CS 225

Data Structures

March 1 – Btrees Analysis and Functions as Data

G Carl Evans

Btree Properties

A **BTrees** of order **m** is an m-way tree:

- All keys within a node are ordered
- All leaves contain no more than **m-1** keys.

- All internal nodes have exactly **one more child than keys**
- Root nodes can be a leaf or have **[2, m]** children.
- All non-root, internal nodes have **[ceil(m/2), m]** children.

- All leaves are on the same level

BTree Analysis

The height of the BTree determines maximum number of _____ possible in search data.

…and the height of the structure is: _____.

Therefore: The number of seeks is no more than _____.

…suppose we want to prove this!

BTree Analysis

In our AVL Analysis, we saw finding an upper bound on the height (given **n**) is the same as finding a lower bound on the nodes (given **h**).

We want to find a relationship for BTrees between the number of keys (**n**) and the height (**h**).

BTree Analysis

Strategy:

We will first count the number of nodes, level by level.

Then, we will add the minimum number of keys per node (**n**).

The minimum number of nodes will tell us the largest possible height (**h**), allowing us to find an upper-bound on height.

BTree Analysis

The minimum number of **nodes** for a BTree of order m **at each level**:

root:

level 1:

level 2:

level 3:

…

level h:

BTree Analysis

The **total number of nodes** is the sum of all of the levels:

BTree Analysis

The **total number of keys**:

BTree Analysis

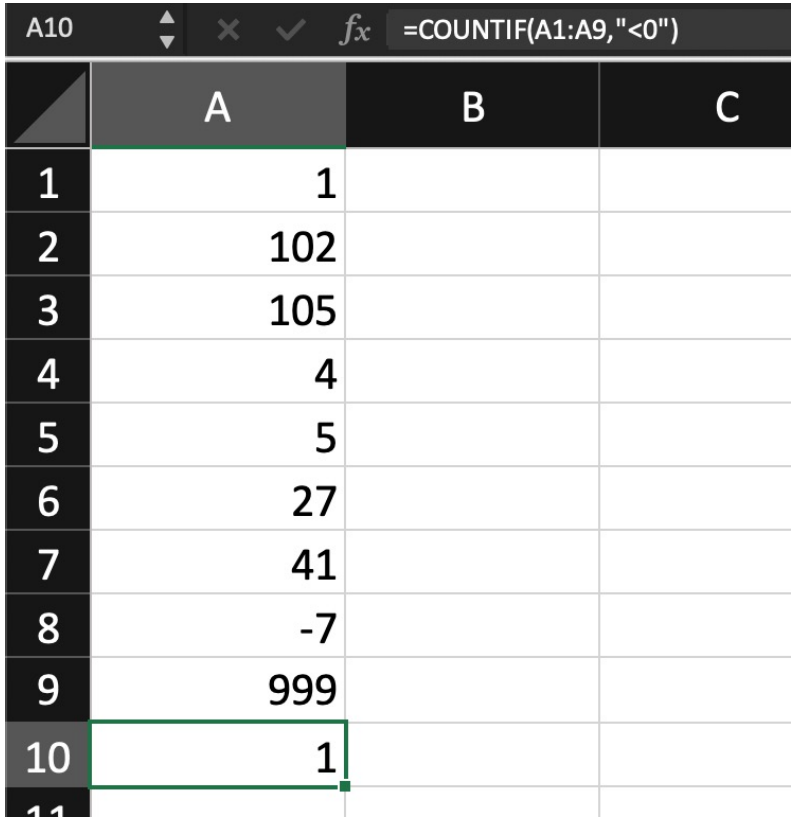
The **smallest total number of keys** is:

So an inequality about **n**, the total number of keys:

Solving for **h**, since **h** is the number of seek operations:

Functions As Data

Consider the function from Excel
COUNTIF(*range*, *criteria*)



COUNTIF in C++

Countif.hpp

```
10  template <typename Iter, typename Pred>
11  int Countif(Iter begin, Iter end, Pred pred) {
12    int count = 0;
13    auto cur = begin;
14
15    while(cur != end) {
16      if(pred(*cur))
17        ++count;
18      ++cur;
19    }
20
21    return count;
22  }
```

Ways to use Countif()

main.cpp

```
12 bool isNegative(int num) { return (num < 0); }
13
14 class IsNegative {
15 public:
16     bool operator() (int num) { return (num < 0); }
17 };
18
19 int main() {
20   std::vector<int> numbers = {1, 102, 105, 4, 5, 27, 41, -7, 999};
21
22   auto isnegl = [](int num) { return (num < 0); };
23   auto isnegfp = isNegative;
24   auto isnegfuctor = IsNegative();
25
26   std::cout << "There are " << Countif(numbers.begin(), numbers.end(), _______)
27     << " negative numbers" << std::endl;
```

Lambdas in C++ (functions with no name)

[ ]( ){ }

Power of the lambda

main.cpp

```
29    int big;
30    std::cout << "How big is big? ";
31    std::cin >> big;
32
33    auto isbig = [big](int num) { return (num >= big); };
34
35    std::cout << "There are " << Countif(numbers.begin(), numbers.end(), isbig)
36      << " big numbers" << std::endl;
37  }
38
```