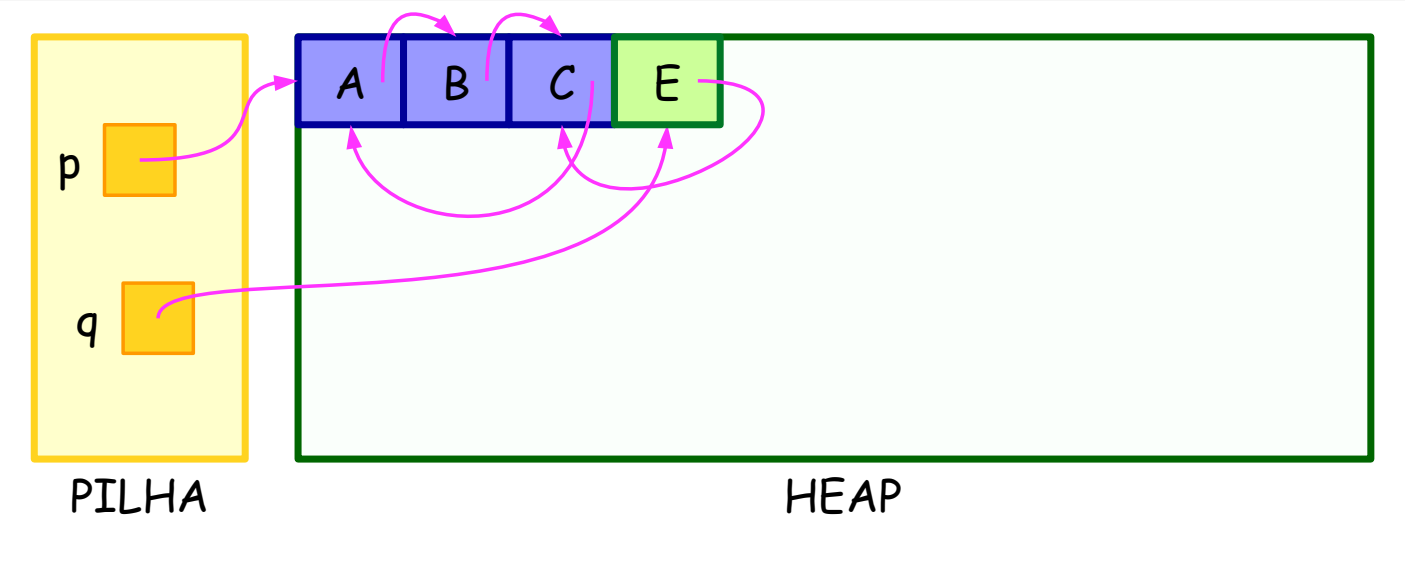
Exemplo de coleta de lixo



Procedimentos

1. Marcar nós acessíveis
2. Liberar nós inacessíveis
3. Compactar os blocos reservados

Exemplo de coleta de lixo



Procedimentos

1. Marcar nós acessíveis
2. Liberar nós inacessíveis
3. Compactar os blocos reservados
   (copia dados e atualiza ponteiros)

Contagem de referências

Ideia: guardamos o número de referências para um nó

Contagem de referências

Ideia: guardamos o número de referências para um nó

- Ao criarmos um nó:

Contagem de referências

Ideia: guardamos o número de referências para um nó

- Ao criarmos um nó:
  - reservamos espaço para o nó e um contador

Contagem de referências

Ideia: guardamos o número de referências para um nó

- Ao criarmos um nó:
  - reservamos espaço para o nó e um contador
  - inicializamos o contador com valor 1

Contagem de referências

Ideia: guardamos o número de referências para um nó

- Ao criarmos um nó:
  - reservamos espaço para o nó e um contador
  - inicializamos o contador com valor 1

- Ao copiarmos uma referência:

Contagem de referências

Ideia: guardamos o número de referências para um nó

- Ao criarmos um nó:
  - reservamos espaço para o nó e um contador
  - inicializamos o contador com valor 1
- Ao copiarmos uma referência:
  - incrementamos o contador do nó

Contagem de referências

Ideia: guardamos o número de referências para um nó

- Ao criarmos um nó:
  - reservamos espaço para o nó e um contador
  - inicializamos o contador com valor 1
- Ao copiarmos uma referência:
  - incrementamos o contador do nó
- Ao apagarmos uma referência:

Contagem de referências

Ideia: guardamos o número de referências para um nó

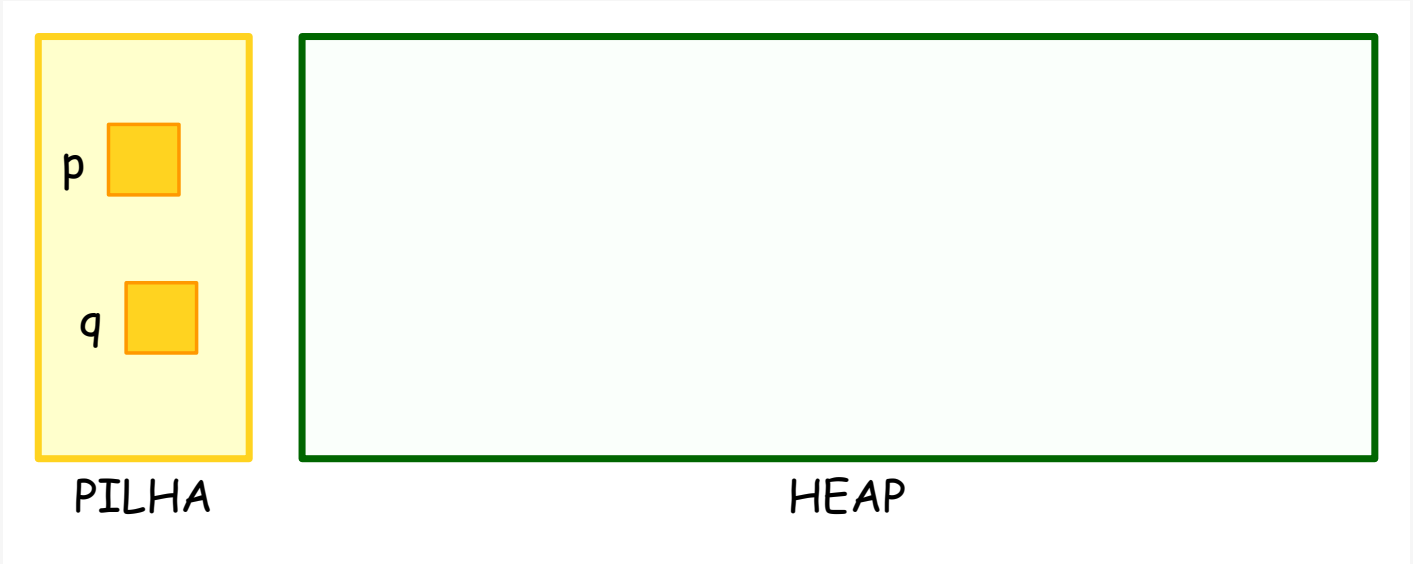
- Ao criarmos um nó:
  - reservamos espaço para o nó e um contador
  - inicializamos o contador com valor 1
- Ao copiarmos uma referência:
  - incrementamos o contador do nó
- Ao apagarmos uma referência:
  - decrementamos o contador do nó

Contagem de referências

Ideia: guardamos o número de referências para um nó

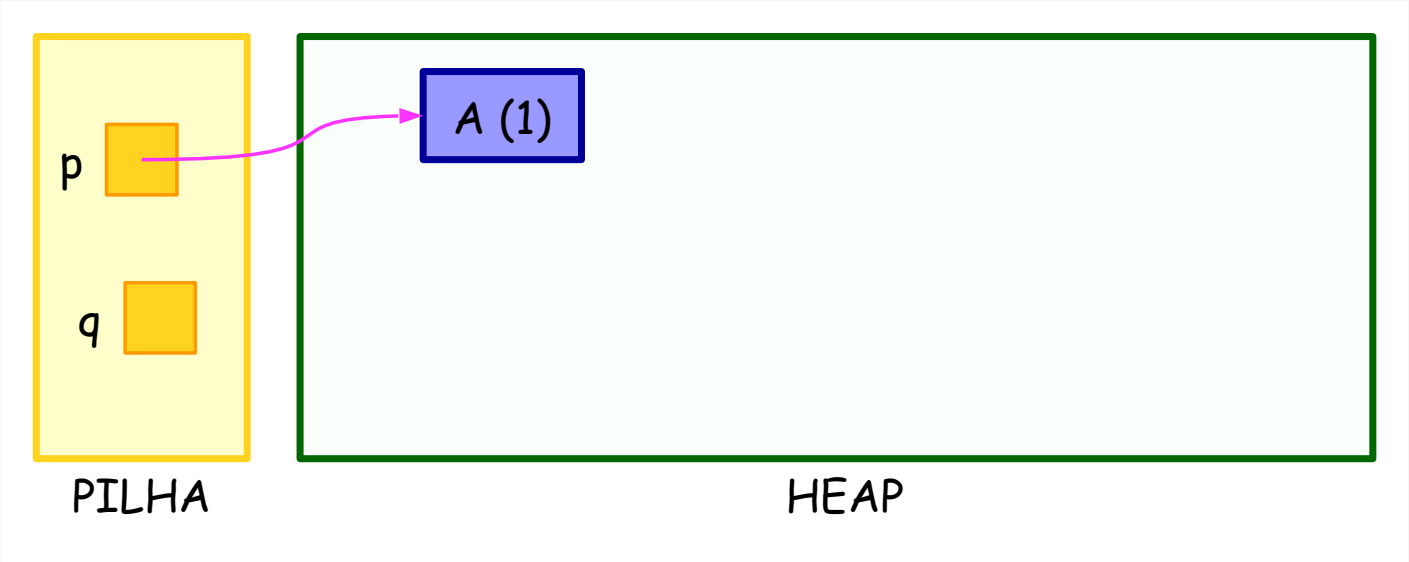
- Ao criarmos um nó:
  - reservamos espaço para o nó e um contador
  - inicializamos o contador com valor 1
- Ao copiarmos uma referência:
  - incrementamos o contador do nó
- Ao apagarmos uma referência:
  - decrementamos o contador do nó
  - liberamos o nós se o contador tiver valor 0

Exemplo de contagem de referências



Criando uma lista

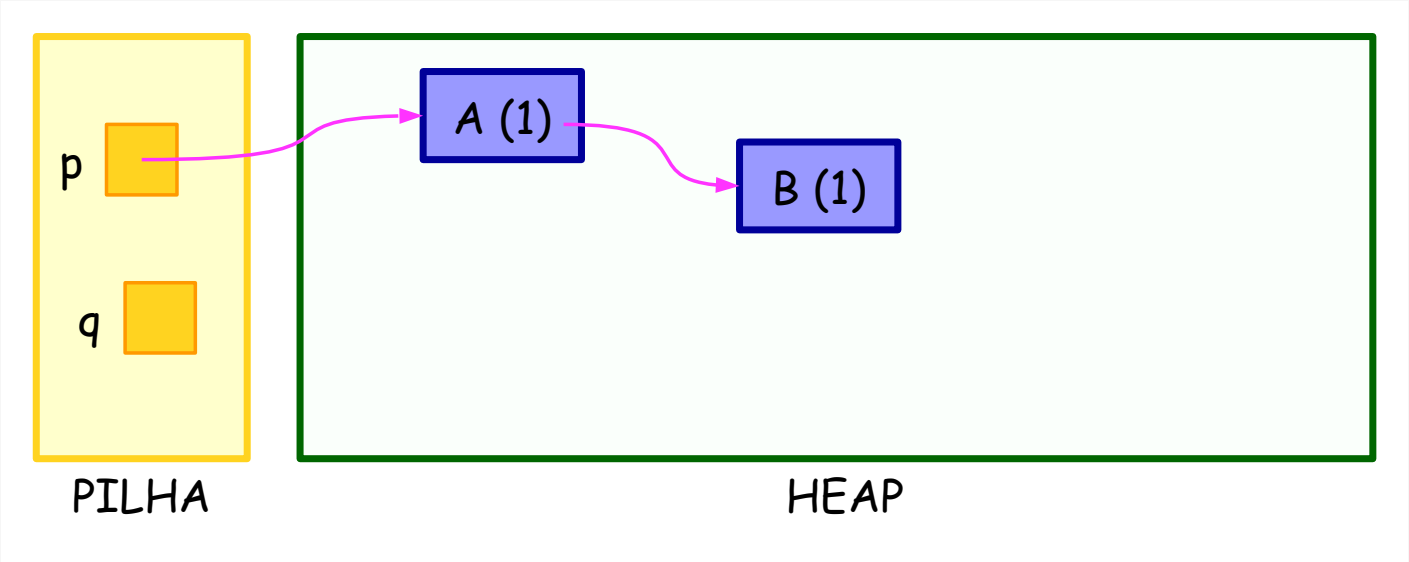
Exemplo de contagem de referências



Criando uma lista

1. $p \leftarrow$ cria nó A

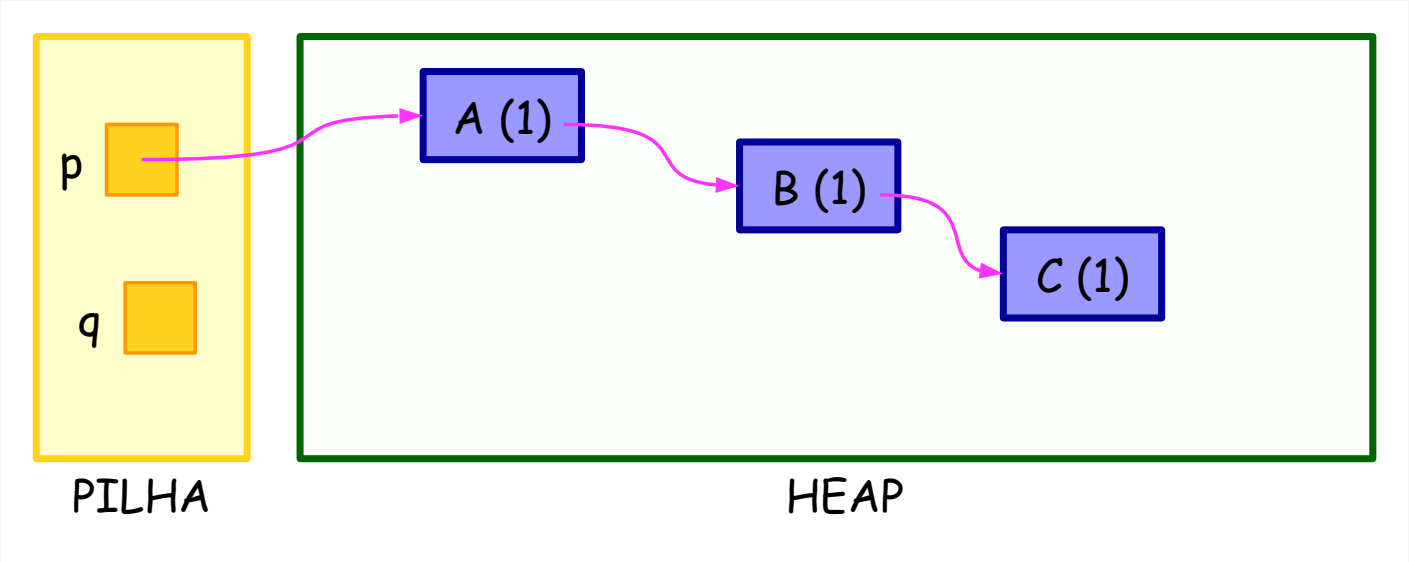
Exemplo de contagem de referências



Criando uma lista

1. $p \leftarrow$ cria nó A
2. $p.prox \leftarrow$ cria nó B

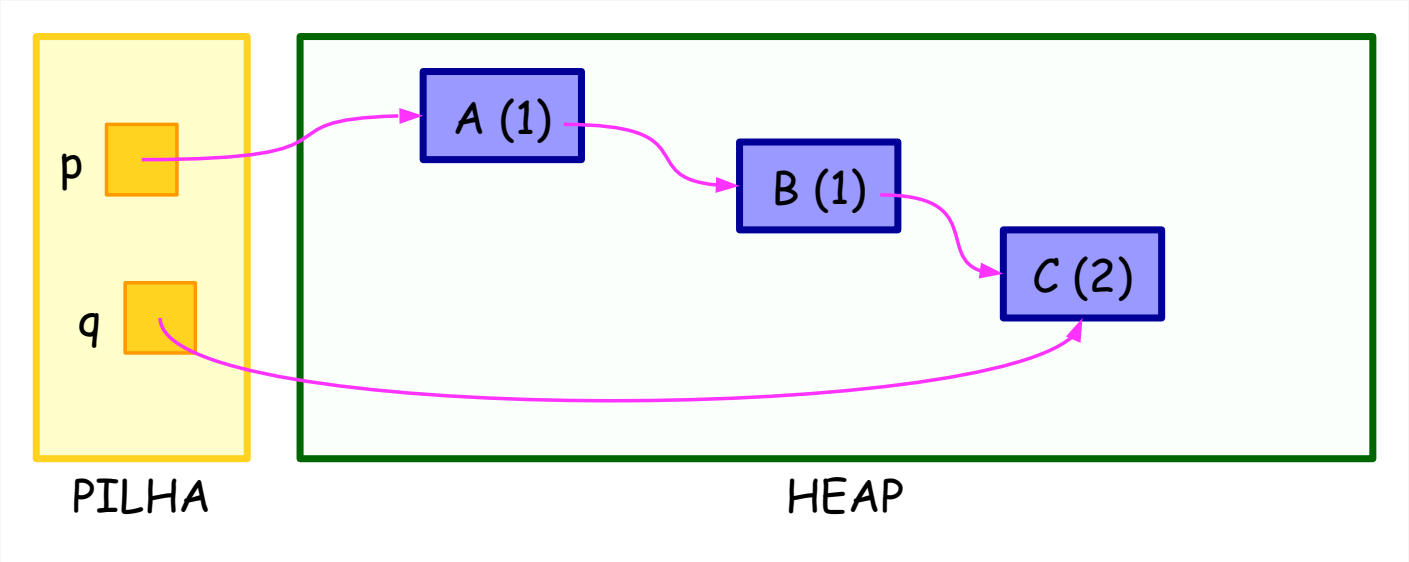
Exemplo de contagem de referências



Criando uma lista

1. $p \leftarrow$ cria nó A
2. $p.prox \leftarrow$ cria nó B
3. $p.prox.prox \leftarrow$ cria nó C

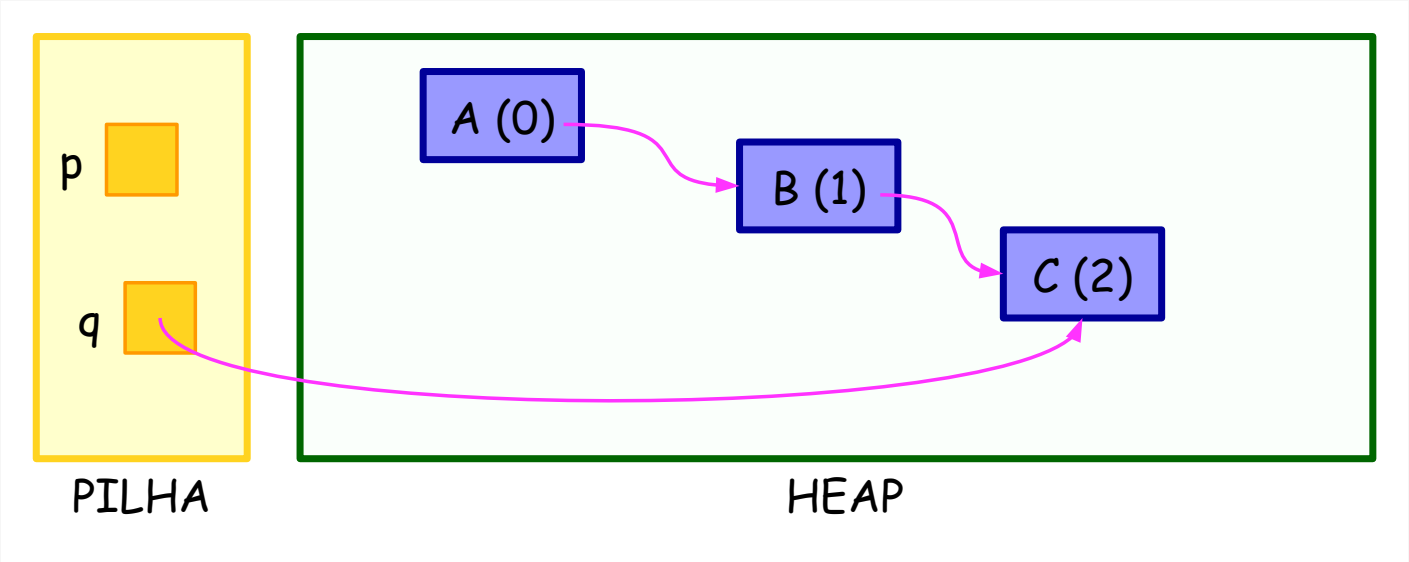
Exemplo de contagem de referências



Criando uma lista

1. $p \leftarrow$ cria nó A
2. $p.prox \leftarrow$ cria nó B
3. $p.prox.prox \leftarrow$ cria nó C
4. $q \leftarrow p.prox.prox$

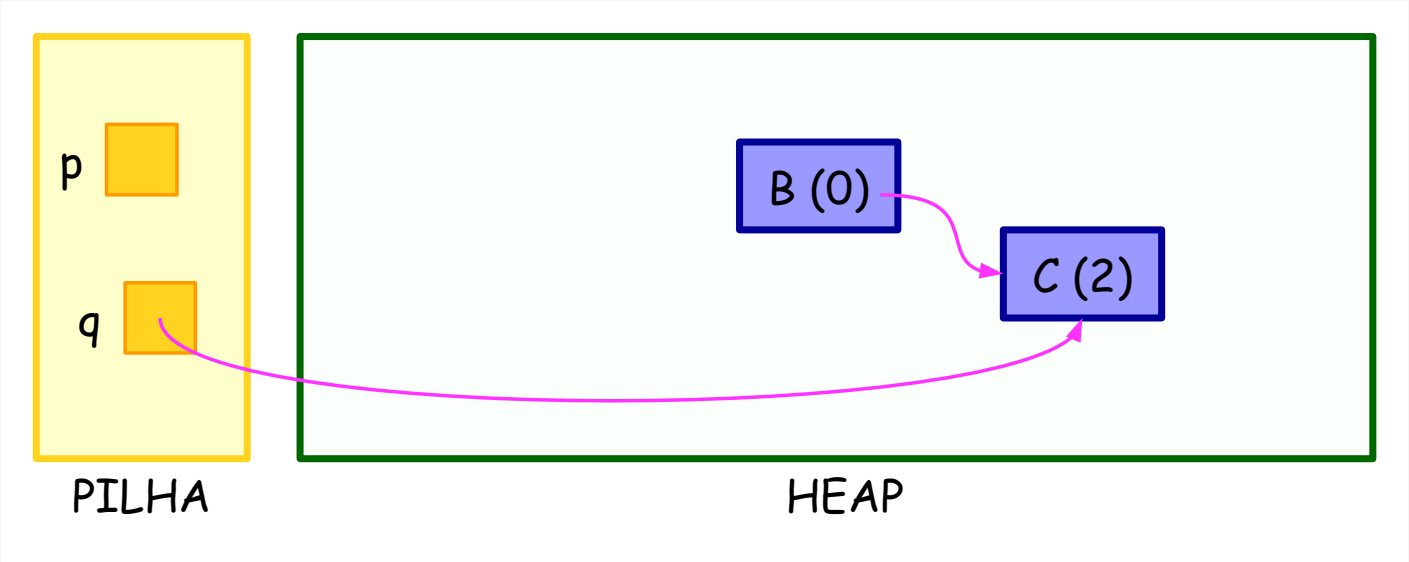
Exemplo de contagem de referências



Criando uma lista

1. $p \leftarrow$ cria nó A
2. $p.prox \leftarrow$ cria nó B
3. $p.prox.prox \leftarrow$ cria nó C
4. $q \leftarrow p.prox.prox$
5. $p \leftarrow$ `NULL`

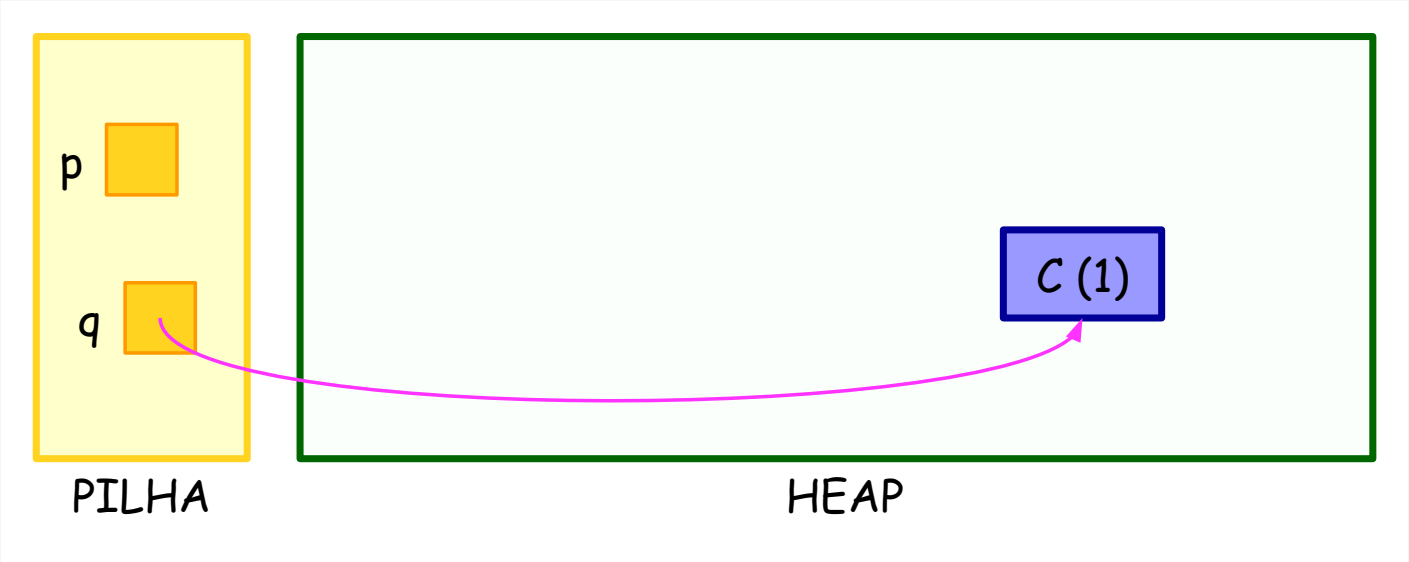
Exemplo de contagem de referências



Criando uma lista

1. $p \leftarrow$ cria nó A
2. $p.prox \leftarrow$ cria nó B
3. $p.prox.prox \leftarrow$ cria nó C
4. $q \leftarrow p.prox.prox$
5. $p \leftarrow$ `NULL`

Exemplo de contagem de referências



Criando uma lista

1. $p \leftarrow$ cria nó A
2. $p.prox \leftarrow$ cria nó B
3. $p.prox.prox \leftarrow$ cria nó C
4. $q \leftarrow p.prox.prox$
5. $p \leftarrow$ `NULL`

Dúvidas?