CS 3353 – Algorithms – Spring 2020 Online

Homework 2 – B+ Trees

Due April 16th at 11:59pm

1. If completely full, how many keys can be stored in the internal nodes of B+ Tree of order 5 with 4 levels (3 levels of internal nodes and 1 level of leaf nodes)? Support your answer.

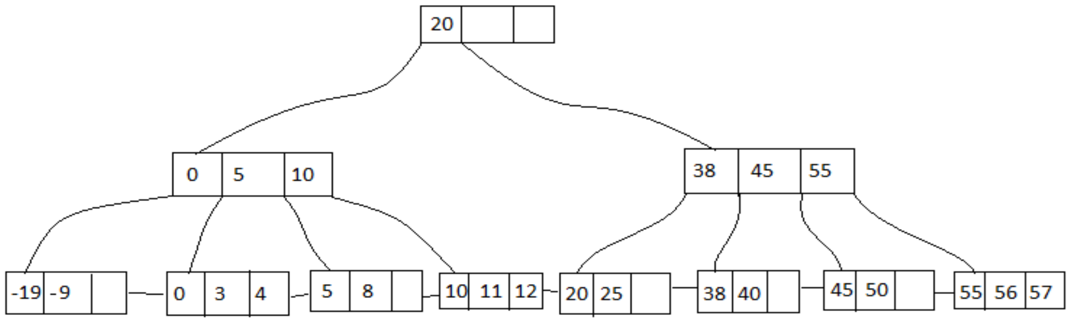
   Each internal node can have four keys, as that is one less than the order of the tree. Each level can have five times as many nodes as the previous level. So among three levels of internal nodes, there can be as many as 4 + 20 + 100 = 124 keys.

2. If minimally full, how many keys would be stored in the internal nodes a B+ Tree of order 7 with 5 levels (4 levels of internal nodes and 1 level of leaf nodes)? Support your answer

   A B+ tree of order seven requires at least three keys and four subtrees stored in each node. The first level would be a root node with at least one key, the second level must have at least two nodes with at least three keys each, and subsequent levels must store at least four times as many keys as the previous level. There are 1 + 6 + 24 + 96 = 127 keys.

3. Below is a B+ Tree. Insert the following values into the B+ Tree showing the resulting tree after any node splits.

   8, 25, 45, 50, -9, 11, 5, 4, 3, 55, 56, 57



4. Assume a B+ Tree node class has the following structure:

```
template<typename T, int M>
class BPlusTreeNode{
public:
   BPlusTreeNode();
   BPlustTreeNode(const T&);
private:
   bool isLeaf;
   int numKeys; //number of keys in this node currently
   T keys[M – 1];
   BPlusTreeNode *children[M];
   friend BPlusTree<T, M>;
};
```

Provide a pseudocode or c++ implementation of function
`BPlusTreeNode*& BPlusTree::findChildPointer(const T& k);`
that would locate and return the proper child pointer to follow for a given key value k.

```
findChildPointer(k)
     for i in 0 to numKeys-1
          if k < keys[i]
               return children[i]
     return children[numKeys]
```

5. Using the same BPlusTreeNode class from question 4, provide a pseudocode or c++ implementation of a function
`BPlusTreeNode* splitInternalNode(BPlusTreeNode*& curr, const T& k, T& keyToParent);`
that would process the splitting of an internal node of a BPlusTree. The function should return a pointer to the newly created node that contains the proper keys and child pointers. It accepts a pointer to a BPlusTreeNode curr that is currently full and would overflow if k was added, the new key k to be inserted, and a ref var keyToParent in which this function will place the key value to be integrated into the parent.

```
splitInternalNode(*curr, k, keyToParent):
     curr->numKeys /= 2
     if k < curr->keys[M/2]
          keyToParent = curr->keys[M/2]
     else if k > curr->keys[M/2 + 1]
          keyToParent = curr->keys[M/2 + 1]
     else
          keyToParent = k
     node = new BPlusTreeNode
     node->isLeaf = false
     node->numKeys = M – curr->numKeys - 1
     for i in (curr->numKeys) to (M – 2)
          node->keys[i – curr->numKeys] = curr->keys[i]
          node->children[i – curr->numKeys + 1 = curr->keys[i + 1]
     return node
```

6. This question is related to the project that will replace exam 2 (the final).

- Choose an area of algorithms that are of some interest to you.
- Choose a problem from that area that has:
  - a simple/trivial solution, but that only works on small data sets because of complexity.
  - identify two algorithms that solve the same problem but are faster than the simple/trivial solution
- Answer the questions below.

Exceptions: No sorting algorithms or any algorithms covered in 2341 or 3353.

Some places to find organized lists of Algorithms:
- https://en.wikipedia.org/wiki/List_of_algorithms
- Table of Contents of various algorithms books such as:
  - Introduction to Algorithms (Famous CLRS Book)
  - The Algorithm Design Manual (Steve Skiena)
  - MANY others
- A Curated List of Algorithms Resources
- NIST Dictionary of Algorithms and Data Structures

(Bullet points are fine for the questions below).
1. What is the area of algorithms you've chosen?
   Maximum flow algorithms

2. Succinctly describe the problem you've chosen in that area.
   Given a flow network, a weighted directed graph in which the edge weights represent flow capacity, what is the greatest possible flow from a source node to a sink node?

3. Succinctly describe the trivial solution and its complexity.
   Trying all possible flow values for each edge and seeing which combinations respect vertex conservation of flow and edge flow capacity; exponential time complexity.

4. Succinctly describe the two non-trivial solutions and their complexities
   The Ford-Fulkerson algorithm finds an initial path from the source to the sink and finding which edge in the path is bottlenecking the path. The algorithm then greedily attempts to increase flow by finding augmenting paths and updating bottlenecks. Time complexity is O(Ef), where E is the number of edges in the graph and f is the maximum flow of the network.

   Dinic's algorithm builds a level graph by performing a breadth first search and only including edges that make progress towards the sink. Depth first searches are then performed on this level graph, consuming available edge capacities, until a path no longer exists from the source to the sink. These steps are repeated until a level graph reaching the sink can no longer be generated, at which the maximum flow as been found. Time complexity is O(V^2E) where V is the number of vertices and E is the number of edges.