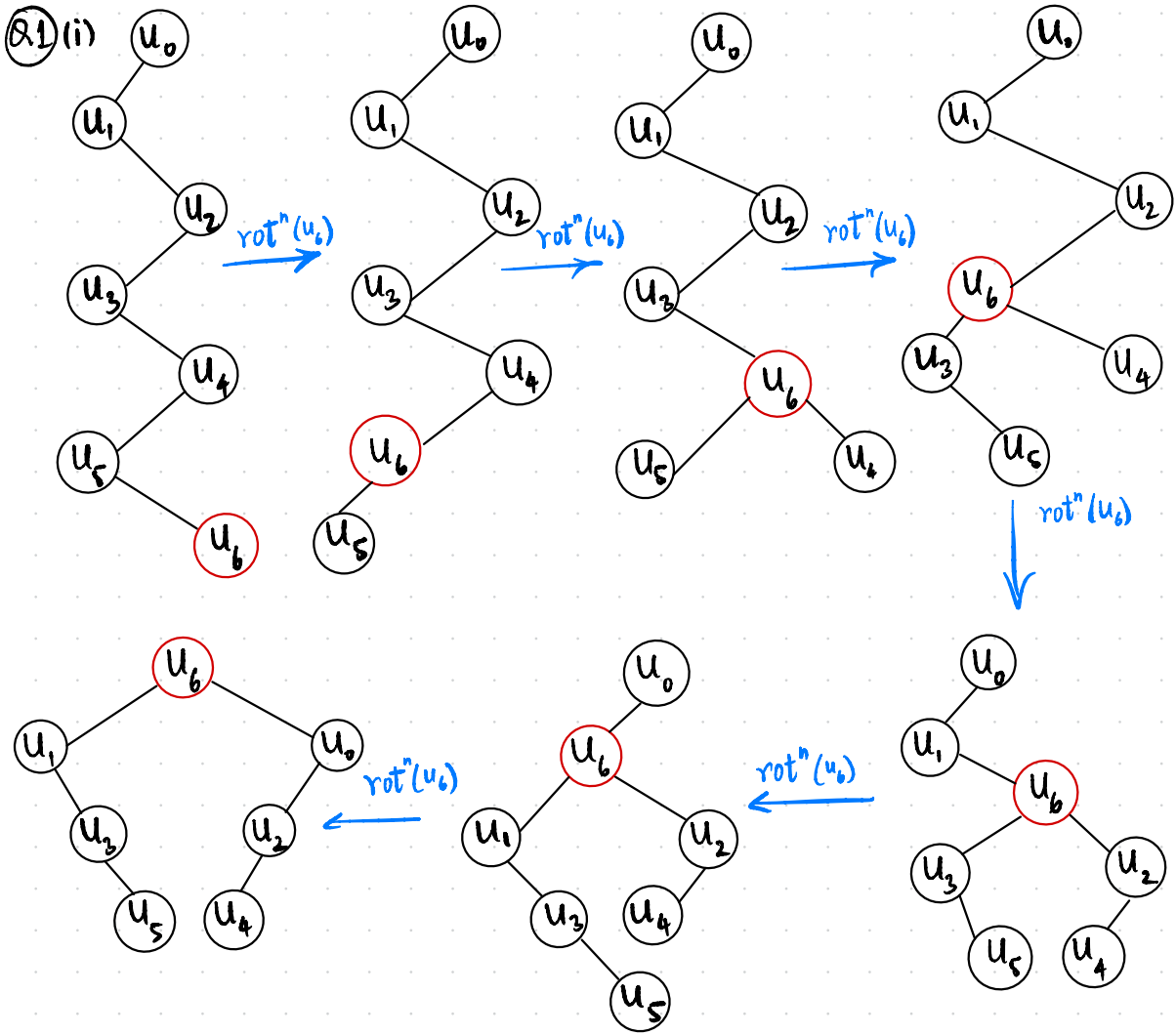
Assignment 3

ADITEYA BARAL (ab12057)

Q1 (i)



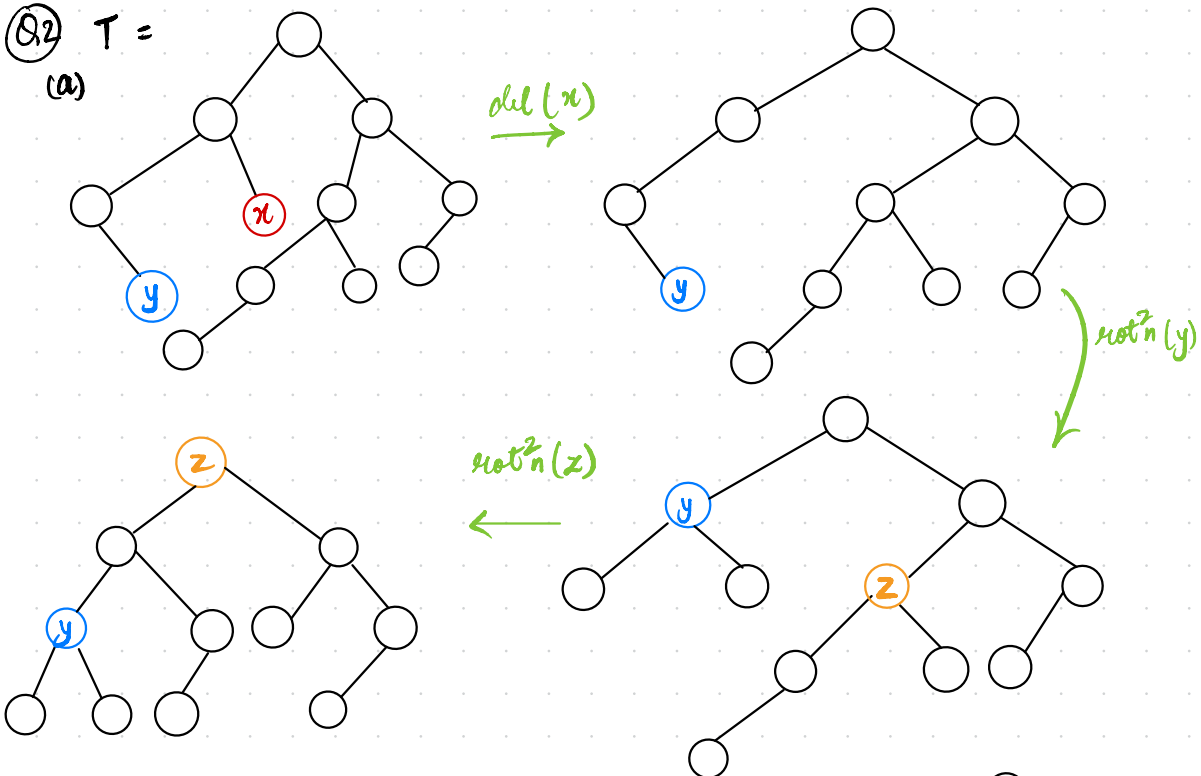
(ii) Left spine : $(u_n - u_1 - u_3 - \ldots)$
Right spine : $(u_n - u_0 - u_2 - \ldots)$

(iii) The height of the tree with n nodes reduces by 1 every 2 rotations.

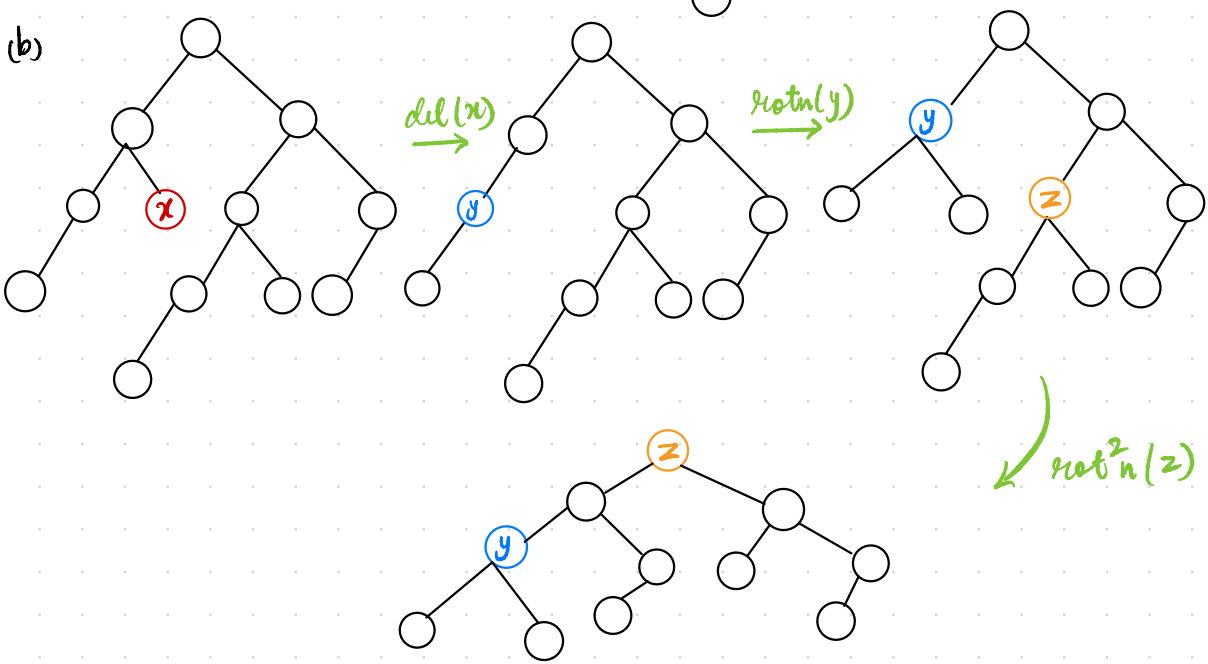
→ Final height of the tree $T_n = \left\lceil \frac{n-1}{2} \right\rceil$

Q2 T =

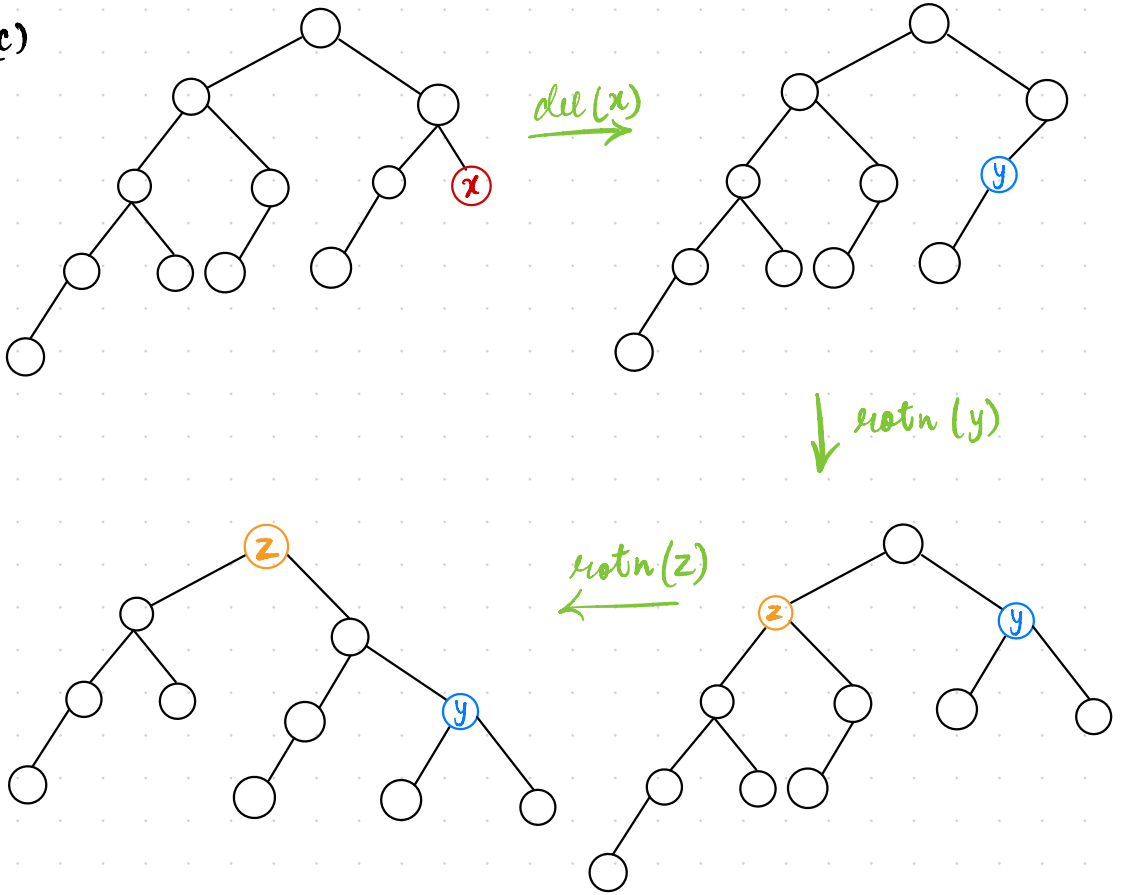
(a)



(b)



(c)



(Q3) (a)

$$T(n) = C + \sum_{i=1}^{k} a_i T(n-i)$$

$$t(n) = T(n) + c$$

$$\Rightarrow T(n) = t(n) - c$$

$$\Rightarrow t(n) - c = C + \sum_{i=1}^{k} a_i (t(n-i) - c)$$

$$t(n) = c + C + \sum_{i=1}^{k} a_i (t(n-i)) - c \sum_{i=1}^{k} a_i$$

$$= C + c\left(1 - \sum_{i=1}^{k} a_i\right) + \sum_{i=1}^{k} a_i (t(n-i))$$

We want $C + c\left(1 - \sum_{i=1}^{k} a_i\right) = 0$

$$\Rightarrow \boxed{c = \frac{C}{\sum_{i=1}^{k} a_i - 1}}$$

Limitation : This transformation is valid only if $\sum_{i=1}^{k} a_i - 1 \neq 0$

When $\sum_{i=1}^{k} a_i = 1$, the denominator becomes zero, and division by zero is invalid in this recurrence.

(b) $\mu(h) = 1 + \mu(h-1) + \mu(h-2)$

We know that this is of the form,

$T(n) = C + \sum_{i=1}^{k} T(n-i)$ where $C=1$, $a_1=1$, $a_2=1$, $\sum_{i=1}^{k} a_i = 2$

$$\Rightarrow c = \frac{C}{\sum_{i=1}^{k} a_i - 1} = \frac{1}{2-1} = 1$$

$$\Rightarrow \boxed{t(h) = \mu(h) + 1}$$

$$t(h) = t(h-1) + t(h-2)$$

$t(0) = \mu(0) + 1 = 2$
$t(1) = \mu(1) + 1 = 3$

Assume solution of $t(h)$ is of the form $t(h) = r^h$

$\Rightarrow r^h = r^{h-1} + r^{h-2}$
$r^2 = r + 1$ [Dividing by r^{h-2}]
$\Rightarrow r^2 - r + 1 = 0$

Roots of characteristic equation $r^2 - r + 1 = \phi, \hat{\phi}$
Since roots are distinct, the general solution to $t(h)$ is,

$\Rightarrow t(h) = A\phi^h + B\hat{\phi}^h$

$\Rightarrow 2 = A\phi^0 + B\hat{\phi}^0 \Rightarrow A + B = 2$

$3 = A\phi + B\hat{\phi}$

$\Rightarrow 3 = A\phi + (2-A)\hat{\phi} = A\phi + 2\hat{\phi} - A\hat{\phi} = A(\phi - \hat{\phi}) + 2\hat{\phi}$

$$\Rightarrow A = \frac{3 - 2\hat{\phi}}{\phi - \hat{\phi}} = 1.8944 \quad \left[\phi = \frac{1+\sqrt{5}}{2},\ \hat{\phi} = \frac{1-\sqrt{5}}{2}\right]$$

$$B = 2 - A = 0.1056$$

$$\Rightarrow t(h) = 1.8944\left(\frac{1+\sqrt{5}}{2}\right)^h + 0.1056\left(\frac{1-\sqrt{5}}{2}\right)^h$$

Since $t(h) = \mu(h) + 1$

$$\Rightarrow \mu(h) = t(h) - 1$$

$$\Rightarrow \boxed{\mu(h) = 1.8944\left(\frac{1+\sqrt{5}}{2}\right)^h + 0.1056\left(\frac{1-\sqrt{5}}{2}\right)^h - 1}$$

(Q4) Size = 100

(a) We know that : $\phi^{h+1} < \mu(h) + 1 \leq 2\phi^{h}$

Minimum number of nodes in an AVL tree of height $h = \mu(h)$

For $n = 100$, we want the largest h such that $\mu(h) \leq 100$

$\Rightarrow \mu(h) + 1 > \phi^{h+1}$

$\phi^{h+1} < 101$

$h+1 < \log_{\phi}(101)$

$\Rightarrow h+1 < 9.5699$ $\quad [\log_{\phi} 100 < \log_{\phi} 101$ and $\log_{\phi} 100 = 9.5699]$

$h < 8.5699$

$\Rightarrow$ $\boxed{h_{max} = 8}$ $\quad$ [We also know that $\mu(8) = 88$ and $\mu(9) = 143$
$\Rightarrow h_{max} = 8$.]

(b) Minimum height h_{min} is the smallest height such that the max number of nodes ≥ 100. We thus want a tree that is as dense as possible.

Maximum number of nodes in a binary tree of height $h = 2^{h+1} - 1$

$\Rightarrow 2^{h+1} - 1 \geq 100$

We can express $2 = \phi^{\log_{\phi} 2} = \phi^{1.4404}$

$\Rightarrow \phi^{1.4404(h+1)} \geq 101$

$1.4404(h+1) \geq \log_{\phi} 101$

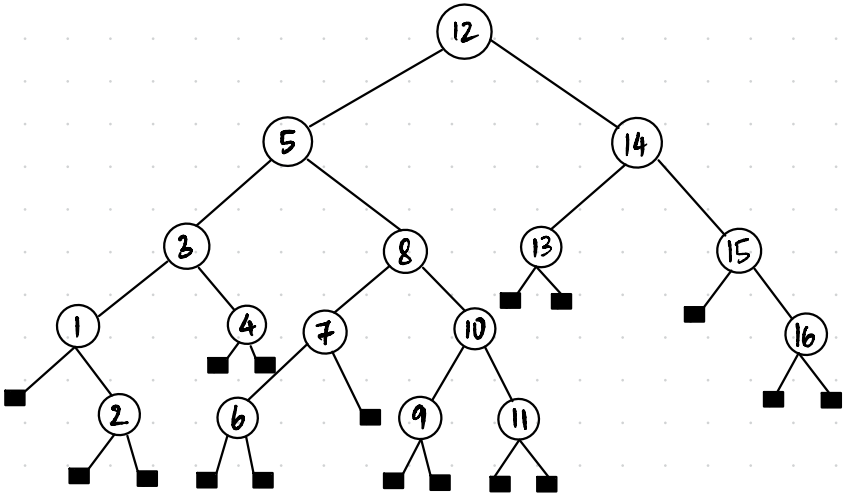
$\Rightarrow h+1 \geq \frac{9.5699}{1.4404}$

$h \geq 5.6439$

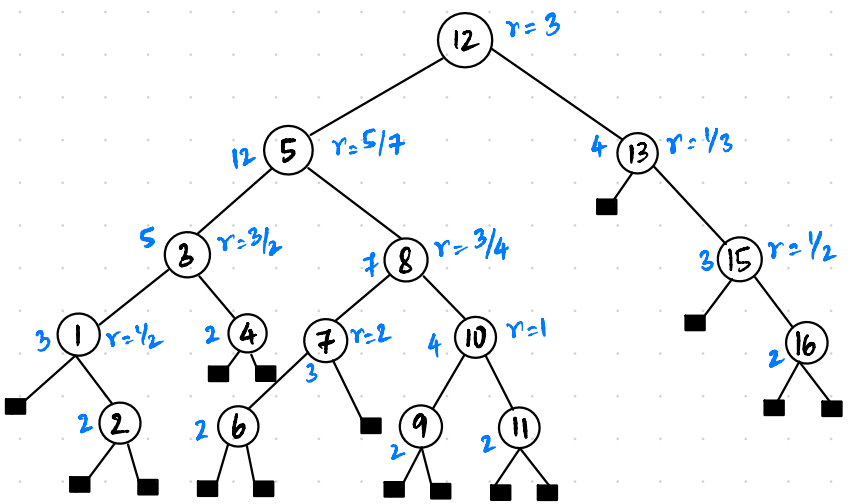
$\Rightarrow \boxed{h_{min} = 6}$

$\Rightarrow \boxed{\text{Range of heights} = [6,8] = \{6,7,8\}}$

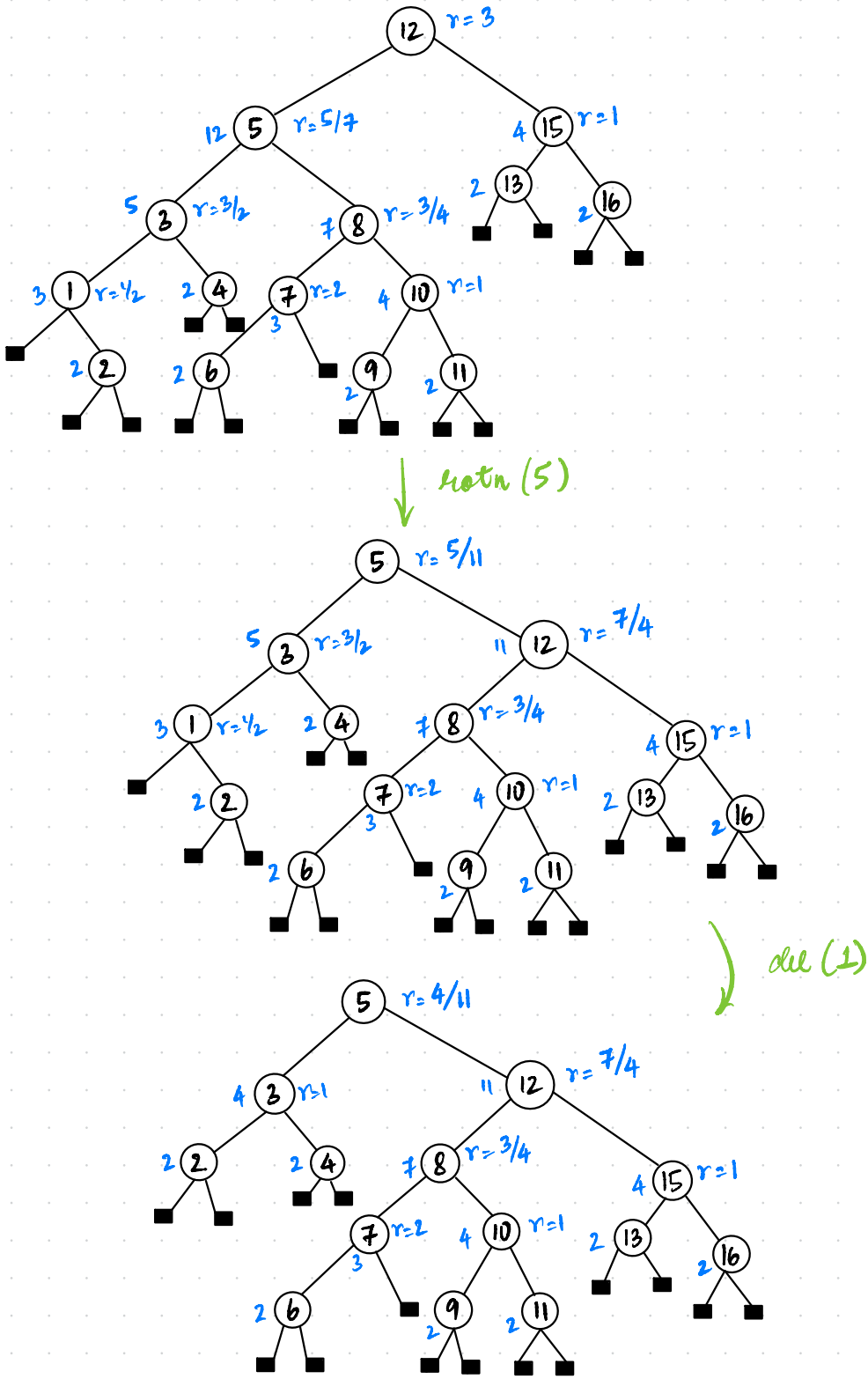
Q5

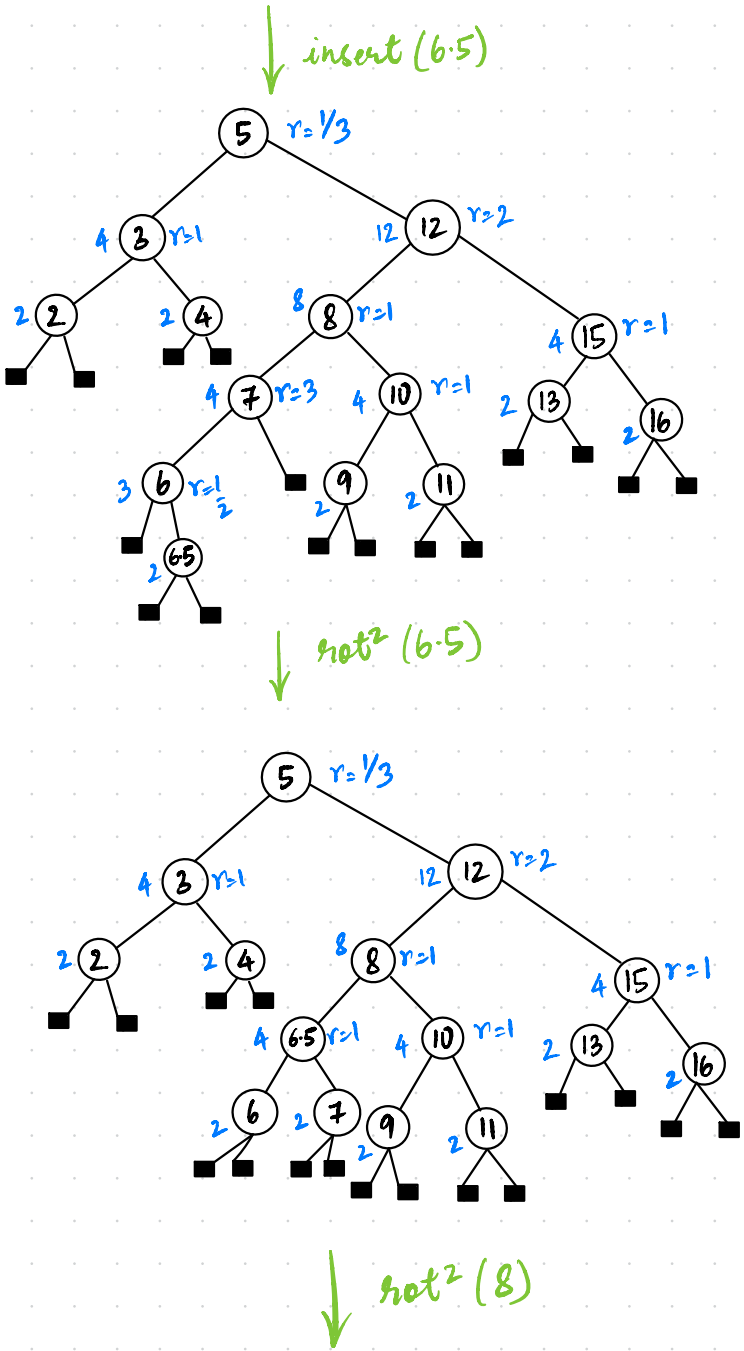


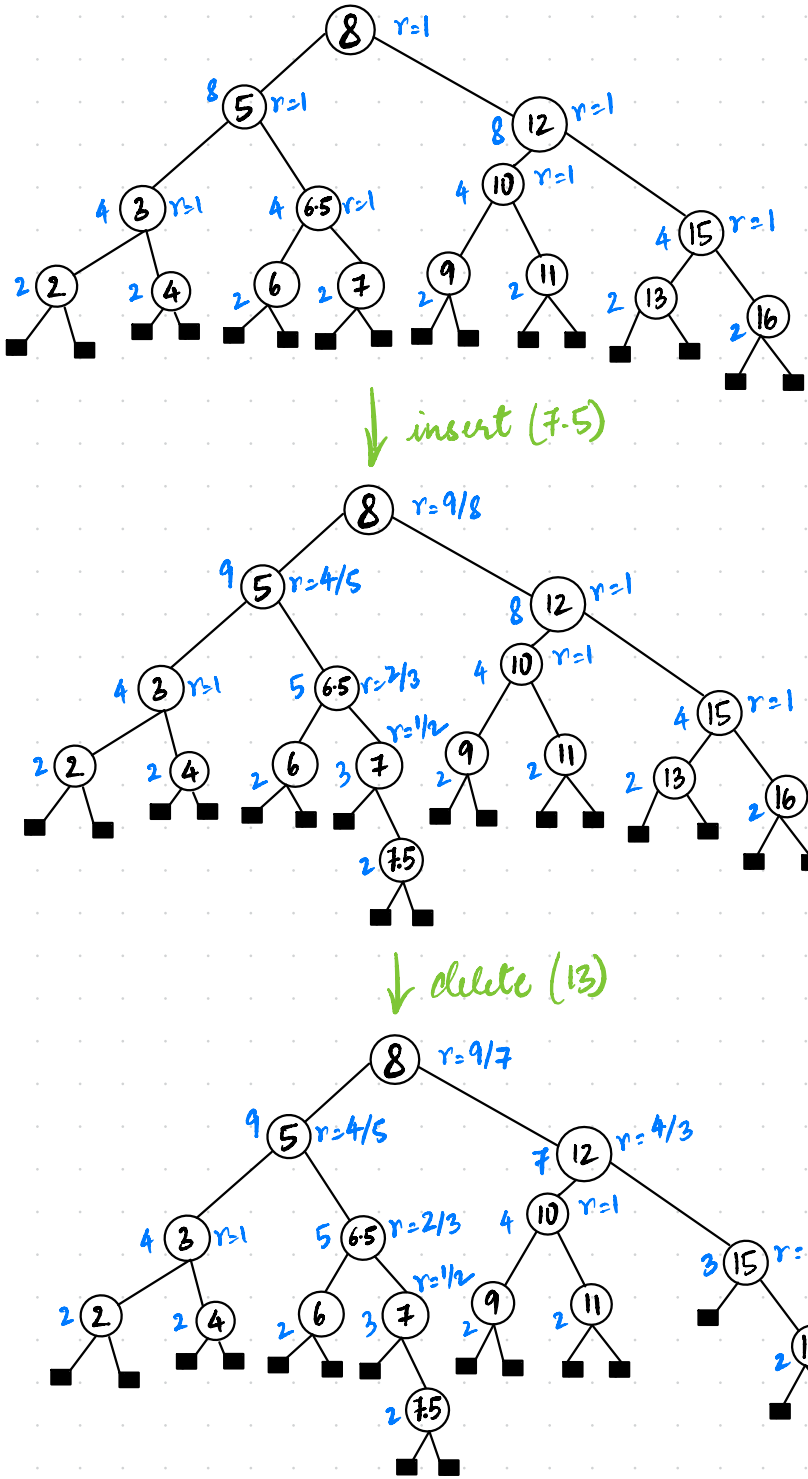
del (14)



rot (15)


r=3
r=5/7
r=1
r=3/2
r=3/4
r=1/2
r=2
r=1
rotn (5)
r=5/11
r=7/4
r=3/2
r=3/4
r=1/2
r=2
r=1
del (1)
r=4/11
r=7/4
r=1
r=3/4
r=2
r=1


insert (6.5)
5 r=1/3
4 3 r=1
12 12 r=2
2 2
2 4
8 8 r=1
4 15 r=1
4 7 r=3
4 10 r=1
2 13
2 16
3 6 r=1/2
2 9
2 11
2 6.5
rot² (6.5)
5 r=1/3
4 3 r=1
12 12 r=2
2 2
2 4
8 8 r=1
4 15 r=1
4 6.5 r=1
4 10 r=1
2 13
2 16
2 6
2 7
2 9
2 11
rot² (8)


r=1
r=1
r=1
r=1
r=1
r=1
r=1
insert (7.5)
r=9/8
r=4/5
r=1
r=1
r=2/3
r=1/2
r=1
r=1
delete (13)
r=9/7
r=4/5
r=4/3
r=1
r=2/3
r=1/2
r=1/2

(Q6) RB[β] $\Rightarrow$ β < ratio(u) < $1/\beta$ where u is the root node.

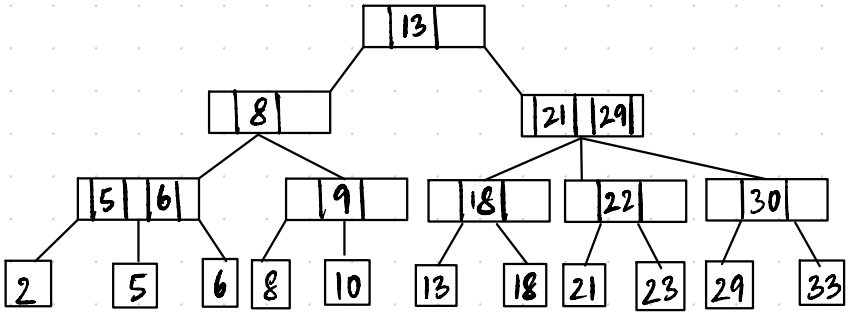
For $\beta < \frac{1}{2}$, we have $\frac{1}{2}$ < ratio(u) < 2. For small trees such as with 2 nodes, ratio(u) = 1/2 or 2 depending on which child branch is nil. If T(u) = 2, then ratio = 1/2 if u.left = nil or ratio = 2 if u.right = nil. If we allow $\beta \geq 1/2$ and T is β-ratio balanced, it implies that no subtree in T can be of size 2 and the tree T must be of odd size to prevent any subtrees with an even number of nodes. This is a severe restriction since it rules out various configurations of trees. This restriction might also degenerate the structure of the tree into something similar to a linked list since it would not maintain the logarithmic height of well-balanced trees. Thus to prevent this, we set $\beta \in (0, 1/2)$.

(a) As explained above, RB[2/3] would put a severe restriction for smaller trees. If $T_u = 2$, then the ratio is either 1/2 or 2. In both cases, ratio = 1/2 $\not>$ 2/3 and ratio = 2 $\not<$ 3/2. Thus, no trees of size = 2 can exist and all trees in RB[2/3] must have an odd total size since an even size will not meet the ratio constraint. Thus, any $\beta > 1/2$ would be severely restrictive since it would disallow any trees with even size and not allow subtrees with an even number of nodes.

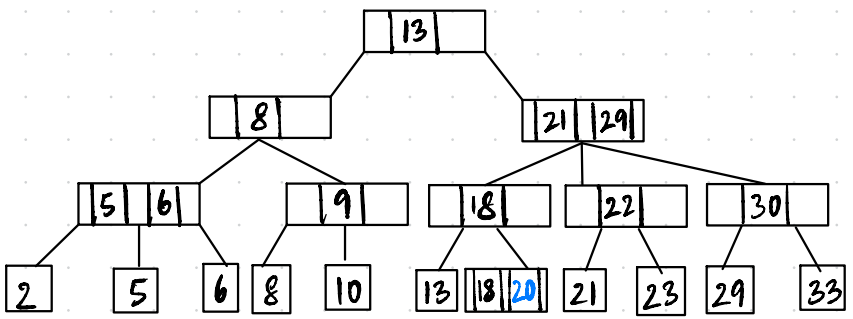
(b) The same problems extend to other values of $\beta \in [1/2, 1]$ when $\beta = (q-1)/q$ and $q = \{2, 3, 4, \ldots\}$. As q increases, $\beta \to 1$. This leads to the constraints becoming stricter. A tree of size = 2 will never satisfy the condition: β < ratio(u) < $1/\beta$ since the only valid ratio would be $\beta = 1/2$, which occurs when the tree has only a right child. As the constraints become stricter, more such trees may be disallowed since they do not fit the ratio constraint, and more subtree configurations may not fit the criteria. The stricter ratio constraint will limit the size of subtrees that the tree can have and make them impractical because only odd sized subtrees will be allowed.

(c) To allow $\beta \in [1/2, 1)$, we can relax the ratio constraints for smaller sized trees and permit dynamic ratio bounds. We can choose to not enforce the ratio bound for tree of sizes $\leq k$ (for example, $k=2$) or choose a looser bound for smaller trees (example, $\beta = (0, 1/2)$) and then gradually increase it to $[1/2, 1)$. We can also set a minimum valid subtree size and smaller trees can be allowed to maintain variations from the standard β ratio $(\alpha) < 1/\beta$ rule. We can also choose to define exceptions for certain subtrees or nodes to ensure the imbalance does not propagate to higher order nodes. The most feasible alternative would be to assume that trees with size $< N$ are always perfectly balanced.

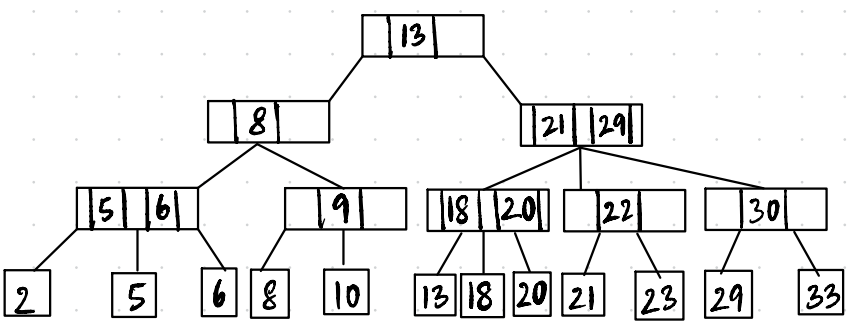
Q7



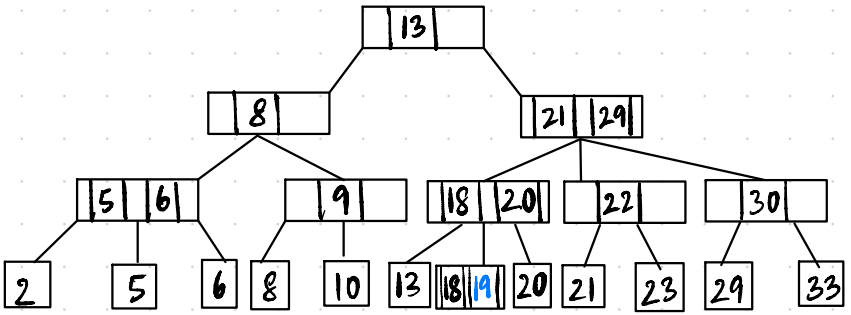
Insert (20)



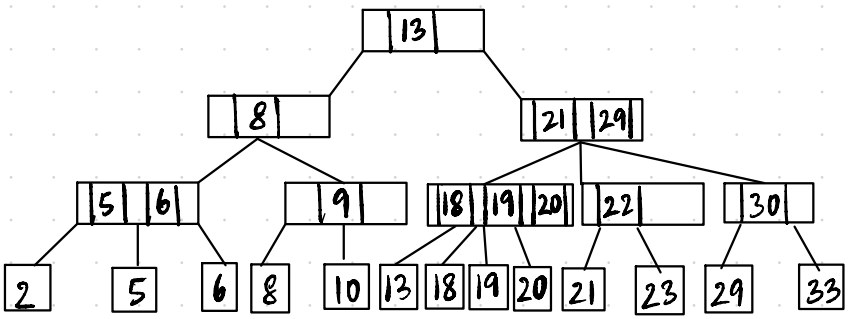
Split (18, 20)



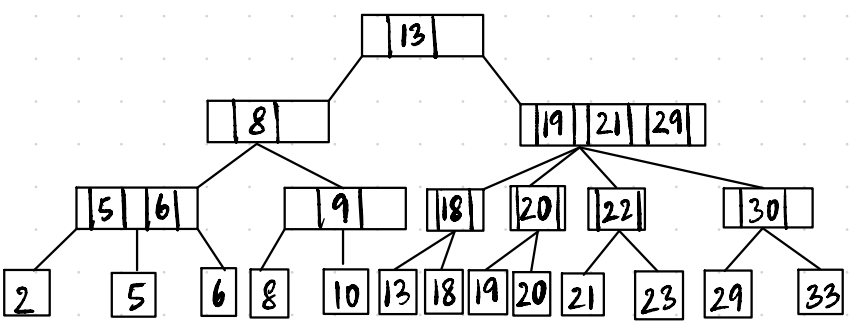
Insert (19)



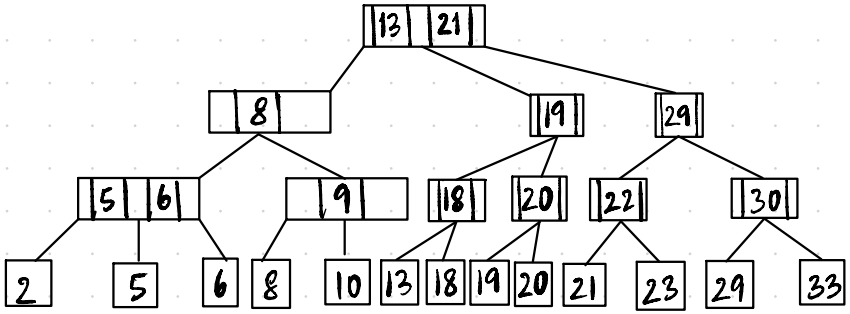
Split (18,19)



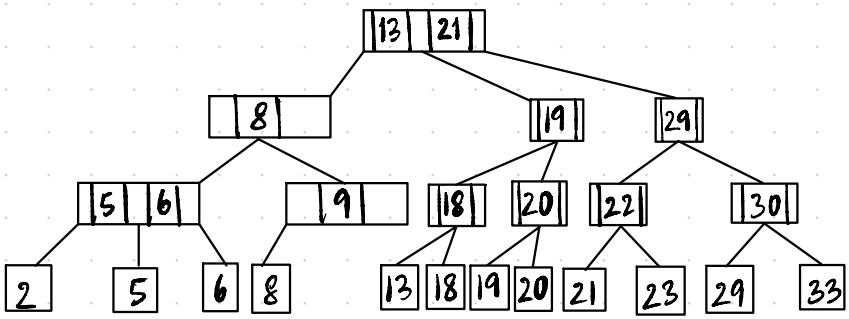
Split (18,19,20)



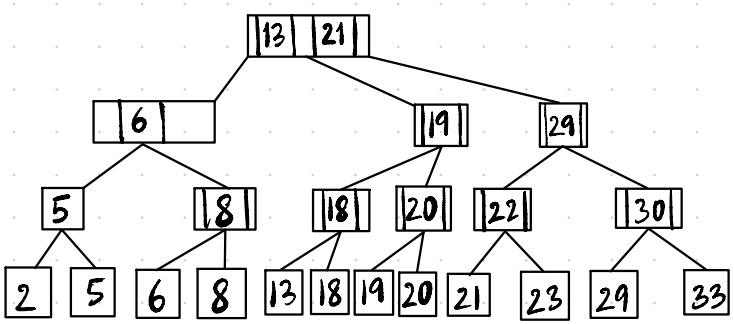
Split (19,21,29)



Add (10)



Borrow (6)



(02) (i) (a) $b-1$ keys $\times$ 6 bytes per key $= 6(b-1)$ bytes

b pointers $\times$ 4 bytes per pointer $= 4b$
Storing degree in the node $= 2$ bytes
Storing parent pointer in the node $= 4$ bytes

$\Rightarrow 6(b-1) + 4b + 2 + 4 \leq 4096$
$6b - 6 + 4b + 6 \leq 4096$
$10b \leq 4096$
$b \leq 409.6$

$\Rightarrow$ Maximum $b = 409$

(b) $n = 2 \times 10^6$
$a = \left\lfloor \frac{409+1}{2} \right\rfloor = 205$
Worst-case height $h = \left\lfloor \log_a \frac{an}{2} \right\rfloor = \lfloor 3.5 \rfloor = 3$

$\Rightarrow$ Levels requiring disk read $= 3$

IO cost $= 3 \times 1000 = 3000$ CPU cycles

Internal search cost per node $= 4 \log_2 b = 4 \log_2 409 \approx 35$ CPU cycles/node

$\Rightarrow$ Total cost $= 3000 + 3 \times 35$

$\Rightarrow$ Total cost $= 3105$ CPU cycles

(iv)(a) Overhead = parent pointer + degree + bookkeeping

$= 4 + 2 + 2 = 8$ bytes

Local pointers to $(b-1)$ children $= 4(b-1)$ bytes
Pointer to corresponding child in B-tree $= 4b$ bytes

$\Rightarrow 6(b-1) + 4(b-1) + 4b + 8 \leq 4096$
$14b - 2 \leq 4096$

$\Rightarrow b \leq 292.7$

$\Rightarrow \boxed{b = 292}$

(b) $a = \left\lfloor \frac{292+1}{2} \right\rfloor = 146$

$n = 2 \times 10^6$

We know the worst-case height $h = \left\lfloor \log_a \frac{an}{2} \right\rfloor = \lfloor 3.7 \rfloor = 3$

Levels requiring disk read = 3

$\Rightarrow$ Total IO cost $= 3 \times 1000 = 3000$ CPU cycles

Internal search cost $= 4 \log_2 (292) = 33$ CPU cycles

Total cost $= 3000 + 3 \times 33 = 3099$ CPU cycles

$\Rightarrow \boxed{\text{Total cost} = 3099 \text{ CPU cycles}}$