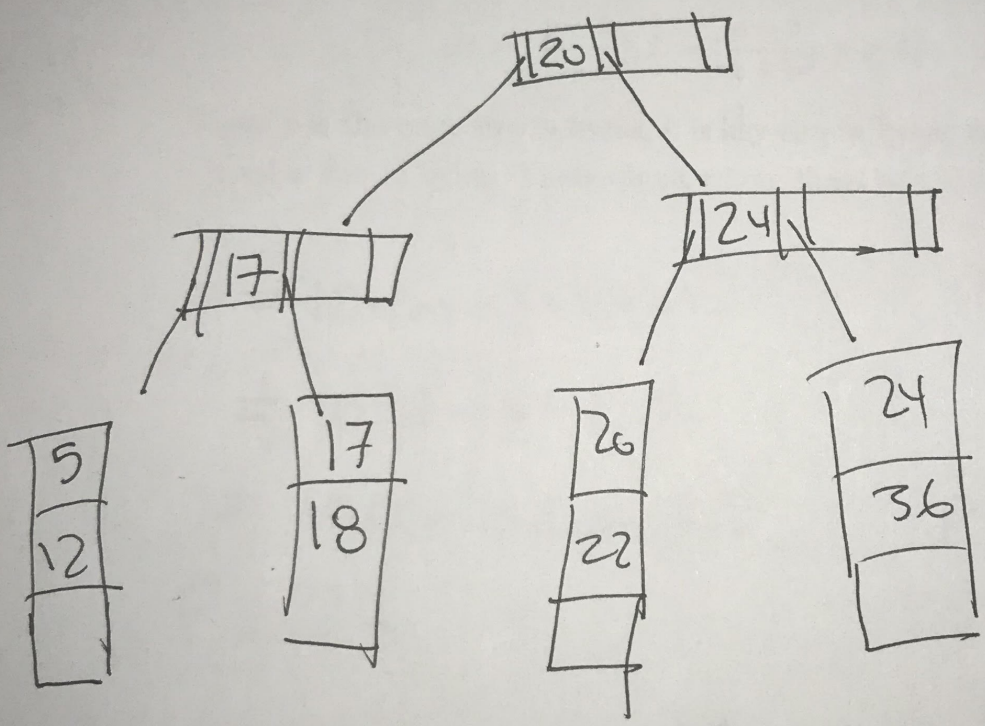
CSE 373: Section 5

Memory and B-Trees

October 25th

B-Trees

1. Insert the following into an empty B-Tree with $M = 3$ and $L = 3$: $12, 24, 36, 17, 18, 5, 22, 20$.



2. Given the following parameters for a B-Tree with $M = 11$ and $L = 8$

- Key Size = 10 bytes
- Pointer Size = 2 bytes
- Data Size = 16 bytes per record (includes the key)

Assuming that M and L were chosen appropriately, what is the likely page size on the machine where this implementation will be deployed? Give a numeric answer and a short justification based on two equations using the parameter values above. **Hint**: The three equations you will need to use are:

$$M = \lfloor \frac{p+k}{t+k} \rfloor, L = \lfloor \frac{p-t}{k+v} \rfloor, p \geq Mt + (M-1)k$$

Where p is the page size in bytes, k is key size in bytes, t is pointer size in bytes, and v is value size in bytes. Think about where these values come from

$P = 10 \cdot (m-1) + 2 \cdot m$

$= 10(11-1) + 2 \cdot 11$

$= 100 + 22 = 122$

$P \leq 122$

$P = 8 \cdot (16) + 2$

$= 128 + 2$

$= 130$

$P \leq 130$

page must be at least 130B

Memory

1. What are the two types of memory locality?

Spatial: memory is brought from disk and into cache in pages

Temporal: pages recently accessed will be accessed again

2. Does this more benefit arrays or linked lists?

This benefits ~~lists~~ arrays because java forces elements in an array to be stored together, enforcing spatial locality. Additionally, iterating through an array gives temporal locality

3. What about Java makes it a poor choice for implementing B-trees?

Java cannot page align its memory allocation

4. Provide and justify the bigO memory analysis for AVL insertion?
   Remember, what needs to be kept track of

Since AVL needs to retain the path from the added node to the root, insertion consumes O(log n) extra memory. This memory is consumed on the process stack