Buddy Allocation Algorithm

Let the leaf size be L and the memory size M' (effective)

$$n(leaf) = \frac{M'}{L}$$

Possible chunk sizes, S

$$= \left\{ |chunk| \;\middle|\; |chunk| = L \cdot 2^{k} \right\}$$

$$|chunk|_{max} = M'$$

$$\Rightarrow L \cdot 2^{k_{max}} = M'$$

$$\Rightarrow k_{max} = \log_2\left(\frac{M'}{L}\right) = \log_2\left(n(leaf)\right)$$

$$\therefore \quad 0 \leq k \leq \log_2\{n(leaf)\} \quad \text{and } k \in \mathbb{Z}$$

and $$n(S) = \log_2\{n(leaf)\} + 1 = k_{max} + 1$$

Consider the chunk C with $k = k$

$$|C_r| = L \cdot 2^{k_r}$$

$$\Rightarrow \frac{M}{|C_r|} = \frac{L \cdot 2^{k_{max}}}{L \cdot 2^{k_r}} = 2^{(k_{max} - k_r)}$$

$$\Rightarrow \frac{M}{|C_r|} = 1 \ll (k_{max} - k_r)$$

since $x \ll n = x \cdot 2^n$

$$\Rightarrow \{n(C_r)\}_{max} = 1 \ll \{(n(S) - 1) - k_r\}$$

For each $C_r \in S$, we maintain a Bit Map of size $\{n(C_r)\}_{max}$. Let the corresponding Bit Map be BM_r.

$$|BM_r| = \frac{\{n(C_r)\}_{max}}{S}$$ rounded up to closest multiple of S.

The ith bit of BM_r tells whether the

ith C_r is free or not.

Additionally for each C_r in $S - \{M'\}$, we maintain another BitMap sBM_r which says whether $(C_r)_i$ has been split or not.

⊙ Allocation

Let the allocation request size be x.

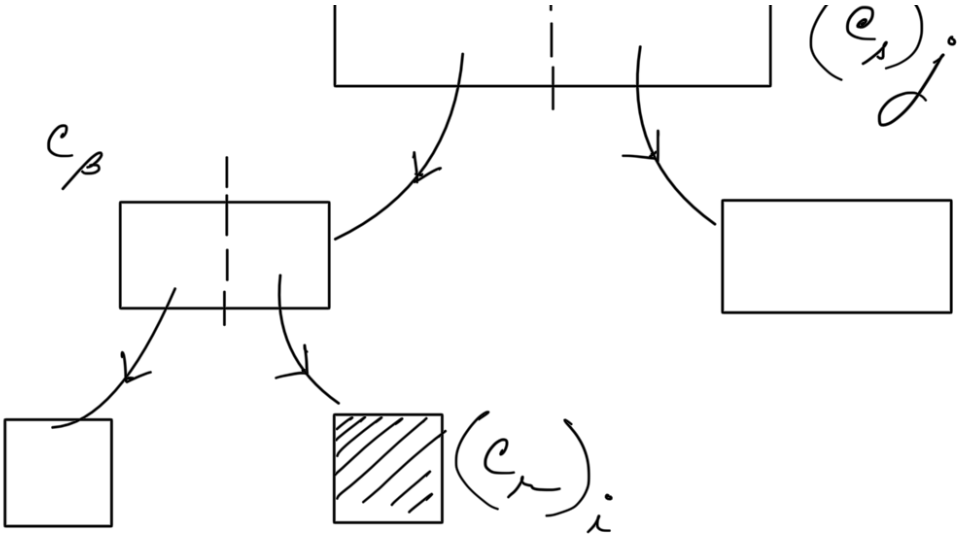
The suitable C_r will have

$$r = \begin{cases} 0 & \text{if } x \leq L \\ \text{ceil}(\log_2 x) & \text{otherwise} \end{cases}$$

We scan the BM_r to find a free $(C_r)_i$.

$$\text{Address}\left((C_r)_i\right) = M'_{start} + i\,|C_r|$$

If no $(C_r)_i$ is empty, we look for a free $(C_s)_j$ where $r < s \leq K_{max}$. Once found we start splitting it.

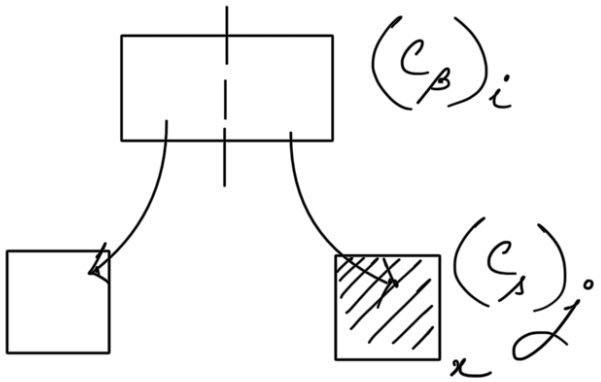


We can get the index of C_β in BM_β using

$$\frac{Address\left((C_r)_i\right) - M'_{start}}{|C_\beta|}$$

⊙ Deallocation

Let we want to deallocate at x.



$$Buddy\left((C_s)_j\right) = \begin{cases} (C_s)_{j-1} & \text{if } j \text{ is odd} \end{cases}$$

$$\left(C_s \right)_{j+1} \quad \text{otherwise}$$

If the buddy is free then we can coalesce.
We can keep coalescing further upwards
if possible.