

CSE 5311 Homework #1 Solutions

Summer 1996

1 Binary Search Tree

Find the optimal binary search tree for the given keys and probabilities. The optimal subtree for all members of each family is given in Figure 1.

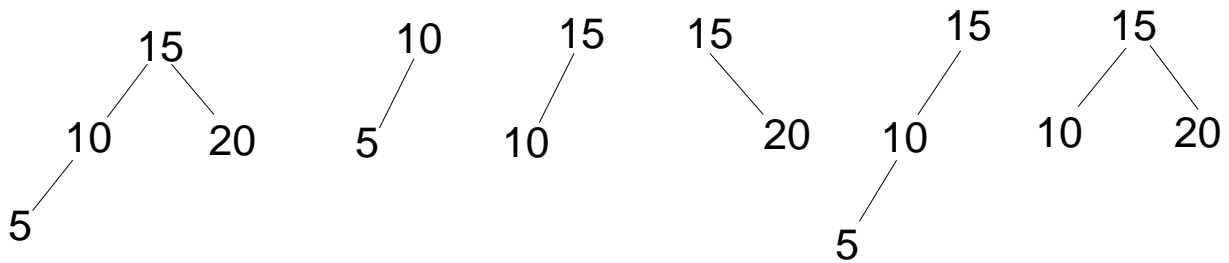


Figure 1: Optimal BST

2 Proof of Probe Length

Prove that the optimal static list for n elements with Zipf's distribution has an average probe length of $\frac{n}{H_n}$.

$$\sum_{i=1}^n p_i c_i = \sum_{i=1}^n i p_i = \sum_{i=1}^n i \frac{1}{i H_n} = \frac{1}{H_n} \sum_{i=1}^n 1 = \frac{n}{H_n}$$

3 Ordered Lists

Assuming that the probability of a key being accessed is independent of its key value, for what cases are ordered lists better than the optimal static list technique? (Binary search not allowed.)

The expected number of probes for the optimal static list where the keys are ordered by decreasing probability is

$$P_1 = \sum_{i=1}^n iP_i + n \sum_{i=0}^n V_i$$

where V_i is the probability of a miss in between key i and key $i + 1$. In this case, a miss requires looking at all n elements.

The expected number of probes for the sorted list (sorted by key value) is

$$P_2 = \sum_{i=1}^n iP_{\text{sorted list}_i} + n \sum_{i=0}^n V_{\text{sorted list}_i}$$

So an ordered list (ordered by key value) is better then $P_2 < P_1$.

4 Circular Lists

Suppose a circular list has 5 elements in order by decreasing Zipf probability and wrapping around. Each search starts with the last element found. What is the expected number of probes required to find an element?

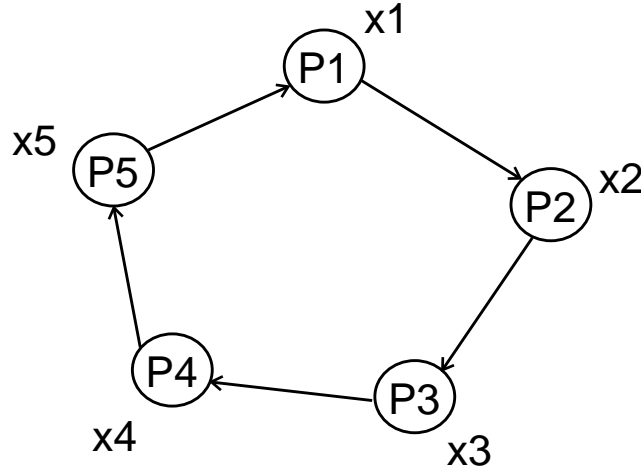


Figure 2: Circular List

Based on Figure 2 the expected number of probes is

$$\begin{aligned} &P_1(P_1 * 0 + p_2 * 1 + P_3 * 2 + P_4 * 3 + p_5 * 4) + \\ &P_2(P_2 * 0 + p_3 * 1 + P_4 * 2 + P_5 * 3 + p_1 * 4) + \\ &P_3(P_3 * 0 + p_4 * 1 + P_5 * 2 + P_1 * 3 + p_2 * 4) + \\ &P_4(P_4 * 0 + p_5 * 1 + P_1 * 2 + P_2 * 3 + p_3 * 4) + \\ &P_5(P_5 * 0 + p_1 * 1 + P_2 * 2 + P_3 * 3 + p_4 * 4) \end{aligned}$$

where $P_i = \frac{1}{iH_n}$.

5 Optimal Binary Search Tree with 5 Keys

See Figure 3

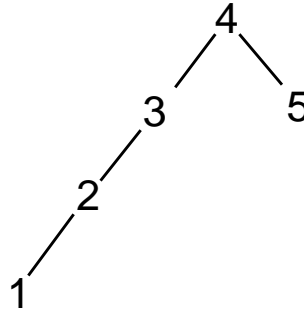


Figure 3: Optimal BST with 5 Keys

6 Cost Function

Not Done.

7 Upper Bound by Substitution

Find an asymptotic upper bound on the recurrence $T(n) = T(\frac{n}{5}) + n$ by using substitution.

Guess: $T(n) = O(n)$. So we need to prove that $T(n) \leq cn$ for some $c > 0$. Assume this hold for $T(\frac{n}{5})$ (ie. $T(\frac{n}{5}) \leq c\frac{n}{5}$). Substituting into the recurrence yields

$$\begin{aligned} T(n) &\leq c\frac{n}{5} + n \\ &= \left(\frac{c}{5} + 1\right)n \\ &= \left(c - \frac{4c}{5} + 1\right)n \\ &\leq cn \text{ if } c \geq \frac{5}{4} \end{aligned}$$

Therefore we have $T(n) = O(n)$.

8 Upper Bound by Iteration

Find an asymptotic upper bound on the recurrence $T(n) = 3T(\frac{n}{4}) + \frac{1}{2}n$ by using iteration.

$$\begin{aligned} T(n) &= \frac{n}{2} + 3T\left(\frac{n}{4}\right) \\ &= \frac{n}{2} + 3\left[\frac{n}{8} + 3T\left(\frac{n}{16}\right)\right] \\ &= \frac{n}{2} + 3\left[\frac{n}{8} + 3\left[\frac{n}{32} + 3T\left(\frac{n}{64}\right)\right]\right] \\ &= \frac{n}{2} + \frac{3n}{8} + \frac{9n}{32} + 27T\left(\frac{n}{64}\right) \end{aligned}$$

The i th term in the series is $\frac{3^i}{2} \frac{n}{4^i}$. The iteration hits $n = 1$ when $\frac{n}{4^i} = 1$ or, equivalently, when i exceeds $\log_4 n$. By continuing the iteration until this point it can be seen that the summation contains a decreasing geometric series:

$$\begin{aligned} T(n) &\leq \frac{n}{2} + \frac{3n}{8} + \frac{9n}{32} + \dots + \frac{1}{2} 3^{\log_4 n} \\ &\leq \frac{n}{2} \sum_{i=0}^{\infty} \left(\frac{3}{4}\right)^i + \Theta(n^{\log_4 3}) \text{ since } 3^{\log_4 n} = n^{\log_4 3} \\ &= \frac{n}{2} 4 + O(n) \text{ since } \log_4 3 < 1, \text{ hence } \Theta(n^{\log_4 3}) = O(n) \\ &= 2n + O(n) \\ &= O(n) \end{aligned}$$

9 Upper and Lower Bound by Substitution

Find an asymptotic upper and lower bound on the recurrence $T(n) = 3T(\frac{n}{3}) + 1$ by using substitution.

1. Show that $T(n)$ is in $\Omega(n)$ (ie. $T(n) \geq cn$ for some $c > 0$).

Assume $T(\frac{n}{3}) \geq c\frac{n}{3}$. Substituting into the recurrence yields

$$\begin{aligned} T(n) &\geq 3c\frac{n}{3} + 1 \\ &= cn + 1 \\ &\geq cn \end{aligned}$$

Thus $T(n)$ is in $\Omega(n)$.

2. Show that $T(n)$ is in $O(n)$ (ie. $T(n) \leq cn$ for some $c > 0$).

Assume $T(\frac{n}{3}) \leq c\frac{n}{3}$. Substituting into the recurrence yields

$$\begin{aligned} T(n) &\leq 3c\frac{n}{3} + 1 \\ &= cn + 1 \end{aligned}$$

which does not imply $T(n) \leq cn$ for any choice of c . This difficulty can be overcome by subtracting a lower-order term from our previous guess. Our new guess is $T(n) \leq cn - b$ where $b \geq 0$ is a constant. We now assume $T(\frac{n}{3}) \leq c\frac{n}{3} - b$. Substituting into the recurrence yields

$$\begin{aligned} T(n) &\leq 3(c\frac{n}{3} - b) + 1 \\ &= cn - 3b + 1 \\ &\leq cn - b \text{ as long as } b \geq \frac{1}{2} \end{aligned}$$

Thus $T(n)$ is in $O(n)$.

Combining these two parts gives $T(n)$ is in $\Theta(n)$.

10 Upper Bound by Iteration Again

Use the iteration method to show that $f(n) = n$ is an asymptotic upper bound on $T(n) = 3T(\frac{n}{3}) + 1$.

$$\begin{aligned} T(n) &= 1 + 3T\left(\frac{n}{3}\right) \\ &= 1 + 3 \left[1 + 3T\left(\frac{n}{9}\right) \right] \\ &= 1 + 3 \left[1 + 3 \left[1 + 3T\left(\frac{n}{27}\right) \right] \right] \\ &= 1 + 3 + 9 + 27T\left(\frac{n}{27}\right) \end{aligned}$$

The i th term is 3^i . The iteration stops when $\frac{n}{3^i} = 1$ or $i = \log_3 n$ (assuming $T(1) = \Theta(1)$). Thus, we have

$$T(n) = \underbrace{1 + 3 + 9 + \cdots}_{\log_3 n \text{ terms}} = \sum_{i=0}^{\log_3 n} 3^i = \frac{\log_3 n + 1}{3 - 1} = \frac{3n - 1}{2} = O(n)$$

11 Open-Address Hash Table

Determine the expected number of probes in a successful search of an open-address hash table which has the properties given in the problem.

When the table is full we have $m = n$. Thus the expected number of probes is

$$\begin{aligned} &= \sum_{i=0}^{n-1} \frac{m}{m-i} \frac{2(n-i)}{n(n+1)} \\ &= \sum_{i=0}^{m-1} \frac{m}{m-i} \frac{2(m-i)}{m(m+1)} \\ &= \frac{2}{m+1} \sum_{i=0}^{m-1} 1 \\ &= \frac{2m}{m+1} \\ &\leq 2 \end{aligned}$$

12 Red-Black Tree Insertion

Figure 4 shows the modifications to the tree resulting from the addition of 160. Nodes which have modified R/B values are depicted by a dashed outline. An empty square represents an unchanged subtree.

13 Red-Black Tree Insertion Again

Figure 5 shows the modifications to the tree resulting from the addition of 95. Nodes which have modified R/B values are depicted by a dashed outline. An empty square represents an unchanged subtree.

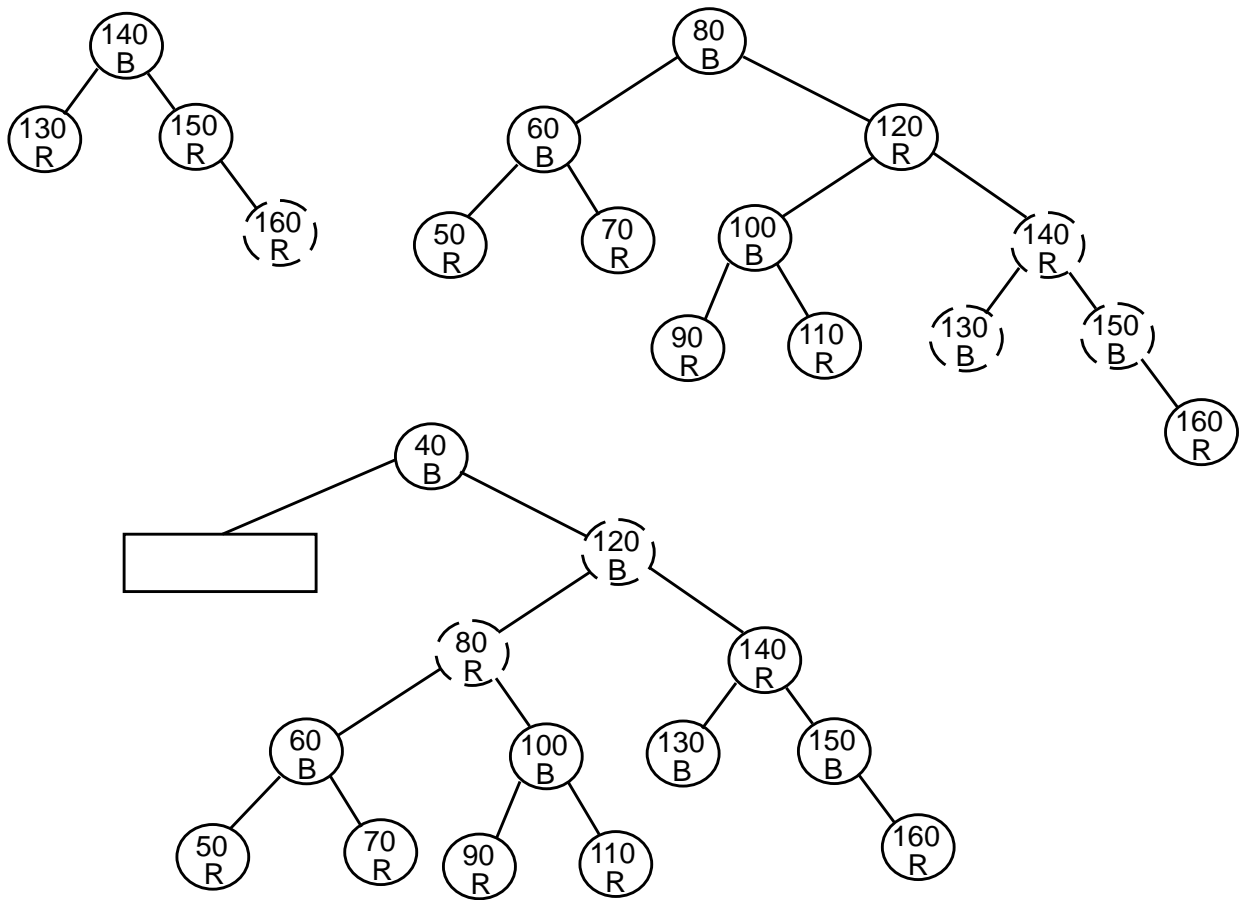


Figure 4: Insertion of 160 into Red-Black Tree

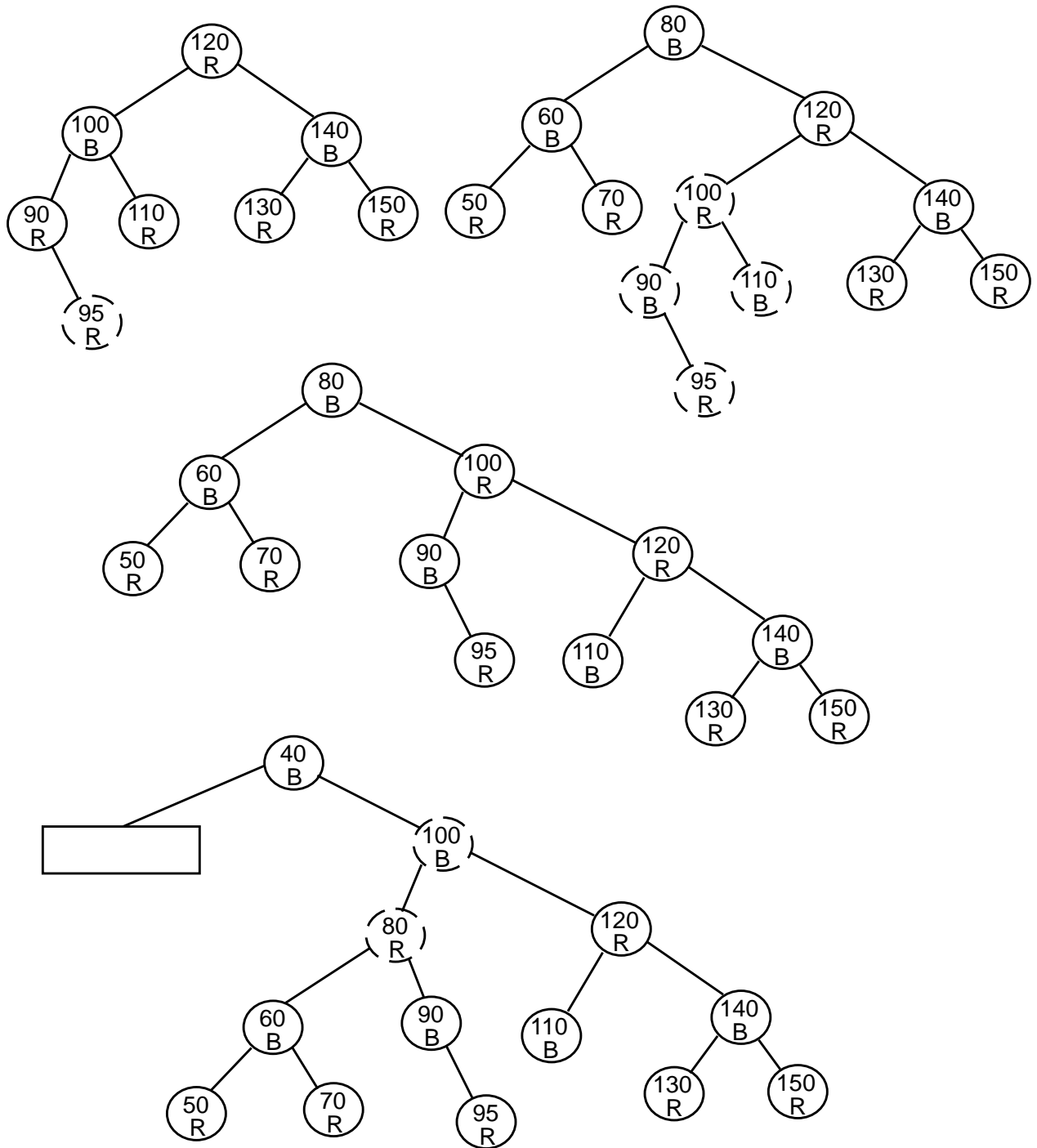


Figure 5: Insertion of 95 into Red-Black Tree

14 Red-Black Tree Deletion

Figure 6 shows the modifications to the tree resulting from the deletion of 80. Nodes which have modified R/B values are depicted by a dashed outline. An empty square represents an unchanged subtree.

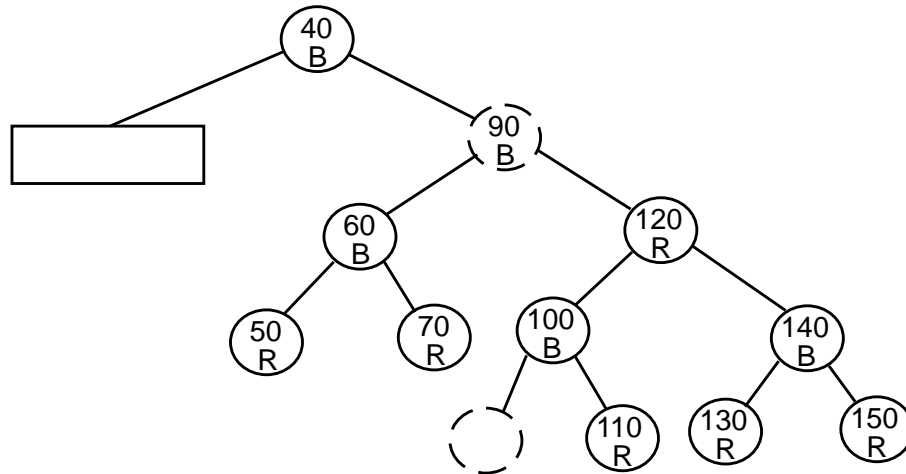


Figure 6: Deletion of 80 from Red-Black Tree

15 Red-Black Tree Deletion Again

Figure 7 shows the modifications to the tree resulting from the deletion of 30. Nodes which have modified R/B values are depicted by a dashed outline. An empty square represents an unchanged subtree.

16 AVL Tree Insertion and Deletion

Figures 8 and 9 show various stages of the insertions. Figure 10 shows the results of the specified deletions.

17 AVL Tree Deletion

Figure 11 shows the results of the specified deletion. Empty boxes are used to denote unchanged subtrees.

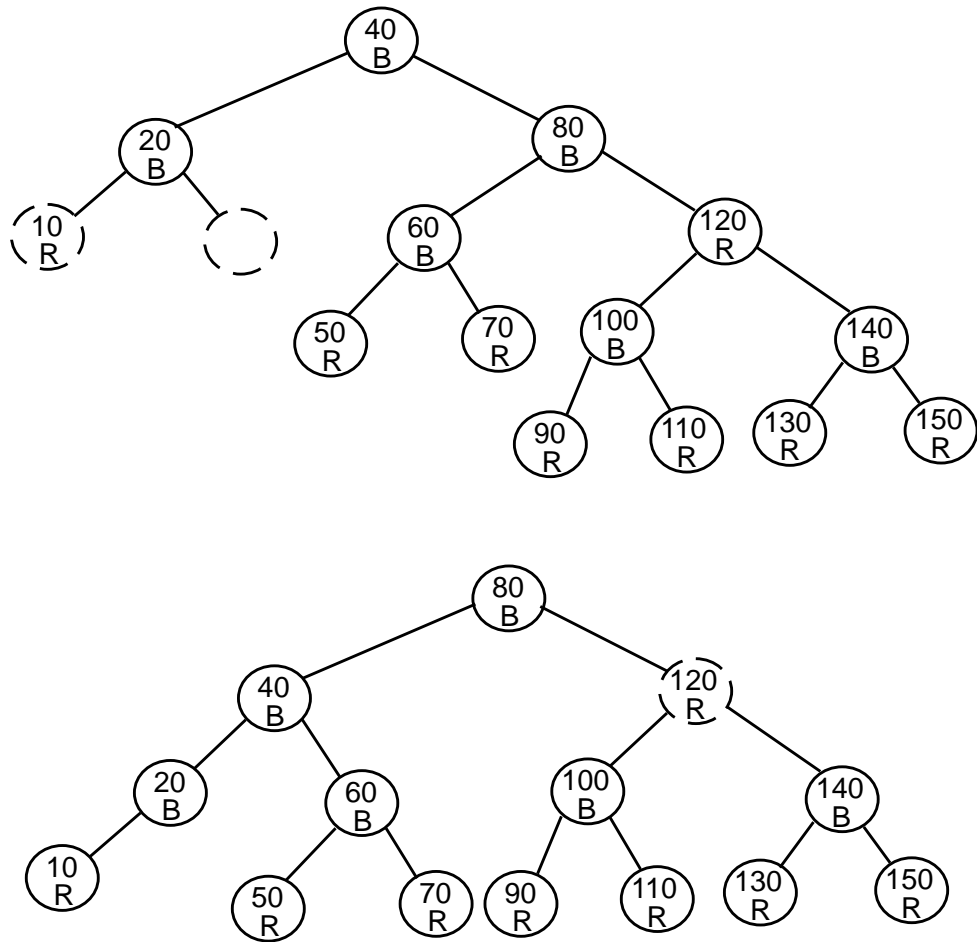


Figure 7: Deletion of 30 from Red-Black Tree

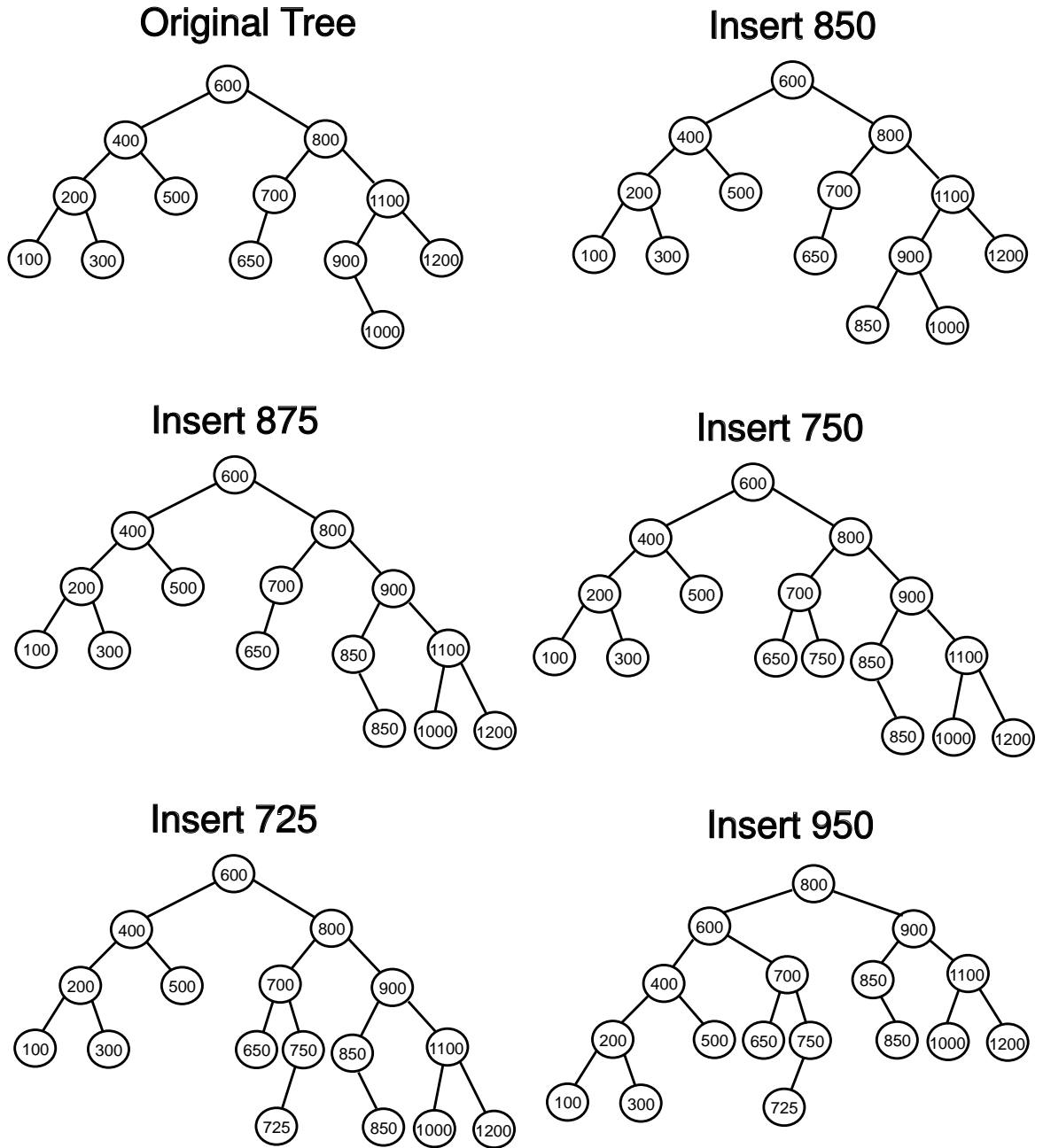


Figure 8: AVL Insertion of 850 thru 950

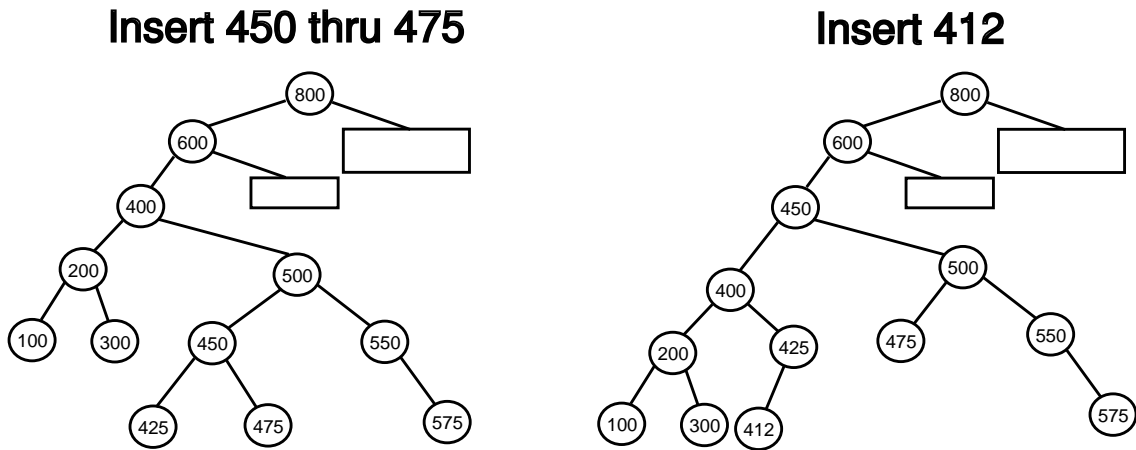


Figure 9: AVL Insertion of 450 thru 412

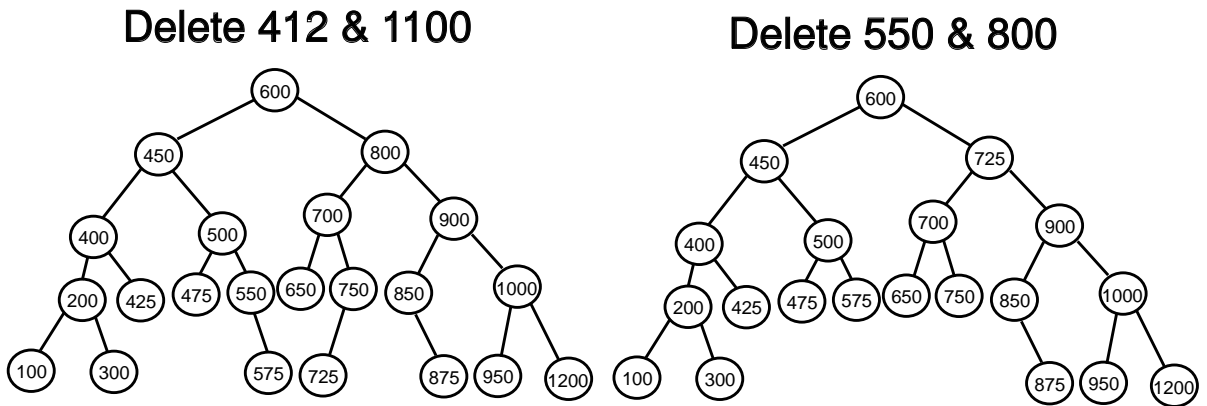


Figure 10: AVL Deletion of 412 thru 800

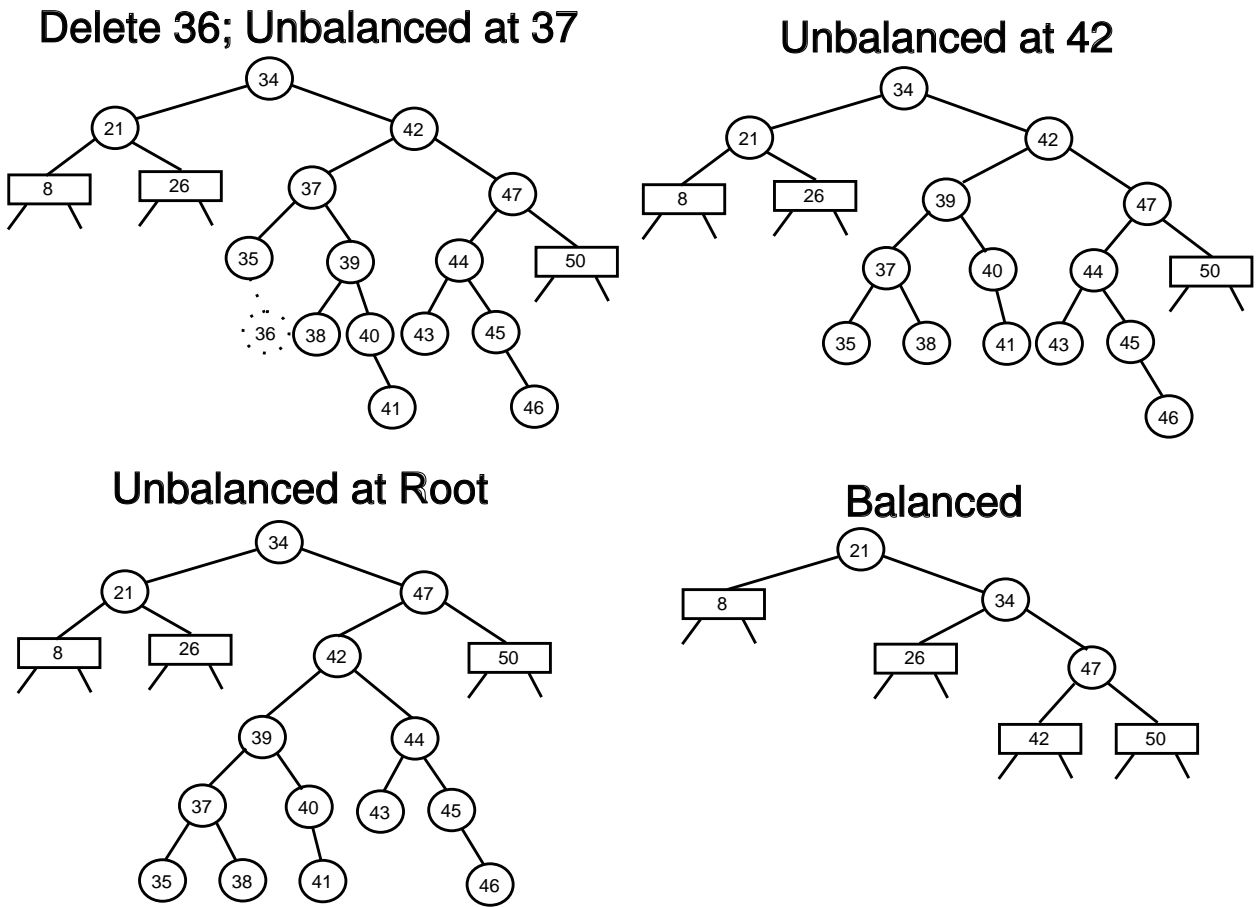


Figure 11: Deletion of 36 from AVL Tree

18 Optimal BST Construction

The answers are as follows:

- building $c(0, 5)$ using roots 2 thru 3
- Average probe length is 2.1500
- $C(0, 5)$ cost 2.1500 $20(10, 30(, 50(40,)))$
- Optimal binary search tree is given in Figure 12.

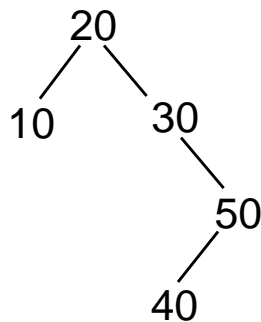


Figure 12: Optimal BST

- cost is 2.15

19 Problem 4.3-1

a)

$$\begin{aligned}
 T(n) &= 4T\left(\frac{n}{2}\right) + n \\
 a &= 4, b = 2, f(n) = n \\
 n^{\log_b a} &= n^{\log_2 4} = n^2 \\
 f(n) &= n = O(n^{\log_b a - \epsilon}), \epsilon(n^2) \text{ case 1} \\
 \text{Thus } T(n) &= \Theta(n^{\log_2 4}) = \Theta(n^2)
 \end{aligned}$$

b)

$$\begin{aligned}
 T(n) &= 4T\left(\frac{n}{2}\right) + n^2 \\
 a &= 4, b = 2, f(n) = n^2 \\
 f(n) &= n^2 = \Theta(n^{\log_2 4}) \text{ case 2} \\
 \text{Thus } T(n) &= \Theta(n^2 \lg n)
 \end{aligned}$$

c)

$$\begin{aligned}
T(n) &= 4T\left(\frac{n}{2}\right) + n^3 \\
a &= 4, b = 2, f(n) = n^3 \\
f(n) &= \Omega(n^{\log_2 4 + \epsilon}), \epsilon = 1 > 0 \text{ case 3} \\
af\left(\frac{n}{b}\right) &\leq cf(n) \text{ for some } c < 1 \\
\text{or } 4\frac{n^3}{8} &\leq cn^3 \\
\text{or } \frac{1}{2} &\leq c \text{ which is true for } \frac{1}{2} \leq c < 1 \\
\text{Thus } T(n) &= \Theta(n^3)
\end{aligned}$$

20 Problem 4.3-2

$$\begin{aligned}
T(n) &= 7T\left(\frac{n}{2}\right) + n^2 \\
a &= 7, b = 2, f(n) = n^2 \\
n^{\log_b a} &= n^{\log_2 7} \sim n^{2.8} \\
f(n) &= n^2 = O(n^{\log_2 7 - 0.8}), \epsilon = 0.8 > 0 \text{ case 1} \\
\text{Thus } T(n) &= \Theta(n^{\log_2 7})
\end{aligned}$$

$$\begin{aligned}
T'(n) &= aT'\left(\frac{n}{4}\right) + n^2 \\
b &= 4, f(n) = n^2 \\
n^{\log_b a} &= n^{\log_4 a} \\
n^{\log_4 a} &= n^{\log_2 7} \text{ or } n^{\log_4 a} = n^{\log_2 7} \\
\text{Thus } a &= 4^{\log_2 7} \approx 48.50293
\end{aligned}$$

So, the answer is $a = 48$.

21 Problem 6-1

a) For every ball we have a choice of putting it in one of b bins.

$$\text{Thus } \overbrace{b \times b \times \cdots \times b}^n = b^n$$

b) Using the hint, $b - 1$ sticks and n balls can be arranged in $(n + b - 1)!$ ways. Since the sticks are identical the unique number of ways are:

$$\frac{(n + b - 1)!}{(b - 1)!}$$

c) Since the n balls are identical, $n!$ permutations must be removed which gives:

$$\frac{(n+b-1)!}{(b-1)!} \frac{1}{n!} = \binom{n+b-1}{n}$$

d) Here the assumption is that the number of balls are less than the number of bins.

This gives us $\binom{b}{n}$ as the number of ways to place the n balls.

e) Since the bins are not to be empty and the balls are identical, the problem shifts to placing $(n-b)$ identical balls into b bins in x ways. Using part c) we get:

$$x = \binom{b+(n-b)-1}{b-1} = \binom{n-1}{b-1}$$

22 Problem 6-2

a) $\frac{1}{n}$

b) $A[j] \leq A[i]$

c) $\frac{1}{i}$ (which is the probability of it being the MAX of the list seen until that iteration)

d) $E[S_i] = 1 \frac{1}{i} = \frac{1}{i}$

e)

$$\sum_{i=1}^n E[S_i] = \sum_{i=1}^n \frac{1}{i} = \ln n + O(1) = E$$

$$E = \ln n + O(1) \text{ using this result}$$

$$\text{Thus } E = \Theta(\lg n)$$

23 Problem 10.3-1

This solution is taken nearly verbatim from the printed solution.

- For groups of 7, the algorithm still works in linear time. The number of elements greater than x (and less than x) is at least $4(\lceil \frac{1}{2} \lceil \frac{n}{7} \rceil \rceil - 2) \geq \frac{2n}{7} - 8$, and the recurrence becomes

$$T(n) \leq T(\lceil n/7 \rceil) + T(5n/7 + 8) + O(n)$$

which can be shown to be $O(n)$ by substitution.

	linear probing		quadratic probing		double hashing		Brent's rehash
0	22	0	22	0	22	0	22
1	88	1	1	1	1	1	1
2	1	2	88	2	59	2	59
3	1	3	17	3	17	3	17
4	4	4	4	4	4	4	4
5	15	5	1	5	15	5	15
6	28	6	28	6	28	6	28
7	17	7	59	7	88	7	88
8	59	8	15	8	1	8	1
9	31	9	31	9	31	9	31
10	10	10	10	10	10	10	10

Table 1: Solutions to Problem 12.4-1

- For groups of 3, however, the algorithm no longer works in linear time. The recurrence becomes

$$T(n) \leq T(\lceil n/3 \rceil) + T(2n/3 + 4) + O(n)$$

which does not have a linear solution. This can be proved by showing that the worst-case time for groups of 3 is $\Omega(n \lg n)$, which can be done by deriving a recurrence for a particular case that takes $\Omega(n \lg n)$ time.

In counting up the number of elements greater than x , consider the particular case in which there are exactly $\lceil \frac{1}{2} \lceil \frac{n}{3} \rceil \rceil$ groups with medians $\geq x$ and in which the “leftover” group contributes 2 elements greater than x . Then the number of elements greater than x is exactly $2(\lceil \frac{1}{2} \lceil \frac{n}{3} \rceil \rceil - 1) + 1$ (the -1 discounts x 's group and the $+1$ is contributed by x 's group) $= 2\lceil n/6 \rceil - 1$, and the recursive step for elements $\leq x$ has $n - (2\lceil n/6 \rceil - 1) \geq \frac{2n}{3} - 1$ elements. This give the recurrence

$$T(n) \geq T(\lceil n/3 \rceil) + T(2n/3 - 1) + \Theta(n) \geq T(n/3) + T(2n/3 - 1) + \Theta(n)$$

from which you can show that $T(n) \geq cn \lg n$ by substitution. You can also see that $T(n)$ is non-linear by noticing that each level of the recursion tree sums to n .

- (In fact, any odd group size ≥ 5 work in linear time.)

24 Problem 12.4-1

Table 1 gives the results of each method. Note: In this case, because of the order of the inserts, Brent's rehash ends up with the same tables as double hashing. If you insert key 31 after key 15, you will notice that the hash table remains unchanged for double hashing but changes quite a bit for Brent's rehash.

25 Problem 15.2-2

Yes. Since the black-height of a node is the number of black nodes in one of its subtrees (left of right—they have the same number of black nodes), this problem is very similar to the one given in the text on maintaining as a field the size of a subtree rooted at a node (page 282–285). The black-heights can be maintained as fields for both insertion and deletion without affecting the asymptotic running times of either operation because the updates will only be local.

26 Problem 15.2-3

No. When the depth of a node is maintained as a field, a rotation requires changing the depth information of all the nodes in the subtrees involved. For example, a rotation at the root will require the depth field of all n nodes in the RB-tree to be updated which cannot be done without affecting the asymptotic ($O(\log n)$) performance of the RB-tree operations.

27 Binomial Queue Question

Why does insertion into a binomial queue take $O(1)$ amortized time?

It can be observed that half of the insertions (every other insertion) require no merging at all, and for the other half of the inserts there are a very limited number of operations. Actually, this problem is similar to that of incrementing a binary counter as given in the text book (pages 358–360) and the sequence of operations and the subsequent analysis is the same as the for the binary counter. Thus, insertion into a binomial queue takes $O(1)$ amortized time.

28 Binomial Queue Merge

Figure 13 shows the steps involved in merging the two binomial queues.

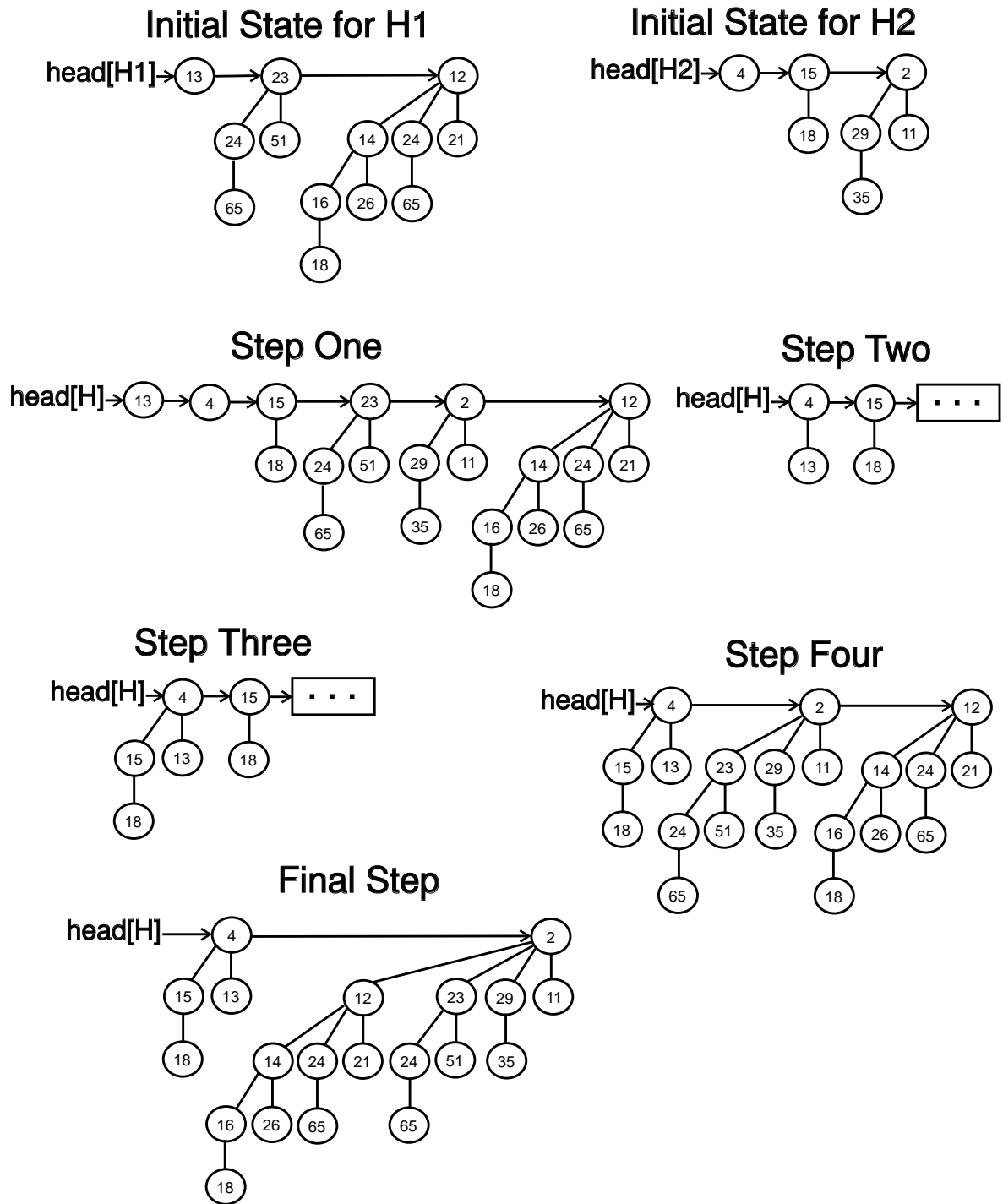


Figure 13: Merge of Binomial Queues