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 $[\text{ceil}(m/2), m]$ 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++

```cpp
template <typename Iter, typename Pred>
int Countif(Iter begin, Iter end, Pred pred) {
    int count = 0;
    auto cur = begin;
    while(cur != end) {
        if(pred(*cur))
            ++count;
        ++cur;
    }
    return count;
}
```
Ways to use Countif()

```cpp
bool isNegative(int num) { return (num < 0); }

class IsNegative {
public:
    bool operator()(int num) { return (num < 0); }
};

int main() {
    std::vector<int> numbers = {1, 102, 105, 4, 5, 27, 41, -7, 999};

    auto isnegl = [](int num) { return (num < 0); };
    auto isnegfp = isNegative;
    auto isnegfuctor = IsNegative();

    std::cout << "There are " << Countif(numbers.begin(), numbers.end(), _______) << " negative numbers" << std::endl;
```
Lambdas in C++ (functions with no name)
int big;
std::cout << "How big is big? ";
std::cin >> big;

auto isbig = [big](int num) { return (num >= big); };

std::cout << "There are " << Countif(numbers.begin(), numbers.end(), isbig) << " big numbers" << std::endl;