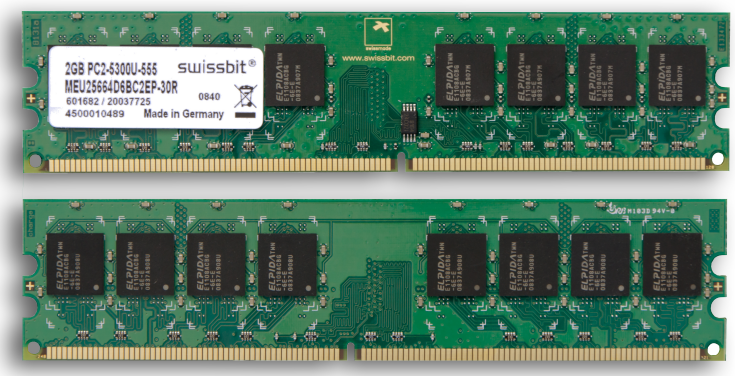
Minne

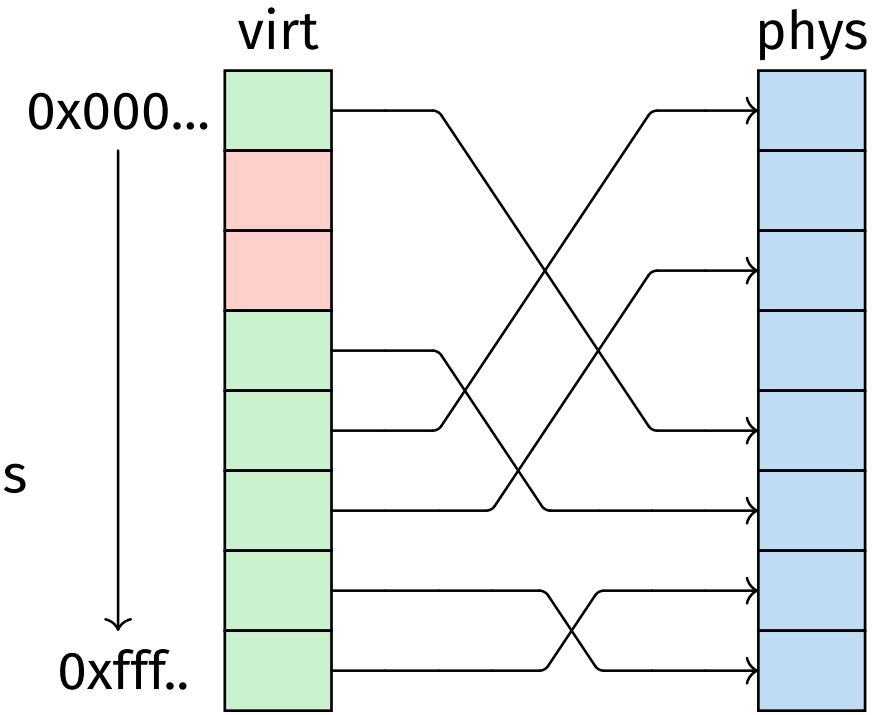
Fysiskt minne

- Eller bara *RAM-minne*, stickorna som sätts i moderkortet
- Datorns arbetsminne
- Basically en array av bytes som kan läsas och skrivas



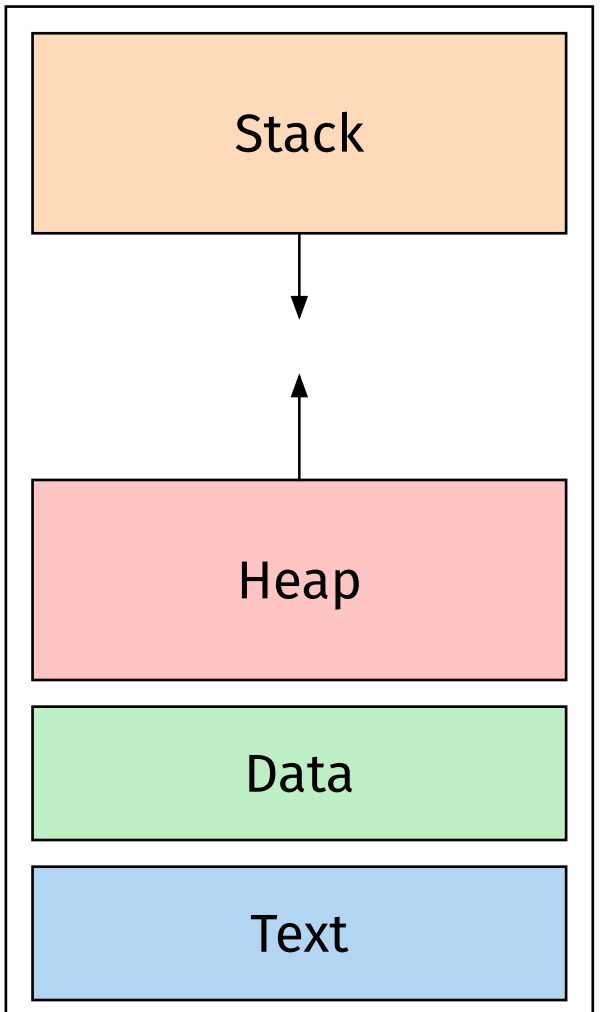
Virtuellt minne

- En adressrymd av *virtuella adresser*
- Det som program ser när de körs
- Varje process har en egen virtuell adressrymd
- Översätts till fysiska adresser när de används
- En process ser bara det fysiska minne som finns *mappat* i dess virtuella adressrymd
- Skyddar program mot sig själva och varandra



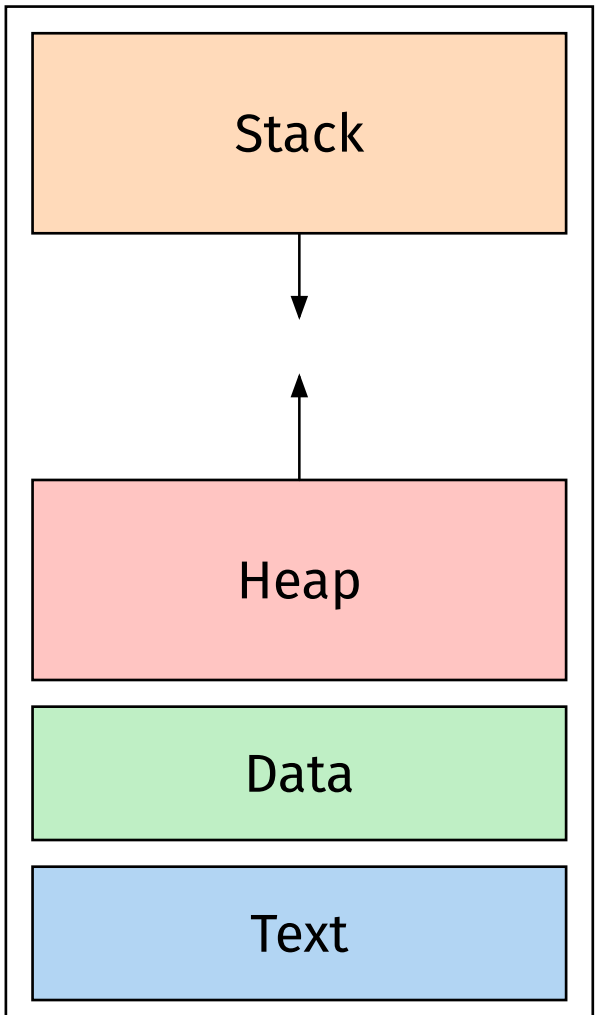
Memory segments

- I virtuellt minne ser minnesrymden ut så här
- Består av virtuella sidor som måste mappas till fysiska sidor av operativsystemet
- För att växa heapen krävs systemanrop



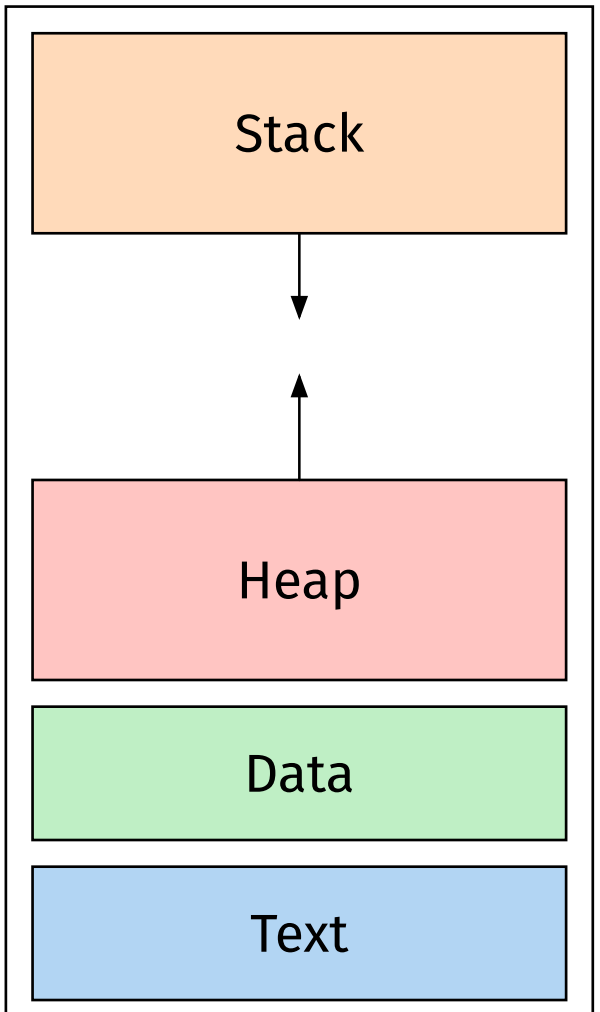
Memory segments

- Systemanrop som `sbrk`, `mmap` (Unix), `HeapAlloc`, `VirtualAlloc` (Windows) används för att uttöka processens minnesrymd
- Detta sköts dock främst av standardbiblioteksfunktioner som `malloc` (C) eller `new` (C++).



Memory segments

- Används `malloc` för att allokera sidor?
- Nej, `malloc`-implementationen är en fullfjädrad algoritm för att tillhanda-hålla minnesregioner av olika storlekar till användaren.



Minnesallokering

Minnesallokering

- Det finns flera krav som kan ställas på en allokeringsalgoritm
- Vi vill kunna allokera och frigöra minne effektivt
- Vi vill oftast kunna allokera regioner av olika storlekar
- Allokering och frigöring kan ske i godtycklig ordning
- Vi vill undvika *extern* och *intern fragmentering*

Intern fragmentering

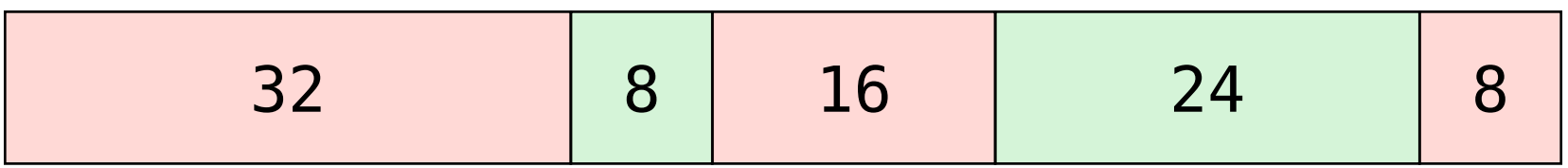
- *Intern fragmentering* innebär att den allokerade regionen är större än vad som faktiskt används
- Exempel:

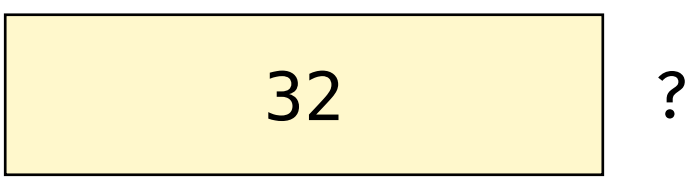
```
char message[] = "Tjena!";
char *copy = malloc(sizeof(message));
memcpy(copy, message, sizeof(message));
```

- `malloc` allokerar 8 bytes (på ett 64-bitars system), men storleken på strängen är 7
- Detta sker för att `malloc` utgår från maximal alignment

Extern fragmentering

- *Extern fragmentering* innebär att det finns tillräckligt med ledigt minne för att rymma en begäran, men det är utspritt.



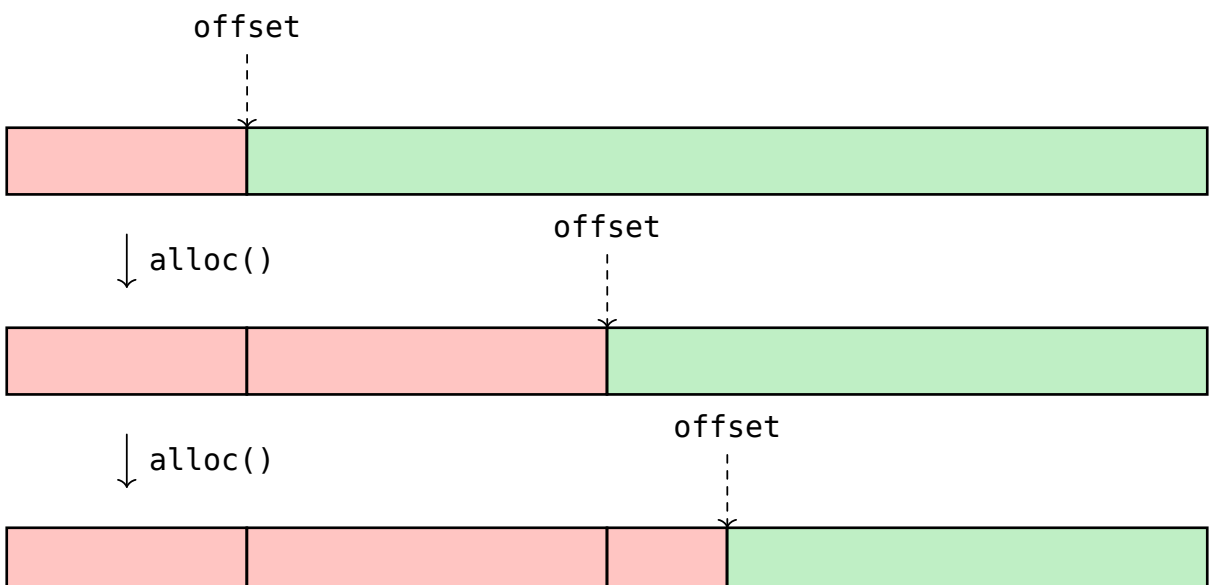


?

Algoritmer för minnesallokering

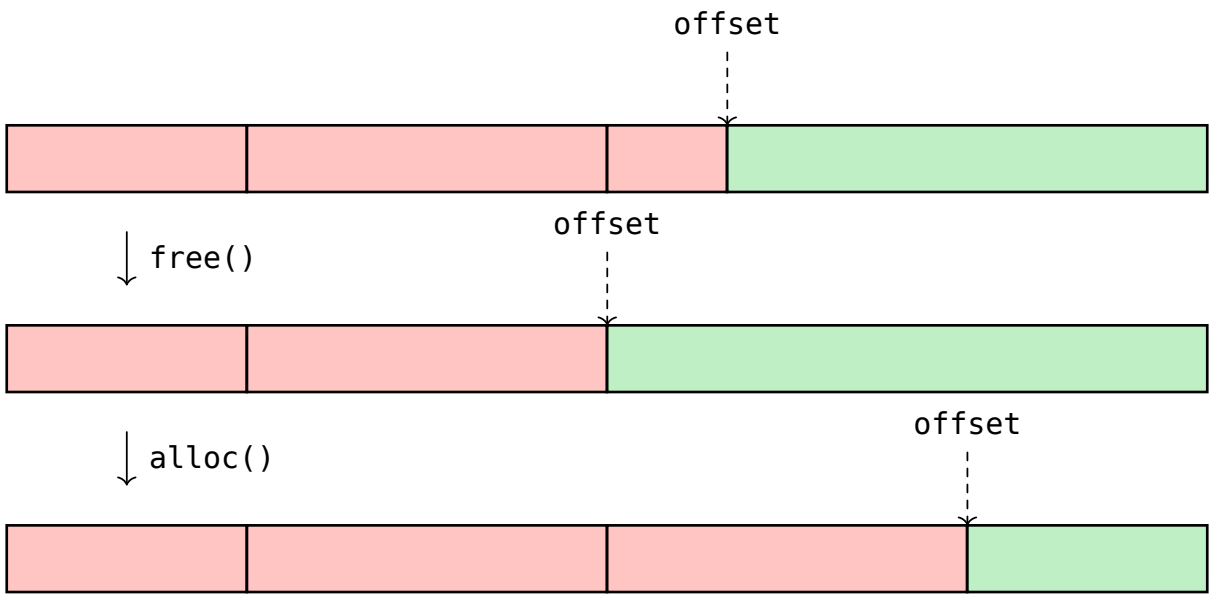
Linear allocator

- Allokerar minne från vänster till höger
- Kallas även *bump allocator*
- Komplexitet? Problem?



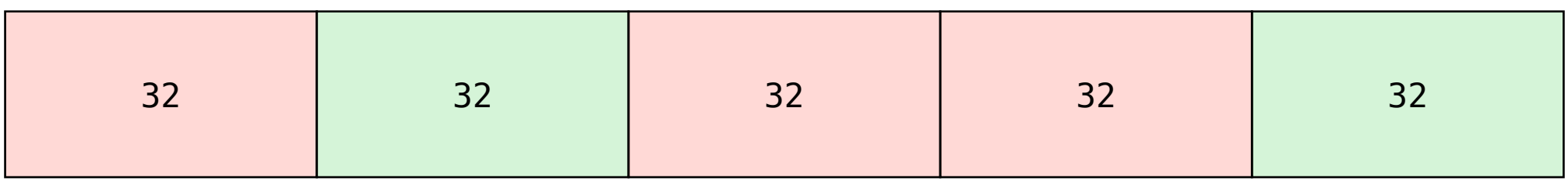
Stack allocator

- Linear allocator där vi även kan frigöra det översta blocket



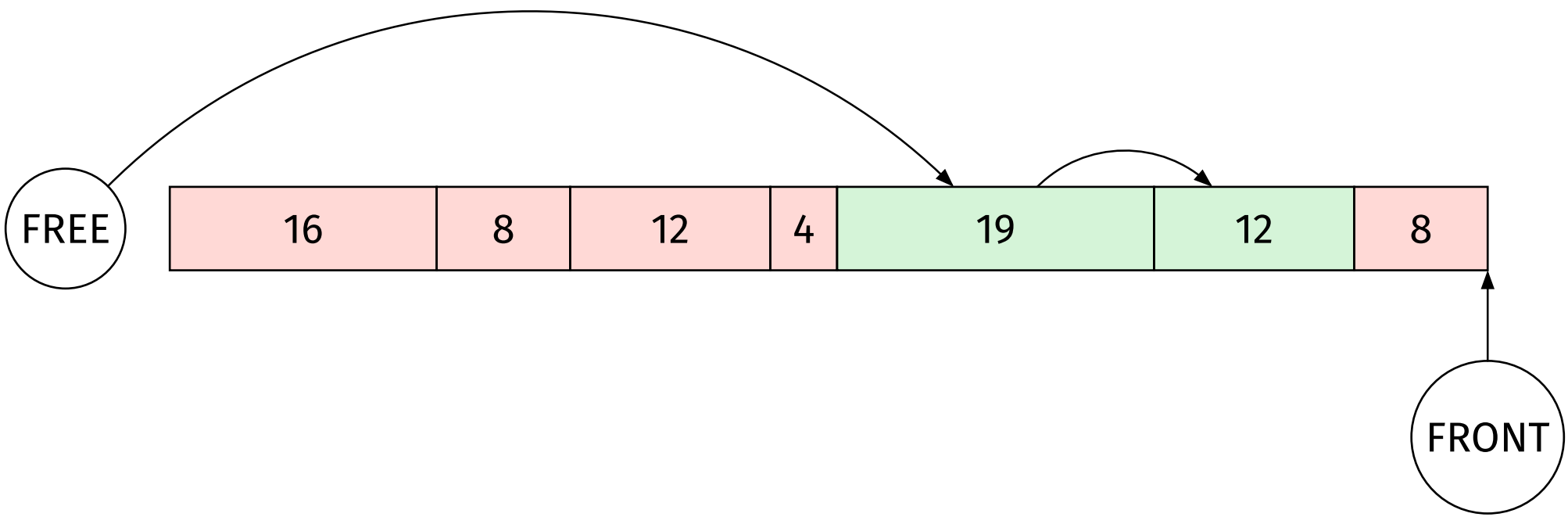
Pool allocator

- Alla allokeringar är av samma storlek
- Användbart om det är samma typ som alltid allokeras



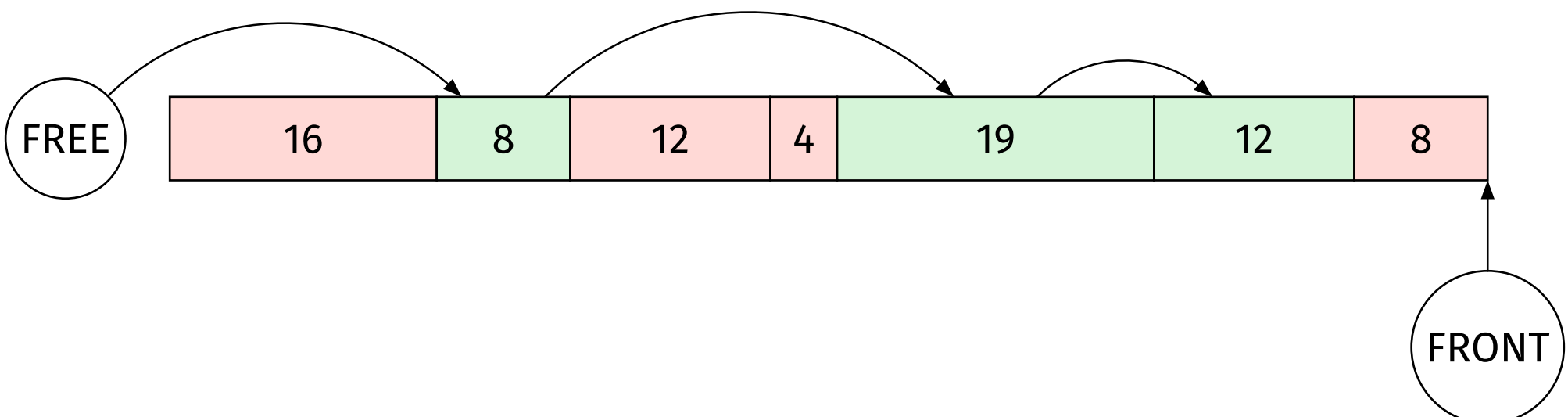
Free list

- Allokeringen växer linjärt
- När vi frigör minne, lägg till i en *free list*
- När vi ska allokera, kolla listan först



Free list

- Allokeringen växer linjärt
- När vi frigör minne, lägg till i en *free list*
- När vi ska allokera, kolla listan först



Free list

- För att välja ett block i listan finns olika metoder
- T.ex. **best fit**, **worst fit**, **first fit** och **next fit**

First fit

- Välj första block i listan som rymmer begäran
- Tidseffektiv, men kan leda till extern fragmentering

Next fit

- Som **First fit** men vi börjar från där vi senaste slutade i listan

Best fit & Worst fit

Best fit

- Välj det minsta block som rymmer begäran
- Leder till små ofta oanvändbara fria block

Worst fit

- Välj det största blocket
- Leder till större överblivna block när vi splittar

Buddy allocator

- Minnesrymden delas reskursivt i två för att rymma begäran
- När vi frigör minne kan vi slå ihop block
- Extern fragmentering kan minskas med chaining

<table>
<tr><td colspan="8">4096</td></tr>
<tr><td colspan="4">2048</td><td colspan="4">2048</td></tr>
<tr><td colspan="2">1024</td><td colspan="2">1024</td><td colspan="2">1024</td><td colspan="2">1024</td></tr>
<tr><td>512</td><td>512</td><td>512</td><td>512</td><td>512</td><td>512</td><td>512</td><td>512</td></tr>
</table>

Memory arenas

- Arenas innbär att man allokerar mycket minne som sedan frigörs i ett svep
- Användbart i t.ex. spelloopar där man skapar objekt som ska frigöras i slutet av varje frame

Läxa

Läxa

- Implementera två typer av memory allocators
- Använd er av något systemnivåspråk, helst C, C++ eller Zig
- Det går bra att först allokera en region med `malloc`, och låta er allocator rå om den regionen
- Annars kan ni försöka att allokera pages direkt med systemanrop
- En kul utmaning är att försöka ersätta `malloc`
- På Linux går det att ”overrida” libc-versionen med

```
$ LD_PRELOAD=./din_malloc.so <nåt kommando>
```