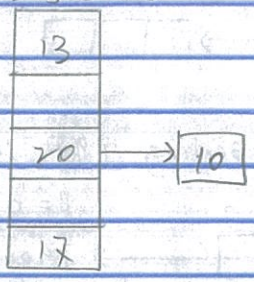
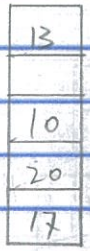
CS 240 Assignment 04 Xiao, Houze 20508228

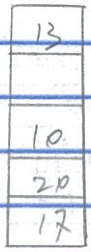
1 a). • $K=17$, $(17+2) \bmod 5 = 4$
• $K=10$ $(10+2) \bmod 5 = 2$
• $K=20$ $(20+2) \bmod 5 = 2$
• $K=13$ $(13+2) \bmod 5 = 0$



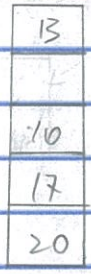
b) Linear probing:



c). Double Hashing with $h_1(K) = (K+2) \bmod 5$ & $h_2(K) = 1 + (K \bmod 4)$:



d). Cuckoo hashing with $h_1(K) = (K+2) \bmod 5$ & $h_2(K) = \lfloor K/5 \rfloor$
• $K=17$ $h_2(17) = \lfloor 17/5 \rfloor = 3$
• $K=10$ $h_2(10) = \lfloor 10/5 \rfloor = 2$
• $K=20$ $h_2(20) = \lfloor 20/5 \rfloor = 4$
• $K=13$ $h_2(13) = \lfloor 13/5 \rfloor = 2$



2. $h(K) = \lfloor K/16 \rfloor + (K \bmod 16)$
$h(191) = 11 + 15 = 26 = 11010_2$
$h(142) = 8 + 14 = 22 = 10110_2$
$h(192) = 12 + 0 = 12 = 01100_2$
$h(248) = 15 + 8 = 23 = 10111_2$
$h(217) = 13 + 9 = 22 = 10110_2$
$h(95) = 5 + 15 = 20 = 10100_2$

↳ d = 0 l = 0

↳ inserting 191, 142, 192

d = 0 l = 0

192	142	191
$(01100)_2$	$(10110)_2$	$(11010)_2$

↳ inserting 248

d = 1 l = 1

0
1

192		
$(01100)_2$		

142	248	191
$(10110)_2$	$(10111)_2$	$(11010)_2$

↳ inserting 217.

d = 2 l = 1

00
01
10
11

192		
$(01100)_2$		

l = 2

217	142	248
$(11011)_2$	$(10110)_2$	$(10111)_2$

l = 2

191		
$(11010)_2$		

↳ inserting 95.

d = 4 l = 1

0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111

192		
$(01100)_2$		

l = 3

l = 4

95		
$(10100)_2$		

l = 4

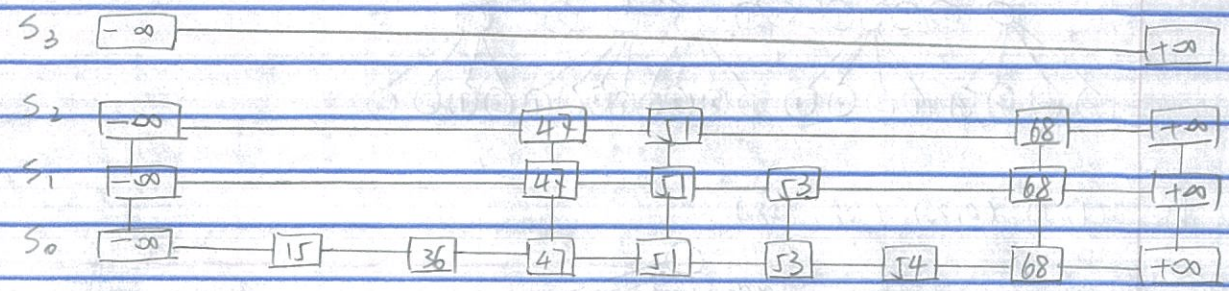
217	142	248
$(11011)_2$	$(10110)_2$	$(10111)_2$

l = 2

191		
$(11010)_2$		

3a) 54, 15, 51, 53, 47, 68, 36.

Skip List:



b). Let S_2 be $-\infty, \lfloor n^{2/3} \rfloor, \lfloor 2n^{2/3} \rfloor, \lfloor 3n^{2/3} \rfloor \cdots \lfloor \lfloor n^{1/3} \rfloor n^{2/3} \rfloor$

Let S_1 be $-\infty, \lfloor n^{1/3} \rfloor, \lfloor 2n^{1/3} \rfloor, \lfloor 3n^{1/3} \rfloor, \cdots, \lfloor \lfloor n^{2/3} \rfloor n^{1/3} \rfloor$.

$\therefore$ S_2 has $\lfloor n^{1/3} \rfloor + 2$ keys & S_1 has $\lfloor n^{2/3} \rfloor + 2$ keys.

Analysis:

In this way, search going through S_2 to its second last key takes $O(\lfloor n^{1/3} \rfloor)$ operations, then goes down. Search going through S_1 to its second last key takes $O(\lfloor n^{1/3} \rfloor)$ operations. For S_0, it takes $O(\lfloor n^{1/3} \rfloor)$ to find the last element.

Thus, $\begin{cases} T(n) = 3O(\lfloor n^{1/3} \rfloor) \in O(\lfloor n^{1/3} \rfloor) \\ T(n) = 3\lfloor n^{1/3} \rfloor + c \geq 3n^{1/3} - 3 \Rightarrow T(n) \in \Omega(n^{1/3}). \end{cases}$

$\therefore T(n) \in \Theta(n^{1/3})$ by property.

c). $Pr(Height = i) = (\frac{1}{4})^i \cdot \frac{3}{4}$

$E(Height) = \sum_{i=0} i \cdot Pr(Height = i) = \sum_{i=0} i \cdot (\frac{1}{4})^i \cdot \frac{3}{4} = \frac{3}{4} \sum_{i=0} i(\frac{1}{4})^i$

Let $S(i) = \sum_{i=0} i \cdot (\frac{1}{4})^i = 0 + \frac{1}{4} + \frac{2}{4^2} + \frac{3}{4^3} \cdots$

$\Rightarrow 4S(i) = 1 + \frac{2}{4} + \frac{3}{4^2} + \frac{4}{4^3} + \cdots$

$\therefore 4S(i) - S(i) = (1 + \frac{2}{4} + \frac{3}{4^2} + \frac{4}{4^3} + \cdots) -$

$(0 + \frac{1}{4} + \frac{2}{4^2} + \frac{3}{4^3} + \cdots)$

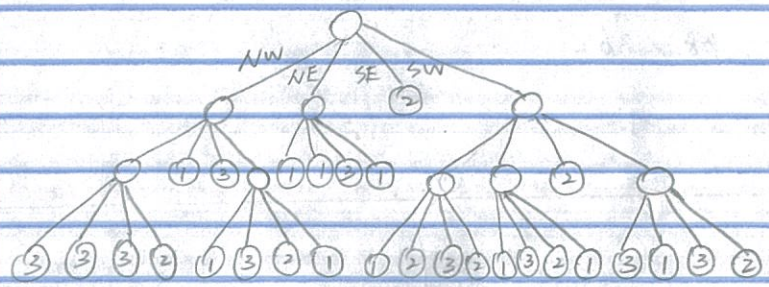
$= 1 + \frac{1}{4} + \frac{1}{4^2} + \frac{1}{4^3} + \cdots$

$= \frac{4}{3}$ by sum of geometric series

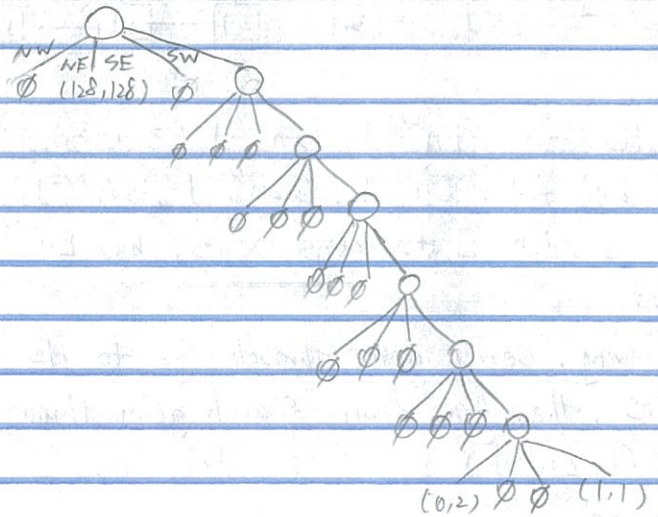
$\Rightarrow 3S(i) = \frac{4}{3} \Rightarrow S(i) = \frac{4}{9}$

$\therefore E(Height) = \frac{3}{4} \cdot \frac{4}{9} = \frac{1}{3}$

4 a).



b). (1, 1) (0, 2) (128, 128).



5 a). See the code

b). If the runtime is O(nlogn), the first method coming to my mind is partition. First, read all nodes and put into a vector. Make a copy of it.
Let V1 be the vector sorted by x-value.
Let V2 be the vector sorted by y-value.
If the turn is even, do { printKD (left)
print (middleNode) $\in O(n)$
printKD (right) }
If the turn is odd, do { printKD (left)
print (middleNode) $\in O(n)$.
printKD (right) }
Then use partition recursively. So.

$$T(n) = 2(T(\lfloor n/2 \rfloor) + O(n) \Rightarrow T(n) = O(n \log n).$$

Because I used built-in sorting fn. at beginning. So total runtime is:
$O(n \log n) + O(n \log n) \in O(n \log n)$

Because I used built-in sorting function at beginning, total runtime is:

$O(n \log n) + O(n \log n) \in O(n \log n)$.