CS 225

Data Structures

March 14 – BTree Analysis

Wade Fagen-Ulmschneider

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:

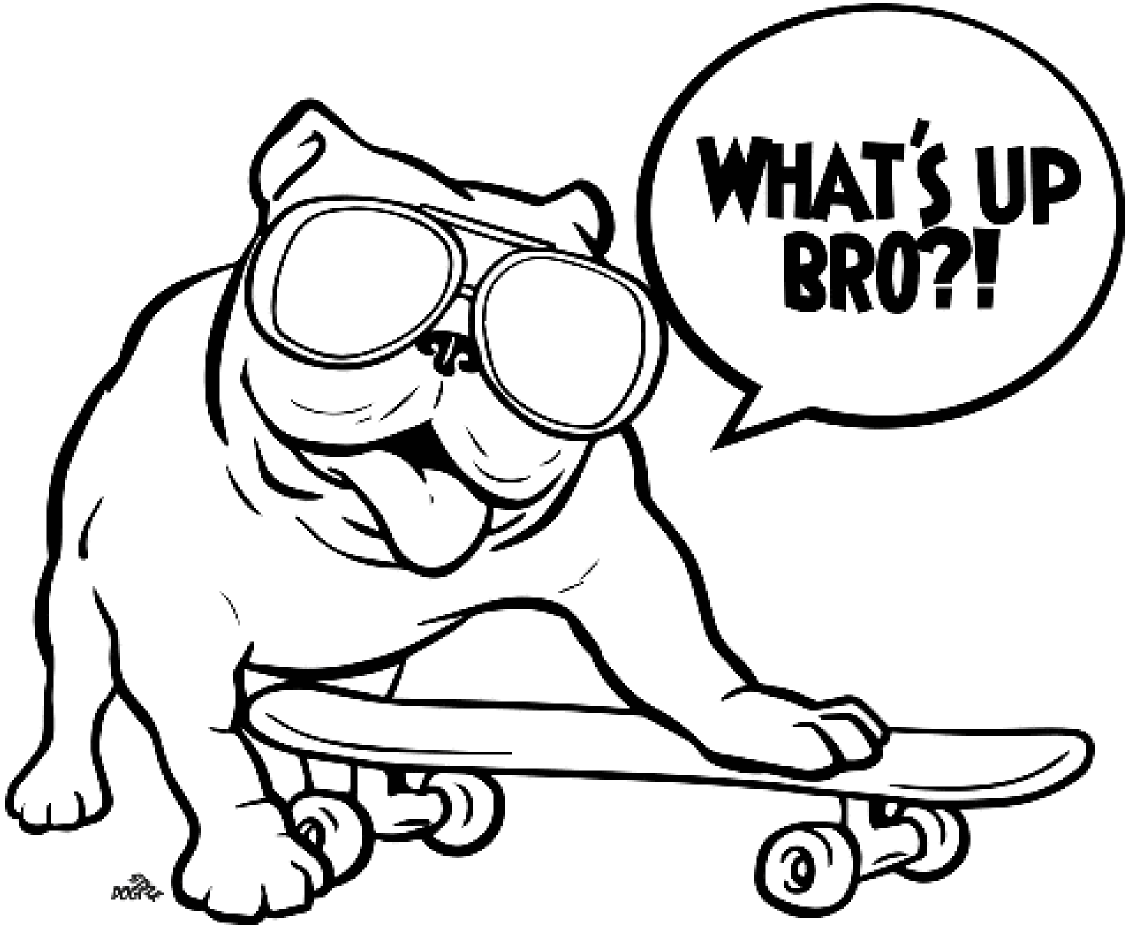
MP4 Animations

M

Making memes in paint
Making memes with Photoshop
Making memes with MP 2: Sticker Sheet
Making memes with MP 4: Flood Fill

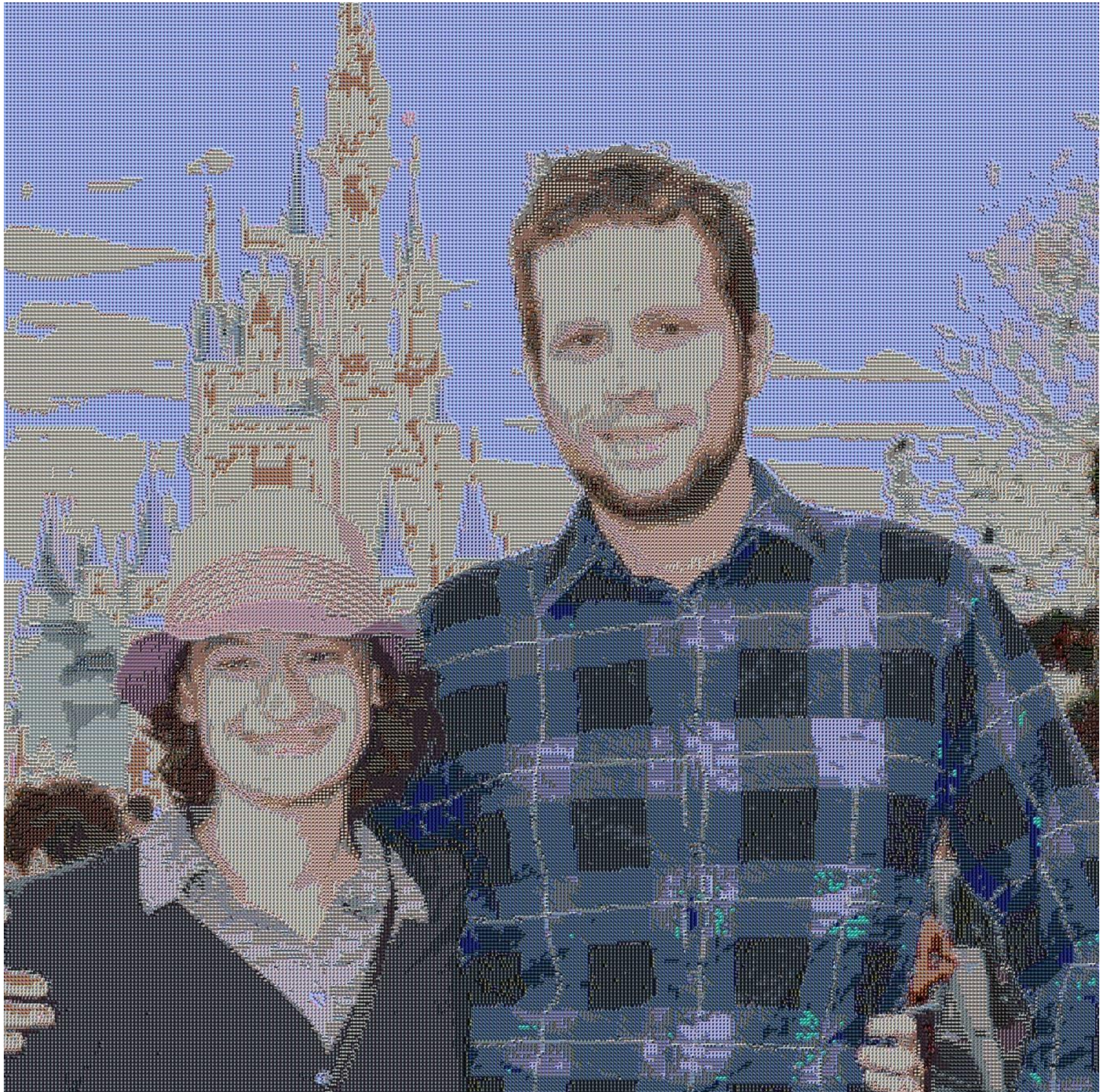
I finished MP4 and all I got was this gif.

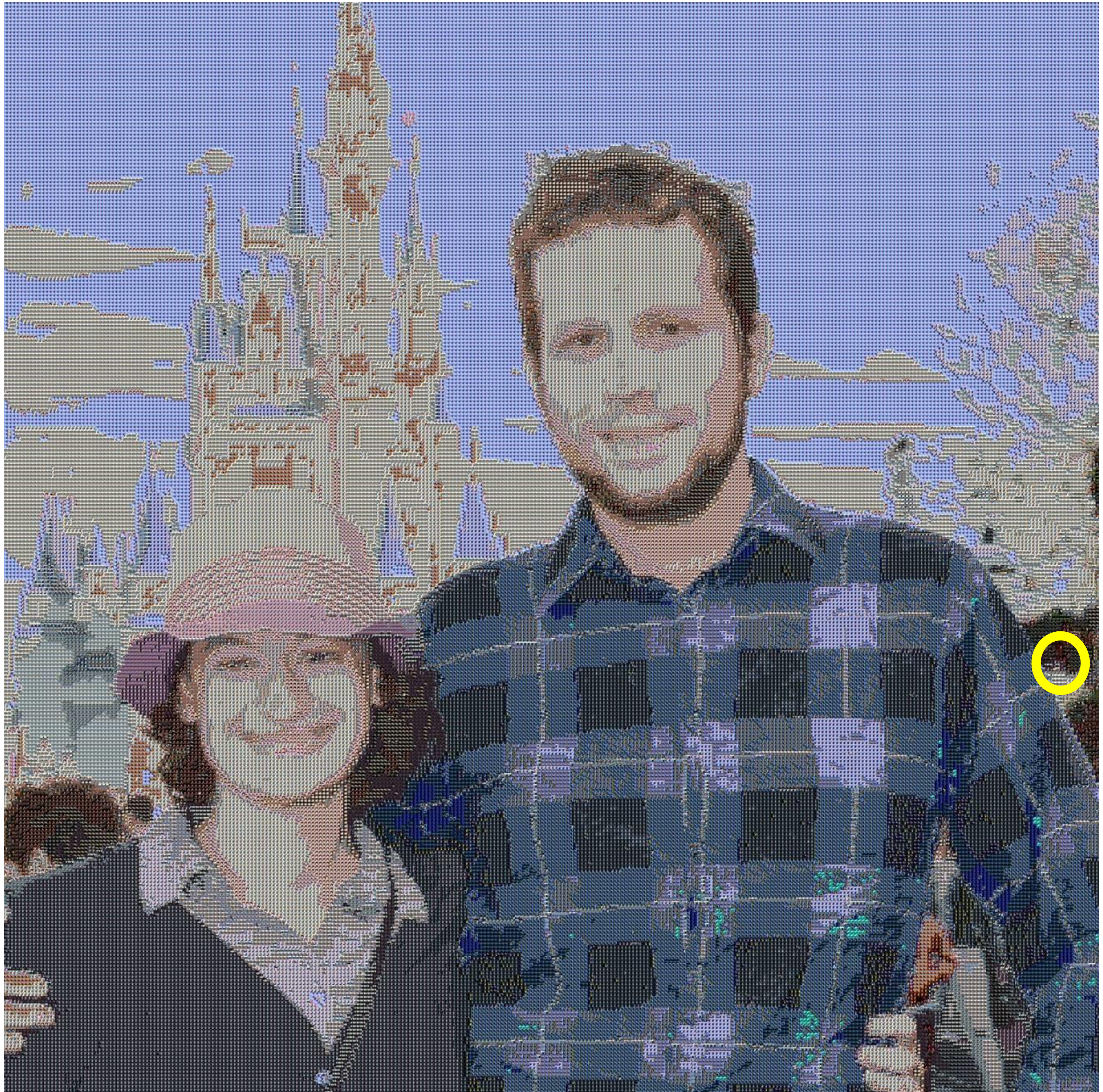
I ♥ GEO

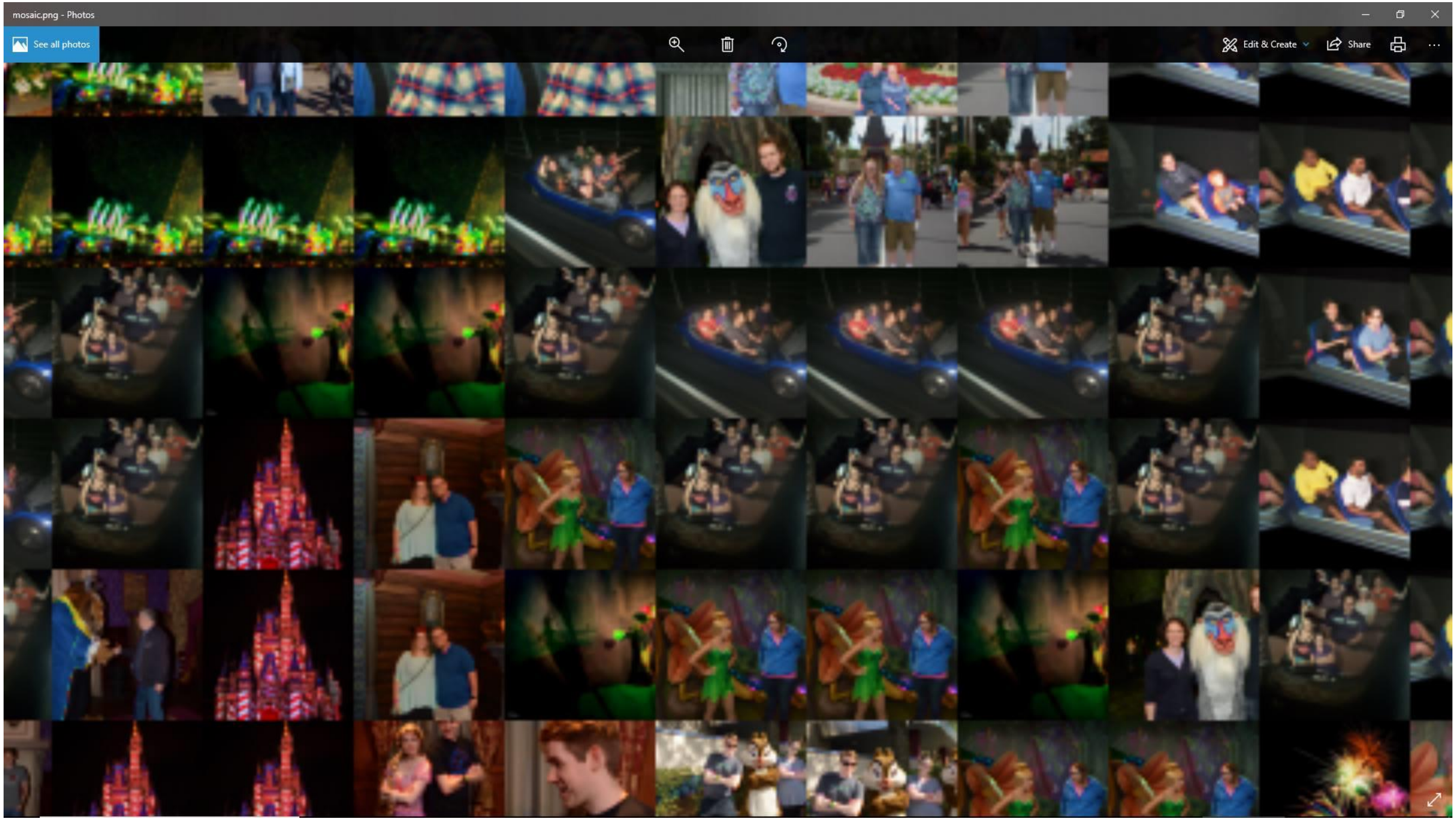

WHAT'S UP
BRO?!

```
cker/SolidColorPicker.o
clang++ -std=c++1y -stdlib=libc++ -g -O0 -pedantic -Wall -Werror -Wfatal-errors -Wextr
raversal/ImageTraversal.o
clang++ -std=c++1y -stdlib=libc++ -g -O0 -pedantic -Wall -Werror -Wfatal-errors -Wextr
S.o
clang++ -std=c++1y -stdlib=libc++ -g -O0 -pedantic -Wall -Werror -Wfatal-errors -Wextr
S.o
clang++ -std=c++1y -stdlib=libc++ -g -O0 -pedantic -Wall -Werror -Wfatal-errors -Wextr
clang++ -std=c++1y -stdlib=libc++ -g -O0 -pedantic -Wall -Werror -Wfatal-errors -Wextr
clang++ -std=c++1y -stdlib=libc++ -g -O0 -pedantic -Wall -Werror -Wfatal-errors -Wextr
lodepng.o
clang++ -std=c++1y -stdlib=libc++ -g -O0 -pedantic -Wall -Werror -Wfatal-errors -Wextr
t1.cpp
clang++ -std=c++1y -stdlib=libc++ -g -O0 -pedantic -Wall -Werror -Wfatal-errors -Wextr
clang++ -std=c++1y -stdlib=libc++ -g -O0 -pedantic -Wall -Werror -Wfatal-errors -Wextr
t2.cpp
clang++ Point.o FloodFilledImage.o Animation.o colorPicker/GridColorPicker.o colorPick
Picker/SolidColorPicker.o imageTraversal/ImageTraversal.o imageTraversal/BFS.o imageTr
ts/testmain.o tests/tests_part2.o -std=c++1y -stdlib=libc++ -lc++abi -lpthread -o test

Segmentation fault (core dumped)
```






mosaic.png - Photos
See all photos
Edit & Create
Share

BTree Analysis

Given **m=101**, a tree of height **h=4** has:

Minimum Keys:

Maximum Keys:

Hashing

Hashing

Goals:

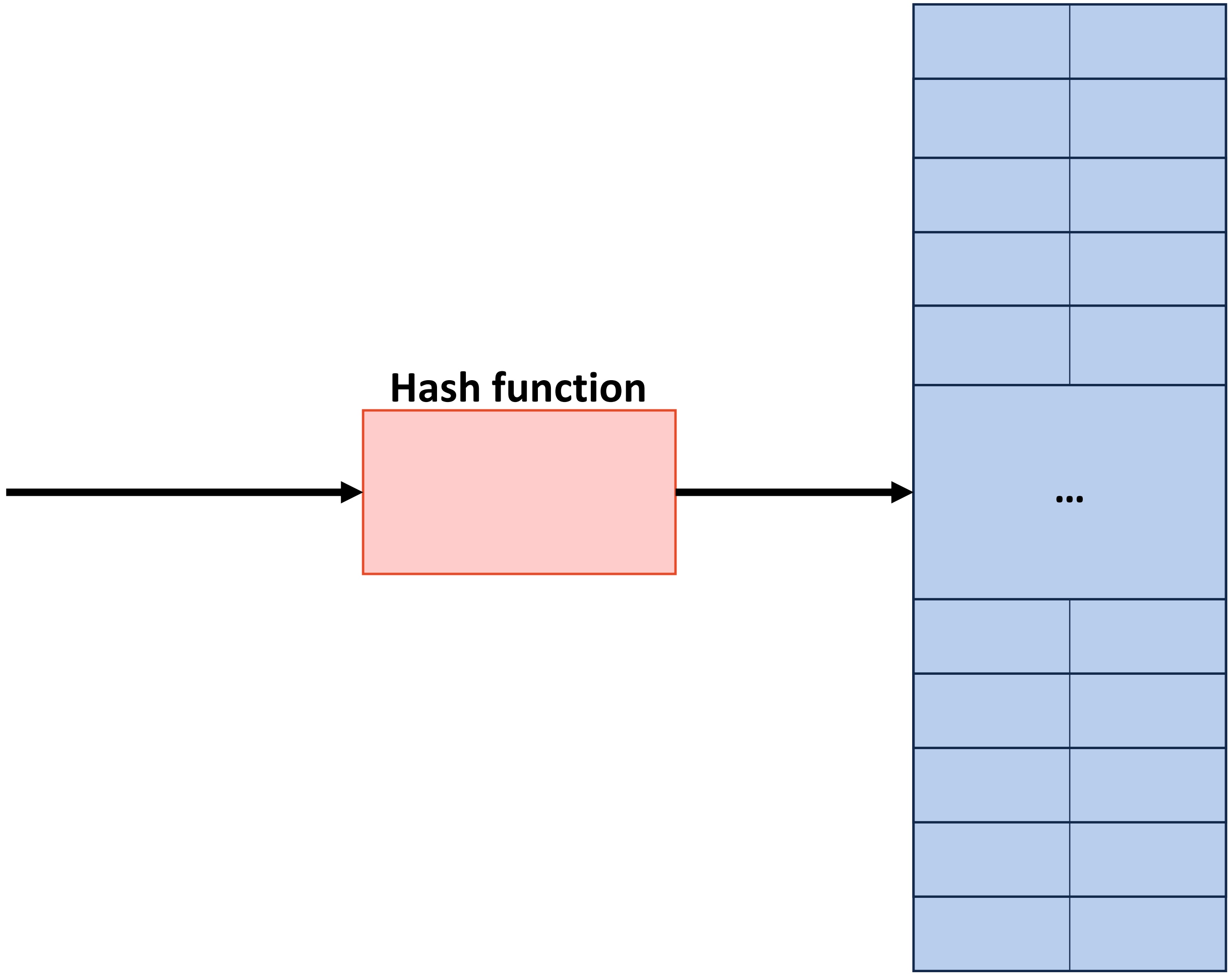
We want to define a **keyspace**, a (mathematical) description of the keys for a set of data.

…use a function to map the **keyspace** into a small set of integers.

Hashing

Locker Number	Name
103	
92	
330	
46	
124	

Hashing



A Hash Table based Dictionary

Client Code:

```
Dictionary<KeyType, ValueType> d;
d[k] = v;
```

A **Hash Table** consists of three things:

1.

2.

3.

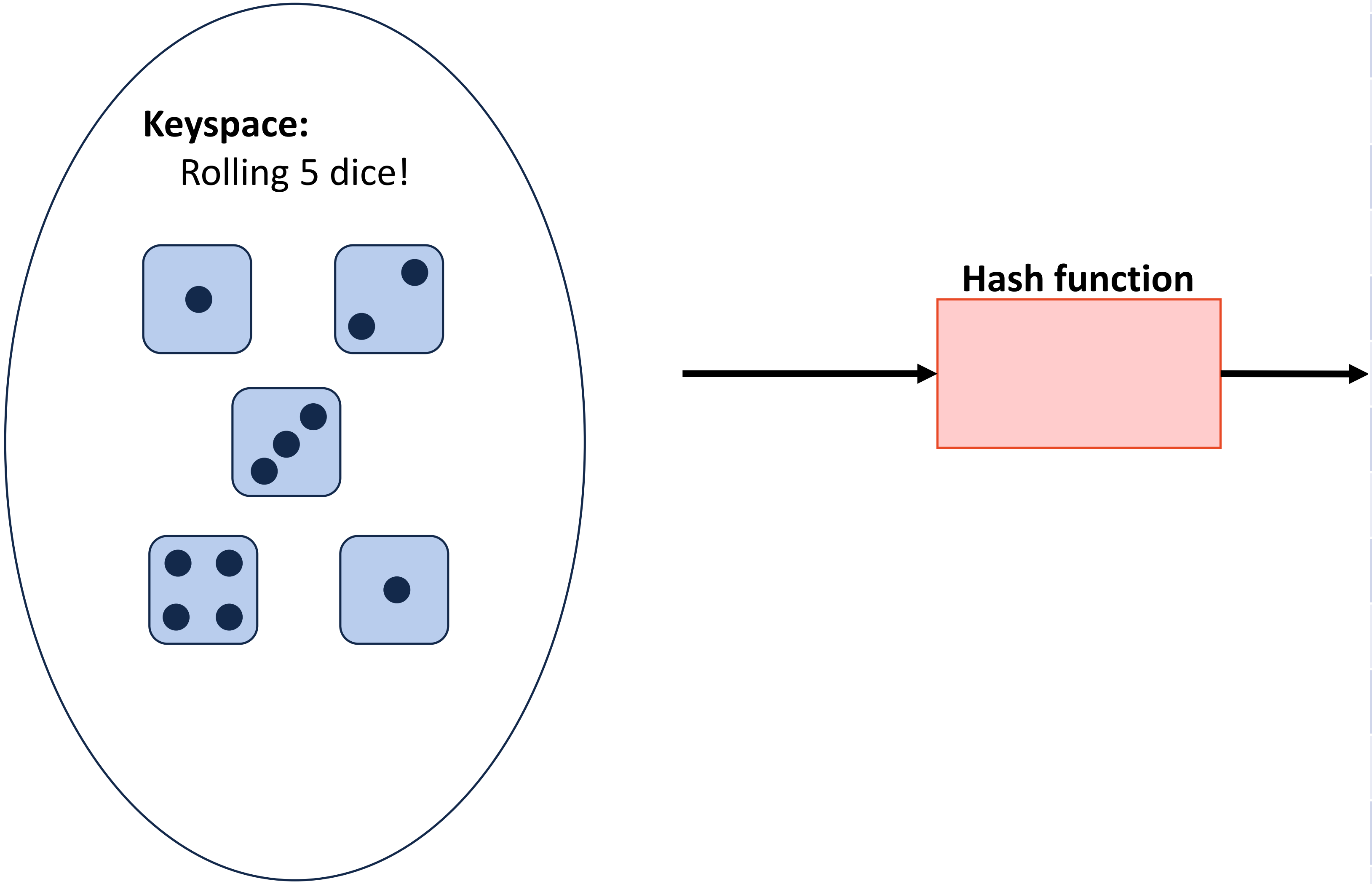
A Perfect Hash Function

(Angrave, CS 241)
(Beckman, CS 421)
(Cunningham, CS 210)
(Davis, CS 101)
(Evans, CS 126)
(Fagen-Ulmschneider, CS 225)
(Gunter, CS 422)
(Herman, CS 233)

Hash function

Key	Value

A Perfect Hash Function



Key	Value
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	