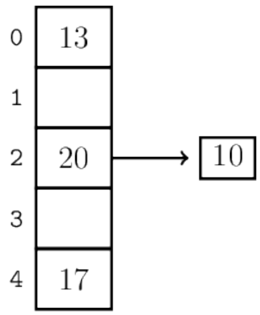
1. a)



b)

0	1	2	3	4
13		10	20	17

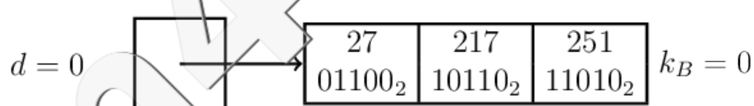
c)

0	1	2	3	4
13		10	20	17

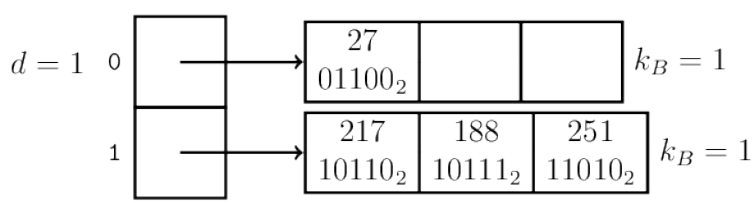
d)

0	1	2	3	4
13		10	17	20

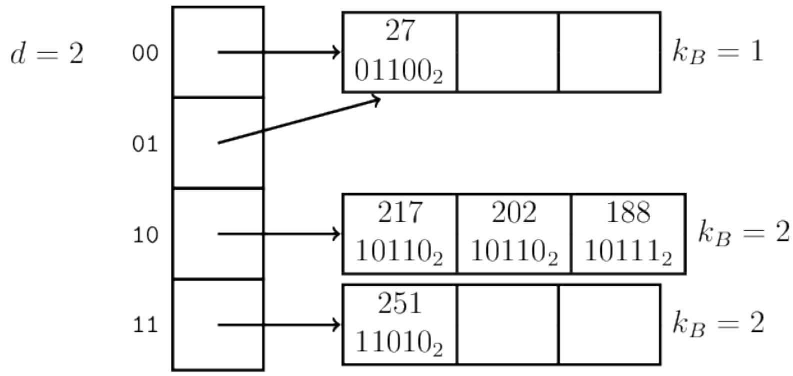
2. The first insertion is 251 with hash value $h(251) = 26 = 11010_2$. The second insertion is 217 with hash value $h(217) = 22 = 10110_2$. The third insertion is 27 with hash value $h(27) = 12 = 01100_2$. All these fit into the single block:



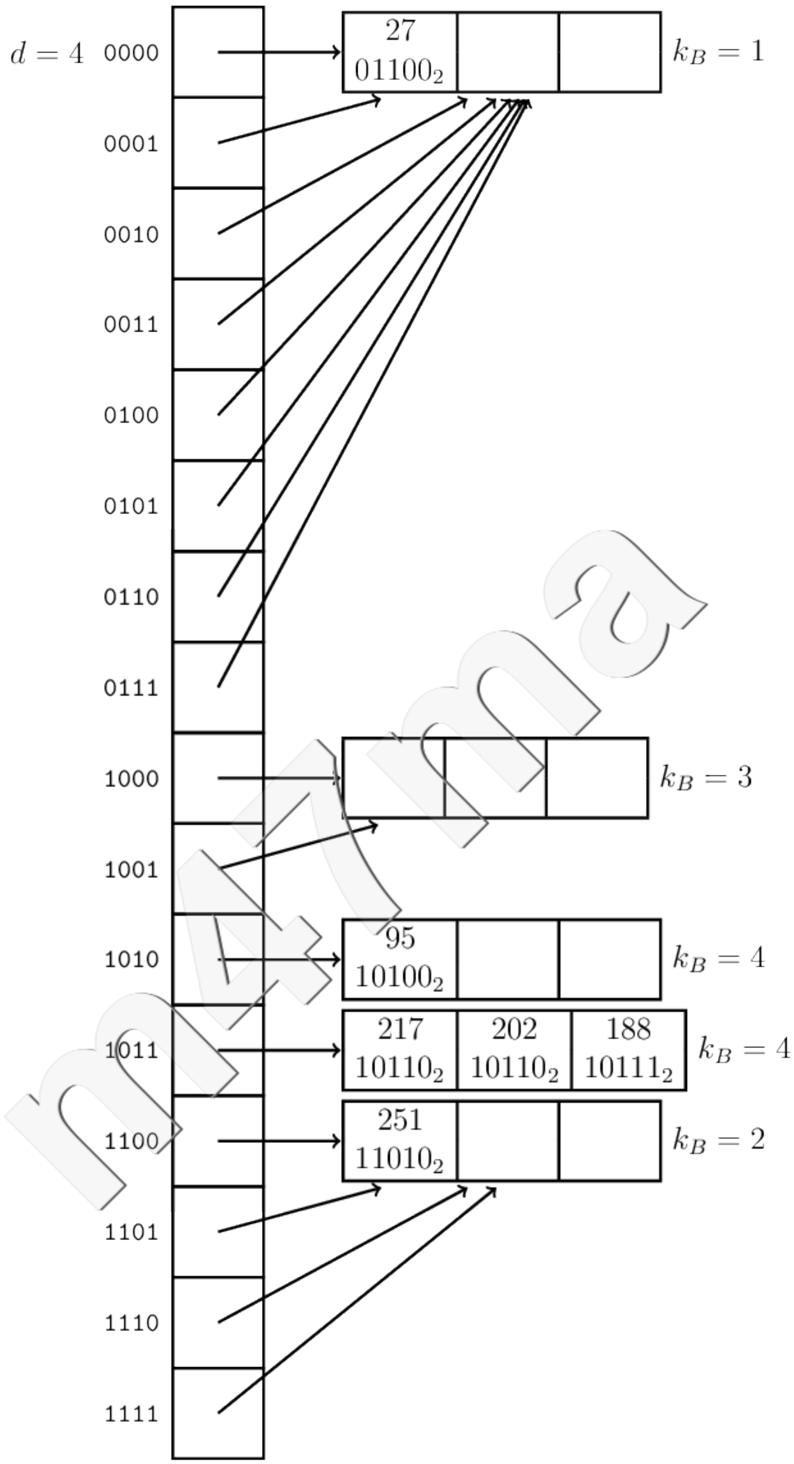
The fourth insertion is 188 with hash value $h(188) = 23 = 10111_2$.
This results in a block split and directory grow:



The fifth insertion is 202 with hash value $h(202) = 22 = 10110_2$.
This results in a directory grow and a block split:



The sixth insertion is 95 with hash value $h(95) = 20 = 10100_2$.
This results in **two** directory grows and a block splits:



3. **a)** The worst-case sequence of searches for the *move-to-front* heuristic consists of:

   – Searching for the last item in the list for the first r searches. Each search will be unique since $r \leq n$, and each costs n comparisons. (rn total work.)

– Searching for the r-th element in the list for the next $m - r$ searches. Each search costs r comparisons. ($r(m-r)$ total work.)

The total work is then $rn + r(m-r) = r(n+m) - r^2 \in \Theta(r(m+n))$.
To see that the Θ bound holds, notice that since $r \leq n, r \leq m, r > 0$:

$$r(m+n) - r^2 \leq r(m+n)$$

and

$$\begin{aligned} r(m+n) - r^2 &= r(m+n) - r \cdot r \\ &\geq r(m+n) - r\Big(\frac{m+n}{2}\Big) \\ &= \frac{1}{2}r(m+n) \end{aligned}$$

b) The worst-case sequence of searches for the *transpose* heuristic is to always search for the last item in the list. Provided $r > 1$, this sequence only looks for two different elements, alternating each time. Each search costs n comparisons and the total cost for m searches is $mn \in \Theta(mn)$.

Note: If $r = 1$, then worst case sequence is to look for the last item in the list (in the initial state) and then keep asking for the same item. If $m \leq n$ then the cost will be:

$$\sum_{i=n+1-m}^{n} i = nm + \frac{1}{2}m - \frac{1}{2}m^2 \in \Theta(mn)$$

When $m \geq n$, the cost will be:

$$\sum_{i=1}^{n} i + \sum_{i=n+1}^{m} 1 = \frac{1}{2}n(n+1) + m - n \in \Theta(n^2 + m).$$

4. We are providing the expected-case analysis for part (a) and the worst-case analysis for part(b).

a) Let p_i be the probability of searching for item k_i. We assume that $p_1 \geq p_2 \geq ... \geq p_n$.
The optimal ordering (if we knew the p_i values) would be $k_1, k_2, \ldots, k_n$, and the expected cost for this optimal ordering would be:

$$C_{OPT} = \sum_{j=1}^{n} jp_j$$

Assume that we are in a steady state. In the *move-to-back* heuristic, the expected cost will be

$$\begin{aligned}
C_{MTB} &= \sum_{j=1}^{n} p_j(\text{cost of finding } k_j) \\
&= \sum_{j=1}^{n} p_j(1 + \text{expected number of keys before } k_j)
\end{aligned}$$

What is the expected number of items before k_j in the *move-to-back* heuristic? This will be the same as the expected number of items **after** k_j in the *move-to-front* heuristic, and this is given by:

$$n - 1 - \sum_{i \neq j} \frac{p_i}{p_i + p_j}$$

Therefore, the expected cost for MTB will be:

$$\begin{aligned}
C_{MTB} &= \sum_{j=1}^{n} p_j\Big(1 + n - 1 - \sum_{i \neq j} \frac{p_i}{p_i + p_j}\Big) \\
&= \sum_{j=1}^{n} p_j n - \sum_{j=1}^{n}\Big(\sum_{i \neq j} \frac{p_i p_j}{p_i + p_j}\Big) \\
&= n - \sum_{j=1}^{n} 2\Big(\sum_{i<j} \frac{p_i p_j}{p_i + p_j}\Big) \\
&= n - 2\sum_{j=1}^{n} p_j\Big(\sum_{i<j} \frac{p_i}{p_i + p_j}\Big) \\
&\geq n - 2\sum_{j=1}^{n} p_j\Big(\sum_{i<j} 1\Big) \\
&= n - 2\sum_{j=1}^{n} p_j(j-1) \\
&= n - 2C_{OPT} + 2.
\end{aligned}$$

b) Consider a list of L items initially positioned as $[a_1, a_2, \ldots, a_L]$. For convenience, we assume L is an even integer. Consider a request sequence formed by repeating a subsequence $\sigma = \langle a_L, a_{L-1} \ldots, a_{L/2+1}, a_{L/2} \rangle$. MTF2 accesses all items at index L after serving σ, i.e., it has a cost of $L/2 \times L$. The positioning of items is the same as initial ordering; so, if we repeat requesting σ for m times, the cost of MTF2 would be $m \times L^2/2$.
Now, consider an algorithm that moves items $a_L, \ldots a_{L/2}$ to the front on their first access and does not move them after that. The cost of the algorithm for the first

subsequence σ would be $L \times L/2$ and for the next $m-1$ subsequence, it would be $L/2 + (L-1)/2 + \ldots + 2 + 1 = (L/2)(L/2+1)/2 \approx L^2/8$. So, for the ratio between the cost of MTF2 and that of an optimal algorithm for this sequence we would have $\frac{MTF2(\sigma)}{OPT(\sigma)} \geq \frac{m \times L^2/2}{L^2/2+(m-1)L^2/8}$, which converges to 4 for large values of m.

5. **a)**

```
S3  [-infty]-------------------------------------------[+infty]
       |                                                   |
S2  [-infty]--------------[47]--[51]--------------[68]--[+infty]
       |                    |     |                 |       |
S1  [-infty]--------------[47]--[51]--[53]--------[68]--[+infty]
       |                    |     |     |           |       |
S0  [-infty]--[15]--[36]--[47]--[51]--[53]--[54]--[68]--[+infty]
```

b) In addition to $\{-\infty, \infty\}$, S_2 should contain the keys with index

$$\{\lfloor n^{2/3} \rfloor, \lfloor 2n^{2/3} \rfloor, \ldots, \lfloor \lfloor n^{1/3} \rfloor n^{2/3} \rfloor\}.$$

In addition to $\{-\infty, \infty\}$, S_1 should contain the union of the keys in S_2 and the keys with index

$$\{\lfloor n^{1/3} \rfloor, \lfloor 2n^{1/3} \rfloor, \ldots, \lfloor \lfloor n^{2/3} \rfloor n^{1/3} \rfloor\}.$$

On each level, the search procedure will do at most $O(n^{1/3})$ moves to the right. The number of levels is constant. Thus the worst-case complexity is $O(n^{1/3})$.